[ENGLISH]

Hello, Cloud Gurus, and welcome to this lecture,

which is going to cover SQS or Simple Queue Service.

And SQS is actually the first ever AWS service that was publicly available,

so it's one of the oldest services that they have,

but you only need to know about it at a very high level.

So what is SQS? Well, it's basically a message queue service.

So it gives you access to a message queue,

which can be used to store messages while waiting for an application,

or a computer to process them.

So it enables web service applications to quickly and

reliably queue messages,

that one component in the application generates,

for another component to consume.

But what do we mean when we say a queue? Well,

a queue is a temporary repository for messages which are awaiting

processing. Now, if all of that sounds a bit abstract,

don't worry because the easiest way to understand it is if I show you a couple

of examples. So my first example is a meme website.

So here's our user and they want to upload an image which is going

to be turned into a meme.

So here's a picture of me and Ryan at the ACG office in London.

So the photo is going to be uploaded to our website and our website is going to

store the photo in an S3 bucket.

So once the photo is uploaded to S3, that is going to trigger a Lambda function,

and

the Lambda function is going to send all the data associated with this

image into SQS.

So this data might include the words that we want to add to the photo,

maybe the color of the text, the position on the photo,

where we want to display the text, maybe the name of the S3 bucket,

where it's stored,

and maybe the name of the user who owns the image and all this

data can be enclosed in a message and sent to SQS.

And this message will sit in SQS, waiting to be processed.

Then you might have a fleet of EC2 instances,

and what they do is poll the SQS queue,

looking for messages, or essentially looking for the next task to complete.

So you can think of SQS as a way of assigning jobs

or tasks for an application to complete.

So a job goes into the queue in the form of a message

and your EC2 instances constantly poll that queue looking

for work.

So they find this message and then they set to work creating the meme.

They add the text to the photo,

and then the EC2 instance then stores the meme in S3.

And once it's finished that job, it goes back to the SQS queue again,

and looks for the next job to do.

And you can actually configure auto scaling groups of EC2

instances,

which automatically scale based on the number of messages currently in

the SQS queue.

So if you get a massive surge of messages in the queue,

this can trigger the auto scaling group to launch additional instances,

to cope with the increased number of messages or jobs to complete.

And if you have a reduced number of messages in the queue,

auto scaling can also reduce the number of EC2 instances

needed to process the messages. Now,

the great thing about this kind of architecture is that if you lose an EC2

instance before the job has been completed successfully,

before it managed to complete the task and save the meme to our S3 bucket,

then you won't lose the job itself because it's still going to be in the queue.

So if we lose our EC2 instance,

then another instance can just go in and pick up the message from the queue and

start processing it. So let's now take a look at a different example,

and this time it's a travel website and here is our user and

she is searching for a holiday.

So she's looking for a beach holiday and she wants to get the best possible

flights available. So our user heads to the travel website,

types in their search criteria, so the destination, preferred dates,

the city that they're traveling from,

and this time our website is running on EC2 instances,

so these are our web servers and the web servers will

take the users search criteria and package it up as a message

in an SQS queue.

And then the queue then gets polled by another EC2 instance.

So this is our application server,

and it's actually polling the messages down from the queue,

and this is key to SQS.

So any application which is going to consume the messages needs to poll the

queue to get the messages.

And you may well come across this in the exam where they give you a scenario

and they ask you to select a pull-based system or a push-based system for

messaging between application components.

So if they are asking you for a pull-based service,

then they're usually talking about SQS. However,

if they are asking you about a push-based system,

then you'll need something like a Simple Notification Service or SNS,

which is another service that we're going to be covering later on in this

section of the course. So we've got our EC2 instances here,

and they are constantly pulling the queue,

looking for new messages and looking for the next job to do.

So, they pick up the message as it comes in and begin to process it.

So let's say our user wants to fly from London to Italy.

Our application server is going to start sending queries to all the different

airlines,

to find the best flights based on our users search criteria.

Once it's received a response from all the airlines,

it then sends the information back to our web server,

which provides everything to our user so that she can select the flights she

likes the best. And once again,

the great thing about SQS is that if any of our application servers crashed,

any messages, which are partially processed,

will not be deleted and will end up back in the queue.

And this is because when a message is first picked up by our application server,

it's going to be marked as invisible so that no other application server can

start processing that message. And this is called the visibility timeout,

and this is the amount of time that the application server gets to

process the message, and as soon as that window has expired,

if the message has not been completely processed,

then it will appear in the queue again and another application server will pick

up that message and begin processing it.

So that is the great thing about SQS. And by using SQS,

you're removing dependencies between individual components in your application.

So your application server can fail,

but the job itself and the information held in our user's query is not

lost and they call this decoupling your infrastructure.

So if you see anything in the exam referring to decoupling your

infrastructure, then they're talking about SQS.

So using SQS, you can decouple your application components.

So you can decouple the components of an application,

allowing them to run independently and easing message management between

components. For example,

if you have one component which is running a lot more quickly than another,

or you're creating jobs faster than your application servers can process them,

then having SQS in the middle,

make sure that those components can run independently and that one doesn't get

overloaded,

and that jobs and tasks are not lost due to your application

servers being overloaded.

Any component of a distributed application can store messages in the queue

and messages can contain up to 256 kilobytes of text

in any format. For example, XML, JSON,

or plain text format and any component can later

retrieve the messages,

programmatically using the Amazon SQS API.

And SQS effectively act as a buffer between the components of your

application.

So SQS acts as a buffer between the component receiving the data

for processing and the component producing the data and saving the

data. And in our main website example, the component receiving the data,

it was of course our Lambda function and the component producing and saving the

data was our EC2 instance.

So with a complex distributed application using SQS helps to

resolve scheduling issues between at the various different components.

So the queue resolves issues,

which arise if the producer is producing work faster than the consumer can

process it,

or if the producer and consumer are only intermittently connected to the

network.

So it resolves issues if one of your application components is running slowly,

if an application server fails,

or if you're reaching out to an external network and network connectivity gets

interrupted. Key facts to remember about SQS,

are that is is pull-based. It's not pushed-based.

The messages can be up to 256 kilobytes in size.

Messages are always text data, including XML, JSON,

and unformatted plain text.

It guarantees that the messages will be processed at least once.

The retention period of messages is up to 14 days.

So messages can be kept in the queue from between one minute to 14 days.

So the minimum is for one minute and the maximum is 14 days,

but the default retention period is four days.

So moving on to my exam tips for SQS,

and the main thing to remember for this lecture is that

SQS is a distributed message queuing system,

and it allows us to decouple the components of an application so that they are

independent. And if 1 of our application servers crashes,

before they complete a task,

you're not going to lose the task because it will reappear in the queue.

And finally, remember that SQS is pull-based, not push-based.

So our application servers are pulling the queue and pulling down the messages

they're not being pushed out. So that is SQS at a high-level. And don't worry,

you only really need to know it at a very high-level for the exam.

If you have any questions, please let me know. Otherwise,

feel free to move on to the next lecture. Thank you.

[SPANISH]

Hola, Cloud Gurus, y bienvenidos a esta conferencia.

que cubrirá SQS o Simple Queue Service.

Y SQS es en realidad el primer servicio de AWS que estuvo disponible públicamente,

entonces es uno de los servicios más antiguos que tienen,

pero solo necesitas saberlo a un nivel muy alto.

Entonces, ¿qué es SQS? Bueno, es básicamente un servicio de cola de mensajes.

Entonces te da acceso a una cola de mensajes,

que se puede utilizar para almacenar mensajes mientras se espera una aplicación,

o una computadora para procesarlos.

Por lo tanto, permite que las aplicaciones de servicios web funcionen rápida y

mensajes en cola confiables ,

que genera un componente en la aplicación,

para que otro componente consuma.

Pero, ¿a qué nos referimos cuando decimos una cola? Bien,

una cola es un depósito temporal para los mensajes que están esperando

procesamiento \_ Ahora, si todo eso suena un poco abstracto,

no te preocupes porque la forma más fácil de entenderlo es si te muestro un par

de ejemplos Así que mi primer ejemplo es un sitio web de memes.

Así que aquí está nuestro usuario y quiere subir una imagen que va

convertirse en un meme.

Así que aquí hay una foto mía y de Ryan en la oficina de ACG en Londres.

Entonces, la foto se cargará en nuestro sitio web y nuestro sitio web se

almacenar la foto en un cubo S3.

Entonces, una vez que la foto se cargue en S3, eso activará una función Lambda,

y

la función Lambda va a enviar todos los datos asociados con este

imagen en SQS.

Entonces, estos datos pueden incluir las palabras que queremos agregar a la foto,

tal vez el color del texto, la posición en la foto,

donde queremos mostrar el texto, tal vez el nombre del cubo S3,

donde se almacena,

y tal vez el nombre del usuario que posee la imagen y todo esto

los datos pueden encerrarse en un mensaje y enviarse a SQS.

Y este mensaje permanecerá en SQS, a la espera de ser procesado.

Entonces podría tener una flota de instancias EC2,

y lo que hacen es sondear la cola de SQS,

buscando mensajes, o esencialmente buscando la siguiente tarea para completar.

Entonces puede pensar en SQS como una forma de asignar trabajos

o tareas para completar una aplicación.

Entonces, un trabajo entra en la cola en forma de mensaje.

y sus instancias EC2 sondean constantemente esa cola buscando

para el trabajo

Entonces encuentran este mensaje y luego se ponen a trabajar creando el meme.

Agregan el texto a la foto,

y luego la instancia EC2 almacena el meme en S3.

Y una vez que ha terminado ese trabajo, vuelve a la cola SQS nuevamente,

y busca el próximo trabajo que hacer.

Y en realidad puede configurar grupos de escalado automático de EC2

instancias ,

que se escalan automáticamente en función del número de mensajes actualmente en

la cola SQS.

Entonces, si recibe una oleada masiva de mensajes en la cola,

esto puede activar el grupo de escalado automático para lanzar instancias adicionales,

para hacer frente a la mayor cantidad de mensajes o trabajos para completar.

Y si tiene un número reducido de mensajes en la cola,

automático también puede reducir la cantidad de instancias EC2

necesarios para procesar los mensajes. Ahora,

lo mejor de este tipo de arquitectura es que si pierde un EC2

instancia antes de que el trabajo se haya completado con éxito,

antes de que lograra completar la tarea y guardar el meme en nuestro cubo S3,

entonces no perderá el trabajo en sí porque todavía estará en la cola.

Entonces, si perdemos nuestra instancia EC2,

entonces otra instancia puede entrar y recoger el mensaje de la cola y

empezar a procesarlo. Así que ahora echemos un vistazo a un ejemplo diferente,

y esta vez es un sitio web de viajes y aquí está nuestro usuario y

ella está buscando unas vacaciones.

Ella está buscando unas vacaciones en la playa y quiere obtener lo mejor posible.

vuelos disponibles. Así que nuestro usuario se dirige al sitio web de viajes,

escribe en sus criterios de búsqueda, por lo que el destino, las fechas preferidas,

la ciudad desde la que viajan,

y esta vez nuestro sitio web se ejecuta en instancias EC2,

estos son nuestros servidores web y los servidores web

tomar los criterios de búsqueda de los usuarios y empaquetarlos como un mensaje

en una cola SQS.

Y luego la cola es encuestada por otra instancia de EC2.

Así que este es nuestro servidor de aplicaciones,

y en realidad está sondeando los mensajes de la cola,

y esto es clave para SQS.

Entonces, cualquier aplicación que vaya a consumir los mensajes necesita sondear el

cola para recibir los mensajes.

Y es posible que te encuentres con esto en el examen donde te den un escenario.

y le piden que seleccione un sistema basado en extracción o un sistema basado en inserción para

mensajería entre los componentes de la aplicación.

Entonces, si le piden un servicio basado en extracción,

entonces generalmente están hablando de SQS. Sin embargo,

si le preguntan sobre un sistema basado en push,

entonces necesitará algo como un Servicio de notificación simple o SNS,

que es otro servicio que cubriremos más adelante en este

sección del curso. Aquí tenemos nuestras instancias EC2,

y están constantemente tirando de la cola,

buscando nuevos mensajes y buscando el próximo trabajo que hacer.

Por lo tanto, recogen el mensaje a medida que llega y comienzan a procesarlo.

Entonces, digamos que nuestro usuario quiere volar de Londres a Italia.

Nuestro servidor de aplicaciones comenzará a enviar consultas a todos los diferentes

aerolíneas ,

para encontrar los mejores vuelos en base a los criterios de búsqueda de nuestros usuarios.

Una vez que haya recibido una respuesta de todas las aerolíneas,

luego envía la información a nuestro servidor web,

que le proporciona todo a nuestra usuaria para que pueda seleccionar los vuelos que

le gusta lo mejor. Una vez mas,

lo bueno de SQS es que si alguno de nuestros servidores de aplicaciones falla,

cualquier mensaje, que se procesa parcialmente,

no se eliminará y terminará de nuevo en la cola.

Y esto se debe a que cuando nuestro servidor de aplicaciones recoge un mensaje por primera vez,

se marcará como invisible para que ningún otro servidor de aplicaciones pueda

empezar a procesar ese mensaje. Y esto se llama el tiempo de espera de visibilidad,

y esta es la cantidad de tiempo que el servidor de aplicaciones llega a

procesar el mensaje, y tan pronto como esa ventana haya expirado,

si el mensaje no ha sido completamente procesado,

luego volverá a aparecer en la cola y otro servidor de aplicaciones seleccionará

ese mensaje y comenzar a procesarlo.

Eso es lo bueno de SQS. Y al usar SQS,

está eliminando dependencias entre componentes individuales en su aplicación.

Entonces su servidor de aplicaciones puede fallar,

pero el trabajo en sí y la información contenida en la consulta de nuestro usuario no es

perdido y a esto lo llaman desacoplar su infraestructura.

Entonces, si ve algo en el examen que se refiera a desacoplar su

infraestructura , entonces están hablando de SQS.

Entonces, al usar SQS, puede desacoplar los componentes de su aplicación.

Para que pueda desacoplar los componentes de una aplicación,

permitiéndoles ejecutarse de forma independiente y facilitando la gestión de mensajes entre

componentes \_ Por ejemplo,

si tiene un componente que se ejecuta mucho más rápido que otro,

o está creando trabajos más rápido de lo que sus servidores de aplicaciones pueden procesarlos,

luego tener SQS en el medio,

asegúrese de que esos componentes puedan ejecutarse de forma independiente y que uno no obtenga

sobrecargado ,

y que no se pierdan trabajos y tareas por su aplicación

servidores sobrecargados.

Cualquier componente de una aplicación distribuida puede almacenar mensajes en la cola

y los mensajes pueden contener hasta 256 kilobytes de texto

en cualquier formato. Por ejemplo, XML, JSON,

o formato de texto sin formato y cualquier componente puede luego

recuperar los mensajes,

programáticamente usando la API de Amazon SQS.

Y SQS actúa efectivamente como un amortiguador entre los componentes de su

aplicación \_

Entonces SQS actúa como un búfer entre el componente que recibe los datos

para el procesamiento y el componente que produce los datos y guarda los

datos \_ Y en nuestro ejemplo de sitio web principal, el componente que recibe los datos,

por supuesto, era nuestra función Lambda y el componente que producía y guardaba el

data era nuestra instancia EC2.

Entonces, con una aplicación distribuida compleja, el uso de SQS ayuda a

resolver problemas de programación entre los diferentes componentes.

Así que la cola resuelve problemas,

que surgen si el productor está produciendo trabajo más rápido de lo que el consumidor puede

procesarlo ,

o si el productor y el consumidor sólo están conectados intermitentemente a la

red \_

Por lo tanto, resuelve problemas si uno de los componentes de su aplicación se ejecuta lentamente,

si un servidor de aplicaciones falla,

o si se comunica con una red externa y la conectividad de la red se vuelve

interrumpido \_ Datos clave para recordar sobre SQS,

son que se basa en la extracción. No está basado en push.

Los mensajes pueden tener un tamaño de hasta 256 kilobytes.

Los mensajes son siempre datos de texto, incluidos XML, JSON,

y texto sin formato sin formato.

Garantiza que los mensajes serán procesados al menos una vez.

El período de retención de los mensajes es de hasta 14 días.

Por lo tanto, los mensajes se pueden mantener en la cola desde un minuto hasta 14 días.

Así que el mínimo es de un minuto y el máximo es de 14 días,

pero el período de retención predeterminado es de cuatro días.

Pasando a mis consejos para el examen de SQS,

y lo principal a recordar para esta conferencia es que

SQS es un sistema de colas de mensajes distribuidos,

y nos permite desacoplar los componentes de una aplicación para que sean

independiente \_ Y si uno de nuestros servidores de aplicaciones falla,

antes de completar una tarea,

vas a perder la tarea porque volverá a aparecer en la cola.

Y, por último, recuerde que SQS se basa en la extracción, no en la inserción.

Entonces, nuestros servidores de aplicaciones están tirando de la cola y bajando los mensajes.

están siendo expulsados. Así que eso es SQS a un alto nivel. Y no te preocupes,

necesitas saberlo a un nivel muy alto para el examen.

Si tiene alguna pregunta, por favor hágamelo saber. De lo contrario,

Siéntete libre de pasar a la siguiente lección. Gracias.