[ENGLISH]

Hello, Cloud Gurus, and welcome to this demo,

where we're going to build

our very first serverless website.

We'll use Lambda to run some code,

API Gateway will be used as a trigger

to invoke our Lambda function,

and we'll host our website in S3.

So the architecture is going to look like this.

Here's our Lambda function with our Python code.

It will be triggered

by API Gateway every time someone visits the website,

which is hosted in S3.

And everything that you need for this lesson

is in the resources section for this course.

So please download all of these files

before you begin.

And to do that, you can select the first file,

select Raw,

and then you can select Save Page As.

Remember to add the .html on the end of this

because it's error.html.

And it will save to your Downloads directory.

So that's just a really easy way to save the course files.

So once you've done that,

please join me in the AWS console.

So here I am in the AWS Management Console,

and the first thing we're going to do

is create our Lambda function.

So search for Lambda, and select that.

And we're going to create a function.

First of all, we'll select Author From Scratch.

We'll give our function a name.

I'm just going to call it mylambdafunction.

The runtime is going to be the latest version of Python.

Architecture is x86\_64

and then down here under permissions,

you will see that by default,

Lambda is going to create a basic Identity

and Access Management role.

And it's called an execution role.

So this is a role which gives Lambda permissions

to do things within your AWS account.

So it might need, for example,

to access an S3 bucket

or to access items in a DynamoDB table,

but by default, the basic permissions that you get

just allows Lambda to send logs into CloudWatch.

And this is going to be fine for what we're doing today.

However, you can go in

and change this default execution role.

For instance, you can create a new role

and select from different policy templates

to provide additional permissions to your Lambda function,

and we're not going to do that today.

We're just going to keep to the default,

which is to create the new role

with basic Lambda permissions.

And then scrolling down,

we can just select Create Function.

And that is our function created.

And if we scroll down,

here's our code source, and

this is where we can add our Python code to the function,

and we can write our code directly into this console,

which is really cool,

and it actually uses the Cloud9 IDE.

And Cloud9 is a company that AWS acquired

quite a few years now,

and they've incorporated it into the Lambda console.

And it's a really nice feature

because it makes it really easy

to add and edit your code in Lambda.

So the first thing I'm going to do

is just delete the sample code,

and I'm going to add my own Python code

from the resources for this lesson.

And it's this file called hellocloudgurus.py.

Here's the code.

And we can just copy and paste it from here.

So this is a very simple piece of code,

and when you run the function,

it's basically going to return whatever we have written

in this body section.

And if the function is successful,

we'll get a status code of 200.

And if you want, you can change this phrase in here

to whatever you like.

So that is our function,

and once you've copied it in,

just hit Deploy, and then we're going to scroll back up

and this is where we can add a trigger,

and if you remember, we use a trigger

to invoke our function.

So click Add Trigger.

And triggers are really important to remember

for the exam.

And there are loads and loads of different triggers

that can be used to invoke a Lambda function.

But the most common ones you're likely to see

are things like API Gateway,

you've also got Alexa, CloudWatch,

DynamoDB,

Kinesis, S3,

SNS, and SQS.

And they've also got some partner event sources

as well in here.

So you can integrate Lambda

with any of these services

and have these services triggering

or invoking your Lambda function.

So you should definitely be aware

that you can invoke a Lambda function

using all these different services.

And it's a great way to automate your application.

For instance, you can invoke your Lambda function

when a new object is saved to S3

or when a new item is added to a DynamoDB table.

Now, we're going to use API Gateway as our trigger.

So select that.

We're going to be creating a new API.

The API type is going to be HTTP.

Under security, you can set up authorization

with JWT tokens.

But we're going to leave it as open

because this is just a public website.

Under additional settings,

here's our API name,

and we can change it if we want to.

I'm just going to stick with the default one it's given us.

And we also need to enable cross-origin resource sharing.

And this is going to allow our webpage

to successfully call the API.

And if you don't do this, it's not going to work.

So then you can scroll down and hit Add.

And that is our API Gateway configured.

And here is our API Gateway endpoint.

And if we right-click and open that up in a new tab,

then this is what you should see,

so this is API Gateway calling our Lambda function.

And Lambda is responding by running our Python code

and returning this response.

So how cool is that?

And now at this point,

we are ready to join everything up with S3.

So I'm going to come back to my console

and search for S3.

I'm going to open that up in a new tab

and select Create Bucket.

We'll give our bucket a unique name.

Just add some random numbers on the end

to make sure it's a unique name.

Under object ownership,

you'll need to enable ACLs

because we'll need these later on

to make our objects publicly accessible,

and if you scroll down,

you will also need to disable

the Block Public Access settings on the bucket.

So disable those options

because this is going to be a website.

So we need to make sure that public access is allowed.

Acknowledge your settings,

and then we can go to the bottom and Create Bucket.

Now, once that's created,

select your bucket.

And come to properties

and then scroll down to the bottom

until you find static website hosting.

Select Edit.

And we're going to enable static website hosting.

Down here, we need to specify our index document

and our error document.

So just type index.html and error.html.

Scroll down.

And Save Changes.

So now we need to upload our website files,

and we've got the 2 files that we need to upload:

our index.html and error.html.

And these are both in our resources for the course.

But before we upload them,

we've got 1 change

that we need to make in index.html.

And we'll need to insert the endpoint URL

of our API Gateway into our HTML code

so that when the user visits the website,

it's going to call API Gateway

and invoke our Lambda function.

So to do that, we'll go back to our Lambda function,

and I'm going to copy our API Gateway endpoint

to my clipboard.

Then open up my index.html, and

we want to replace this with the API\_GATEWAY\_ENDPOINT\_URL.

So this is what it should look like.

I'm just going to save my file,

and let's quickly go through what this code actually does.

So first of all, we've got some JavaScript

with a function called myFunction.

And this function is going to call API Gateway.

So it makes an HTTP request

to the API Gateway URL.

And then moving down,

we've got the body section.

So we've got a heading that says "Hello Cloud Gurus!"

And then we've got a button that says "Click me."

And when you click on this button,

it's going to call our JavaScript function

to make the HTTP request to API Gateway,

which in turn will trigger our Lambda function

to send a response back.

So once you've finished modifying the file,

and once you've saved it,

we're ready to upload our index.html

and our error.html.

But I'll just quickly show you the error.html,

and it just basically displays this error message

if something goes wrong on the website.

So when you're ready to upload those 2 files,

head back to the S3 console.

Here's our S3 bucket.

So select your bucket,

and before you upload them, select Permissions,

and we want to grant public read access on those files.

And you can acknowledge that down here.

And select Upload.

So now still within our S3 bucket,

if we head to Properties,

scroll down to the bottom

until you find static website hosting,

and this is where you'll find

your static website hosting endpoint

for your serverless website.

So this is the website address provided by S3.

And if we select the website,

I'm going to open it in a new tab.

You should see something like this.

So it's taken us to our website.

And then if you click the button,

you should see the website message update

with our message from the Lambda function.

So there you go, we've built a static website in S3.

Our HTML code is calling API Gateway,

which in turn invokes our Lambda function

and displays the response from our function on the website.

So that is Lambda.

And in summary, we created a Lambda function

and added our Python code.

We used API Gateway to trigger the function

every time we hit the button on the webpage,

which was hosted in an S3 bucket,

no servers required.

And with that, we finished the end of this lesson.

So if you have any questions, please let me know.

Otherwise, I will look forward

to seeing you in the next one.

Thank you.

[SPANISH]

Hola, Cloud Gurus, y bienvenidos a esta demostración.

donde vamos a construir

nuestro primer sitio web sin servidor.

Usaremos Lambda para ejecutar código,

API Gateway se utilizará como activador

para invocar nuestra función Lambda,

y alojaremos nuestro sitio web en S3.

Así que la arquitectura se va a ver así.

Aquí está nuestra función Lambda con nuestro código Python.

se activará

por API Gateway cada vez que alguien visita el sitio web,

que está alojado en S3.

Y todo lo que necesitas para esta lección.

se encuentra en la sección de recursos de este curso.

Así que por favor descargue todos estos archivos

antes de que empieces.

Y para hacer eso, puede seleccionar el primer archivo,

seleccione crudo,

y luego puede seleccionar Guardar página como.

Recuerde agregar el .html al final de este

porque es error.html.

Y se guardará en su directorio de Descargas.

Esa es una manera muy fácil de guardar los archivos del curso.

Así que una vez que hayas hecho eso,

Únase a mí en la consola de AWS.

Así que aquí estoy en la consola de administración de AWS,

y lo primero que vamos a hacer

es crear nuestra función Lambda.

Así que busque Lambda y selecciónelo.

Y vamos a crear una función.

En primer lugar, seleccionaremos Autor desde cero.

Le daremos un nombre a nuestra función.

Voy a llamarlo mylambdafunction.

El tiempo de ejecución será la última versión de Python.

La arquitectura es x86\_64

y luego aquí abajo bajo permisos,

verás que por defecto,

Lambda va a crear una identidad básica

y función de gestión de acceso.

Y se llama rol de ejecución.

Este es un rol que otorga permisos de Lambda.

para hacer cosas dentro de su cuenta de AWS.

Entonces podría necesitar, por ejemplo,

para acceder a un depósito S3

o para acceder a elementos en una tabla de DynamoDB,

pero por defecto, los permisos básicos que obtienes

solo permite que Lambda envíe registros a CloudWatch.

Y esto va a estar bien para lo que estamos haciendo hoy.

Sin embargo, puedes entrar

y cambie este rol de ejecución predeterminado.

Por ejemplo, puede crear un nuevo rol

y seleccione entre diferentes plantillas de políticas

para proporcionar permisos adicionales a su función Lambda,

y no vamos a hacer eso hoy.

Simplemente vamos a mantener el valor predeterminado,

que es crear el nuevo rol

con permisos básicos de Lambda.

Y luego, desplazándose hacia abajo,

simplemente podemos seleccionar Crear función.

Y esa es nuestra función creada.

Y si nos desplazamos hacia abajo,

aquí está nuestro código fuente, y

aquí es donde podemos agregar nuestro código Python a la función,

y podemos escribir nuestro código directamente en esta consola,

que es realmente genial,

y en realidad usa el IDE de Cloud9.

Y Cloud9 es una empresa que AWS adquirió

bastantes años ya,

y lo han incorporado a la consola Lambda.

Y es una característica muy agradable

porque lo hace muy fácil

para agregar y editar su código en Lambda.

Así que lo primero que voy a hacer

es simplemente eliminar el código de muestra,

y voy a agregar mi propio código de Python

de los recursos para esta lección.

Y es este archivo llamado hellocloudgurus.py.

Aquí está el código.

Y podemos simplemente copiarlo y pegarlo desde aquí.

Así que este es un código muy simple,

y cuando ejecutas la función,

básicamente va a devolver lo que hayamos escrito

en esta sección del cuerpo.

Y si la función tiene éxito,

obtendremos un código de estado de 200.

Y si quieres, puedes cambiar esta frase aquí.

a lo que quieras

Así que esa es nuestra función,

y una vez que lo hayas copiado,

solo presione Implementar, y luego nos desplazaremos hacia arriba

y aquí es donde podemos agregar un disparador,

y si recuerdas, usamos un gatillo

para invocar nuestra función.

Así que haga clic en Agregar activador.

Y los factores desencadenantes son muy importantes para recordar

para el examen.

Y hay montones y montones de disparadores diferentes

que se puede utilizar para invocar una función Lambda.

Pero los más comunes que es probable que veas

son cosas como API Gateway,

también tienes Alexa, CloudWatch,

dinamodb,

Cinesis, S3,

SNS y SQS.

Y también tienen algunas fuentes de eventos asociadas

también aquí.

Para que pueda integrar Lambda

con cualquiera de estos servicios

y hacer que estos servicios se activen

o invocando su función Lambda.

Así que definitivamente deberías estar consciente

que puede invocar una función Lambda

utilizando todos estos servicios diferentes.

Y es una gran manera de automatizar su aplicación.

Por ejemplo, puede invocar su función Lambda

cuando se guarda un nuevo objeto en S3

o cuando se agrega un nuevo elemento a una tabla de DynamoDB.

Ahora, vamos a utilizar API Gateway como disparador.

Así que selecciona eso.

Vamos a crear una nueva API.

El tipo de API será HTTP.

Bajo seguridad, puede configurar la autorización

con fichas JWT.

Pero lo vamos a dejar como abierto.

porque esto es sólo un sitio web público.

En configuraciones adicionales,

aquí está nuestro nombre de API,

y podemos cambiarlo si queremos.

Voy a quedarme con el predeterminado que nos ha dado.

Y también necesitamos habilitar el intercambio de recursos de origen cruzado.

Y esto va a permitir que nuestra página web

para llamar con éxito a la API.

Y si no haces esto, no va a funcionar.

Entonces puedes desplazarte hacia abajo y presionar Agregar.

Y esa es nuestra API Gateway configurada.

Y aquí está nuestro punto final de API Gateway.

Y si hacemos clic derecho y lo abrimos en una nueva pestaña,

entonces esto es lo que deberías ver,

así que esto es API Gateway llamando a nuestra función Lambda.

Y Lambda está respondiendo ejecutando nuestro código Python

y devolver esta respuesta.

Entonces, ¿qué tan genial es eso?

Y ahora en este punto,

estamos listos para unir todo con S3.

Así que voy a volver a mi consola.

y busque S3.

Voy a abrir eso en una nueva pestaña.

y seleccione Crear depósito.

Le daremos a nuestro cubo un nombre único.

Solo agregue algunos números aleatorios al final

para asegurarse de que es un nombre único.

Bajo la propiedad del objeto,

deberá habilitar las ACL

porque los necesitaremos más adelante

para hacer que nuestros objetos sean públicamente accesibles,

y si te desplazas hacia abajo,

también tendrás que deshabilitar

la configuración de Bloquear acceso público en el depósito.

Así que deshabilite esas opciones

porque esto va a ser un sitio web.

Por lo tanto, debemos asegurarnos de que se permita el acceso público.

Reconocer su configuración,

y luego podemos ir al final y Crear cubo.

Ahora, una vez que se crea,

seleccione su balde.

Y ven a propiedades

y luego desplácese hacia abajo hasta el final

hasta que encuentre alojamiento de sitio web estático.

Seleccione Editar.

Y vamos a habilitar el alojamiento de sitios web estáticos.

Aquí abajo, necesitamos especificar nuestro documento de índice

y nuestro documento de error.

Así que simplemente escriba index.html y error.html.

Desplácese hacia abajo.

Y Guardar Cambios.

Así que ahora tenemos que cargar los archivos de nuestro sitio web,

y tenemos los 2 archivos que necesitamos subir:

nuestro index.html y error.html.

Y ambos están en nuestros recursos para el curso.

Pero antes de subirlos,

tenemos 1 cambio

que necesitamos hacer en index.html.

Y necesitaremos insertar la URL del punto final

de nuestra API Gateway en nuestro código HTML

para que cuando el usuario visite el sitio web,

va a llamar API Gateway

e invoque nuestra función Lambda.

Entonces, para hacer eso, volveremos a nuestra función Lambda,

y voy a copiar nuestro punto final de API Gateway

a mi portapapeles.

Luego abre mi index.html, y

queremos reemplazar esto con API\_GATEWAY\_ENDPOINT\_URL.

Así es como debería verse.

Solo voy a guardar mi archivo,

y repasemos rápidamente lo que realmente hace este código.

En primer lugar, tenemos algo de JavaScript

con una función llamada myFunction.

Y esta función llamará a API Gateway.

Entonces hace una solicitud HTTP

a la URL de puerta de enlace API.

Y luego bajando,

tenemos la sección del cuerpo.

Así que tenemos un encabezado que dice "¡Hola, gurús de la nube!"

Y luego tenemos un botón que dice "Haz clic en mí".

Y al hacer clic en este botón,

va a llamar a nuestra función de JavaScript

para realizar la solicitud HTTP a API Gateway,

que a su vez activará nuestra función Lambda

para enviar una respuesta de vuelta.

Entonces, una vez que haya terminado de modificar el archivo,

y una vez que lo hayas guardado,

estamos listos para cargar nuestro index.html

y nuestro error.html.

Pero te mostraré rápidamente el error.html,

y básicamente muestra este mensaje de error

si algo sale mal en el sitio web.

Entonces, cuando esté listo para cargar esos 2 archivos,

regresa a la consola S3.

Aquí está nuestro cubo S3.

Así que seleccione su cubo,

y antes de subirlos, seleccione Permisos,

y queremos otorgar acceso público de lectura a esos archivos.

Y puedes reconocer eso aquí abajo.

Y seleccione Cargar.

Así que ahora todavía dentro de nuestro cubo S3,

si nos dirigimos a Propiedades,

desplácese hacia abajo hasta la parte inferior

hasta que encuentre alojamiento de sitio web estático,

y aquí es donde encontrarás

su punto final de alojamiento de sitio web estático

para su sitio web sin servidor.

Esta es la dirección del sitio web proporcionada por S3.

Y si seleccionamos el sitio web,

Voy a abrirlo en una nueva pestaña.

Debería ver algo como esto.

Así que nos ha llevado a nuestro sitio web.

Y luego, si hace clic en el botón,

debería ver la actualización del mensaje del sitio web

con nuestro mensaje de la función Lambda.

Ahí lo tienes, hemos creado un sitio web estático en S3.

Nuestro código HTML está llamando a API Gateway,

que a su vez invoca nuestra función Lambda

y muestra la respuesta de nuestra función en el sitio web.

Así que eso es Lambda.

Y en resumen, creamos una función Lambda

y agregamos nuestro código Python.

Usamos API Gateway para activar la función.

cada vez que presionamos el botón en la página web,

que estaba alojado en un depósito S3,

no se requieren servidores.

Y con eso, terminamos el final de esta lección.

Entonces, si tiene alguna pregunta, hágamelo saber.

De lo contrario, esperaré

a verte en la próxima.

Gracias.