[ENGLISH]

Hello cloud guru's and welcome to this lesson,

where we'll configure Lambda destinations

and dead-letter queues.

We'll begin by creating a new function

and we'll use the latest Python runtime.

Next, we'll create a standard SNS topic

and subscribe to the topic using our email address.

Next, we'll create a destination for successful events.

We'll invoke the function asynchronously,

and I'll show you how to do that.

And then check if an email was received,

and the email will include details

of our invocation record, including the function payload.

Next, we're gonna remove the destination

because we're gonna be adding a dead-letter queue instead.

We'll update our function,

so that it throws an exception,

then invoke the function again

and check if an email was received.

And you'll see the difference

between the kind of emails you get

from the dead-letter queue and those you get

from the destination.

So if you are ready to have a play

with destinations and dead-letter queues,

please join me in the AWS console.

In the console, first of all, search for Lambda.

Create Function, call it "my function."

And the runtime is just gonna be the latest

Python runtime that's available.

Accept the rest of the defaults and Create Function.

After your function has successfully created,

search for SNS and open that in a new tab.

And we're gonna create a standard SNS topic.

We'll call it "my topic," select Next,

make sure that Standard is selected.

Scroll down to the bottom and Create Topic.

Next, we'll create a subscription.

So select Create Subscription,

Protocol is gonna be email

and then Endpoint is going to be the email address

that will receive notifications from SNS.

And I want you to add your own email address in here.

And once you've done that, select Create Subscription.

Now it's still saying here pending confirmation,

and that's because we need to confirm our subscription.

And to do that, you will need to click the link

in the email that you've received from SNS.

So here I am in my email and I've got an AWS notification

asking me to confirm my subscription.

And if you can't find it in your inbox,

you may find that these messages

are going to your spam folder.

And that's what's happened for me.

So it's just letting me know that I've chosen

to subscribe to the topic and I need

to confirm by clicking this link.

So select, Confirm Subscription

and you should get this message

from SNS saying that you successfully subscribed.

And once you've successfully subscribed

your subscription status will show as confirmed.

So next, let's go back to our function.

So select the Lambda tab and select Configuration.

On the left hand side you'll find Destinations

and select Add Destination.

The type of invocation

is gonna be an Asynchronous Invocation,

so we'll invoke our function asynchronously,

the condition for using the destination.

This time I'm gonna add a destination for successful events.

Destination Type can be an SQSQ, a topic, a Lambda function,

or even EventBridge,

but we're gonna select an SNS topic,

and then down here, select your topic.

It's giving us this message

that the function execution role doesn't have permission

to send results to our SNS topic,

but it's automatically gonna add

the permission that we need.

So then scroll down to the bottom and hit Save.

After we've done that, we are ready to invoke the function.

Now this works with an Asynchronous Invocation

and we can invoke our function asynchronously

by using the Cloud Shell.

So open up the Cloud Shell,

and you can open it up from the bottom here.

And I've got a command

that you will find in the resources for this lesson.

And here is our command.

AWs Lambda invoke, the name of our function,

and the invocation type is asynchronous

and it uses the parameter Event

to denote that the function will be invoked asynchronously.

So we can just copy that command,

come back to your Lambda console,

and paste in the command, and hit ENTER.

And if you get a status code of 202,

then it has successfully invoked.

So now that we've invoked our function asynchronously,

we should have received an email with our invocation record.

So let's check our email.

And there we go.

I've got a message from AWS notifications,

and here is my invocation record.

So it's got all the information about the invocation,

the timestamp, the request ID, the ARN of the function,

and the request payload, including the status code

and the body message from Lambda.

So you get quite a lot of information

from Lambda destinations.

So now, in the next step,

we're gonna remove the destination

because we're gonna add our dead-letter queue instead.

I'm gonna resize the Cloud Shell,

select my Destination and select Remove.

And to add the dead-letter queue,

on the left and side, select Asynchronous Invocation,

then select Edit, and then under dead-letter queue service,

select the dropdown and select Amazon SNS.

Then select your Topic and hit Save.

Next we're gonna update our code

with some code that is going to fail.

So select Code in your function.

I'll remove the code and you'll find a link

to my updated code in the resources for this lesson.

So I'm just gonna copy everything,

then back in my function,

paste everything into my function,

and this code is just going to raise an exception.

So, then select Deploy,

and then we're gonna invoke our function again,

so back in the Cloud Shell we'll run the same command again.

I'm gonna use up arrow

to run the command again and hit ENTER.

And if it's all worked,

you should see the same status code of 202.

So now let's check our email again

and see if we've received anything

from the dead-letter queue.

And it might just take a few minutes

for the email to arrive.

So just be patient and it should be with you soon.

And there we go, we've received another email,

this time it's from the dead-letter queue.

And just notice that it doesn't really contain

any useful information.

And that's because the purpose of this

is to let Lambda know that the invocation failed

so that it knows to try again.

It's not gonna be useful to us

to find out what went wrong or to get any information

about the request or the invocation.

So if you wanna get information

about what actually went wrong

and you want useful information,

then Lambda destinations is gonna be better for that.

This is just to let Lambda know

that there was a failed invocation.

So for the exam, remember that dead-letter queues

and destinations can be used for error handling in Lambda.

Lambda can be configured to send failed events

to the dead-letter queue for further processing,

and both SNS or SQS are supported.

However, no invocation record is gonna be included,

so you don't get any human readable information.

It's just used to let Lambda know

that there's been a failed event that needs to be retried.

Whereas with Lambda destinations,

you can handle either successful or failed invocations.

It works with SNS, SQS, Lambda and EventBridge.

And with destinations you get the invocation record.

So when we used SNS it sent us the invocation record

in the email which includes the response payload

from the function.

So that is it for this lesson.

If you have any questions, let me know.

Otherwise, I will see you in the next lesson.

Thank you.

[SPANISH]

Hola gurús de la nube y bienvenidos a esta lección,

donde configuraremos destinos Lambda

y colas de mensajes fallidos.

Comenzaremos creando una nueva función.

y usaremos el último tiempo de ejecución de Python.

A continuación, crearemos un tema SNS estándar

y suscríbase al tema utilizando nuestra dirección de correo electrónico.

A continuación, crearemos un destino para eventos exitosos.

Invocaremos la función de forma asíncrona,

y te mostraré cómo hacerlo.

Y luego verifique si se recibió un correo electrónico,

y el correo electrónico incluirá detalles

de nuestro registro de invocaciones, incluida la carga útil de la función.

A continuación, vamos a eliminar el destino.

porque vamos a agregar una cola de mensajes fallidos en su lugar.

Actualizaremos nuestra función,

para que arroje una excepción,

luego invoque la función de nuevo

y verifique si se recibió un correo electrónico.

Y verás la diferencia

entre el tipo de correos electrónicos que recibes

de la cola de mensajes fallidos y los que recibes

del destino

Así que si estás listo para jugar

con destinos y colas de mensajes fallidos,

Únase a mí en la consola de AWS.

En la consola, en primer lugar, busque Lambda.

Crear función, llámela "mi función".

Y el tiempo de ejecución será el último

Tiempo de ejecución de Python que está disponible.

Acepte el resto de los valores predeterminados y cree la función.

Después de que su función se haya creado con éxito,

busque SNS y ábralo en una nueva pestaña.

Y vamos a crear un tema SNS estándar.

Lo llamaremos "mi tema", seleccione Siguiente,

asegúrese de que esté seleccionado Estándar.

Desplácese hacia abajo hasta la parte inferior y Crear tema.

A continuación, crearemos una suscripción.

Así que seleccione Crear suscripción,

El protocolo será el correo electrónico.

y luego Endpoint será la dirección de correo electrónico

que recibirá notificaciones de SNS.

Y quiero que agregues tu propia dirección de correo electrónico aquí.

Y una vez que haya hecho eso, seleccione Crear suscripción.

Ahora sigue diciendo aquí pendiente de confirmación,

y eso es porque necesitamos confirmar nuestra suscripción.

Y para hacer eso, deberá hacer clic en el enlace

en el correo electrónico que ha recibido de SNS.

Así que aquí estoy en mi correo electrónico y tengo una notificación de AWS

pidiéndome que confirme mi suscripción.

Y si no lo encuentras en tu bandeja de entrada,

usted puede encontrar que estos mensajes

van a tu carpeta de correo no deseado.

Y eso es lo que me ha pasado.

Así que solo me hace saber que he elegido

para suscribirme al tema y necesito

para confirmar haciendo clic en este enlace.

Así que seleccione, Confirmar suscripción

y deberías recibir este mensaje

de SNS diciendo que se suscribió con éxito.

Y una vez que te hayas suscrito con éxito

el estado de su suscripción se mostrará como confirmado.

A continuación, volvamos a nuestra función.

Así que seleccione la pestaña Lambda y seleccione Configuración.

En el lado izquierdo encontrará Destinos

y seleccione Agregar destino.

El tipo de invocación

va a ser una invocación asíncrona,

entonces invocaremos nuestra función de forma asíncrona,

la condición para usar el destino.

Esta vez voy a agregar un destino para eventos exitosos.

El tipo de destino puede ser un SQSQ, un tema, una función Lambda,

o incluso EventBridge,

pero vamos a seleccionar un tema de SNS,

y luego aquí abajo, seleccione su tema.

Nos está dando este mensaje.

que el rol de ejecución de la función no tiene permiso

para enviar resultados a nuestro tema SNS,

pero se agregará automáticamente

el permiso que necesitamos.

Entonces, desplácese hacia abajo hasta la parte inferior y presione Guardar.

Después de haber hecho eso, estamos listos para invocar la función.

Ahora esto funciona con una invocación asíncrona

y podemos invocar nuestra función de forma asíncrona

mediante el uso de Cloud Shell.

Así que abre Cloud Shell,

y puedes abrirlo desde abajo aquí.

Y tengo un comando

que encontrará en los recursos para esta lección.

Y aquí está nuestro comando.

AWs Lambda invoque, el nombre de nuestra función,

y el tipo de invocación es asíncrono

y usa el parámetro Evento

para indicar que la función se invocará de forma asíncrona.

Así que podemos simplemente copiar ese comando,

regrese a su consola Lambda,

y pegue el comando, y presione ENTER.

Y si obtiene un código de estado de 202,

entonces se ha invocado con éxito.

Ahora que hemos invocado nuestra función de forma asíncrona,

deberíamos haber recibido un correo electrónico con nuestro registro de invocación.

Así que revisemos nuestro correo electrónico.

Y ahí vamos.

Tengo un mensaje de notificaciones de AWS,

y aquí está mi registro de invocación.

Así que tiene toda la información sobre la invocación,

la marca de tiempo, el ID de la solicitud, el ARN de la función,

y la carga útil de la solicitud, incluido el código de estado

y el mensaje del cuerpo de Lambda.

Entonces obtienes mucha información.

desde destinos Lambda.

Así que ahora, en el siguiente paso,

vamos a eliminar el destino

porque vamos a agregar nuestra cola de mensajes fallidos en su lugar.

Voy a cambiar el tamaño de Cloud Shell,

seleccione mi Destino y seleccione Quitar.

Y para agregar la cola de mensajes fallidos,

a la izquierda y al costado, seleccione Invocación asíncrona,

a continuación, seleccione Editar y, a continuación, en el servicio de cola de mensajes fallidos,

seleccione el menú desplegable y seleccione Amazon SNS.

Luego seleccione su Tema y presione Guardar.

A continuación, actualizaremos nuestro código.

con algún código que va a fallar.

Así que seleccione Código en su función.

Eliminaré el código y encontrarás un enlace.

a mi código actualizado en los recursos para esta lección.

Así que voy a copiar todo,

luego de vuelta en mi función,

pegar todo en mi función,

y este código solo generará una excepción.

Entonces, seleccione Implementar,

y luego vamos a invocar nuestra función de nuevo,

así que de vuelta en Cloud Shell ejecutaremos el mismo comando nuevamente.

voy a usar la flecha hacia arriba

para ejecutar el comando nuevamente y presione ENTER.

Y si todo ha funcionado,

debería ver el mismo código de estado de 202.

Así que ahora revisemos nuestro correo electrónico nuevamente

y ver si hemos recibido algo

de la cola de mensajes fallidos.

Y puede que solo tome unos minutos

para que llegue el correo.

Así que ten paciencia y debería estar contigo pronto.

Y ahí vamos, hemos recibido otro correo electrónico,

esta vez es de la cola de mensajes fallidos.

Y noten que en realidad no contiene

cualquier información útil.

Y eso es porque el propósito de esto

es hacerle saber a Lambda que la invocación falló

para que sepa intentarlo de nuevo.

no nos va a servir

para averiguar qué salió mal o para obtener cualquier información

sobre la solicitud o la invocación.

Así que si quieres obtener información

sobre lo que realmente salió mal

y quieres información útil,

entonces los destinos Lambda serán mejores para eso.

Esto es solo para que Lambda sepa

que hubo una invocación fallida.

Entonces, para el examen, recuerde que las colas de mensajes fallidos

y los destinos se pueden usar para el manejo de errores en Lambda.

Lambda se puede configurar para enviar eventos fallidos

a la cola de mensajes fallidos para su posterior procesamiento,

y se admiten tanto SNS como SQS.

Sin embargo, no se incluirá ningún registro de invocación,

por lo que no obtiene ninguna información legible por humanos.

Solo se usa para que Lambda sepa

que ha habido un evento fallido que necesita ser reintentado.

Mientras que con los destinos Lambda,

puede manejar invocaciones exitosas o fallidas.

Funciona con SNS, SQS, Lambda y EventBridge.

Y con destinos obtienes el registro de invocación.

Entonces, cuando usamos SNS, nos envió el registro de invocación

en el correo electrónico que incluye la carga de respuesta

de la función.

Eso es todo por esta lección.

Si tiene alguna pregunta, hágamelo saber.

De lo contrario, te veré en la próxima lección.

Gracias.