[ENGLISH]

Hello, Cloud Gurus, and welcome to this lesson,

which is going to be a demo of "Step Functions."

And we'll begin with a quick overview of Step Functions

so that you can understand what they are

and the reason for using them.

We'll review the different types of workflow

that are available with Step Functions,

so sequential workflows, parallel workflows,

and branching workflows.

We'll then have a quick demo using a sample project

and finish off with my exam tips.

Now, Step Functions provide a visual interface

for serverless applications,

allowing you to build and run serverless applications

as a series of steps.

Each step in your application

executes in the order defined by your business logic,

and the output of 1 step

can be the input into the next step.

So Step Functions provide orchestration,

and they manage the logic of your application.

And if you think of an application

consisting of many different Lambda functions,

it makes sense that you will need

to somehow orchestrate everything,

including sequencing, error handling, and retry logic,

so that everything executes

in the correct order as you expect.

And Step Functions also log the state of each step.

So when things go wrong,

this is going to help you to diagnose

and debug problems really quickly

because you'll be able to identify

which step in the process failed.

Now, I have a few examples of the kind of workflows

that we can manage using Step Functions.

And the first one is a sequential workflow.

And this is where the steps happen one after the other.

And in this workflow, we're beginning by decoding some data.

We then generate some statistics,

process the data by removing special characters,

we tokenize the outcome, and provide account,

and then that's the end of the workflow.

So all of these steps happen one after the other,

and the output of 1 step is the input for the next step.

And when we consider a workflow like this,

each of these steps is going to be fulfilled

by a different Lambda function.

Now, Step Functions consist of state machines

and tasks,

and a state machine is simply the workflow itself,

and a task is a single step or a single unit of work

that makes up the workflow.

Now, in my next example,

we have an example of a parallel workflow,

and we start off by processing a photo,

and then the workflow splits into 3 parallel tasks.

So we're extracting metadata, resizing the image,

and then performing some facial recognition.

And all of these activities are happening in parallel

by separate Lambda functions.

And then finally, the 3 parallel tasks converge

and with saving some data to a database.

So that is the parallel workflow.

And once again, the output of each step

is feeding into the next step.

And remember, each of these steps is handled

by a different Lambda function,

which needs to be coordinated.

And then the whole thing is coordinated by Step Functions.

And then in my last example, we have a branching workflow,

and it begins as a sequential workflow.

So it's triggering code build to complete a build.

Next, we get our test results.

And then if all of the tests pass,

we have 1 course of action,

so we notify success.

And if any of the tests fail,

then we have a different course of action.

So it's going to notify failure.

So in this workflow, we're actually making a decision

as to which path we're going to take.

And once again, all of these different steps or tasks

are going to be managed by different Lambda functions,

and then the whole thing is coordinated by Step Functions.

Now, for the exam,

you only need to know what Step Functions are

and why we would use them.

But I want to give you a quick demo

just because it makes it a lot easier to remember

once you've seen it in action.

So if you'd like to join me in the AWS console,

we will get our hands dirty with Step Functions.

So here I am in the console,

and in the search box,

I want you to search for Step Functions.

And then open up the menu on the left

and select State Machines.

We're going to create a state machine,

and we're going to run a sample project.

Now, under the sample project,

I want you to scroll down until you find this one,

Job Polar, and select that.

And this is going to create a sample application using

Lambda. Scroll down to the bottom,

and this is where we can view the definition

of our state machine or workflow.

Now, state machines are defined

using Amazon States Language.

And don't worry,

you don't need to learn this language for the exam.

And then, on the right-hand side,

there's also this cool diagram as well.

And this is where you get your visual representation

of your state machine.

Now, this is a fairly simple workflow,

and it starts off by submitting a job,

then it waits for a number of seconds,

then it gets the job status.

If the job is not complete,

it's going to go back around in this loop.

So it will wait again,

and then it will try to get the job status again,

and it's going to do that for a maximum of 3 attempts.

And if after 3 attempts, the job is still not complete,

it's going to tell us that the job failed.

So it's a pretty simple workflow.

And now we can just scroll down,

and select Next,

and Deploy Resources.

Now, it's going to build everything for us

using CloudFormation.

And just be aware

that it normally takes around 10 minutes to complete.

So at this point, you can just take a break,

come back in 10 minutes, and we should be good to go.

So now, it's been a few minutes.

I think it was a little less than 10 minutes, actually,

and everything has complete.

And if it's all been successful,

then this is what you should see.

And we are ready to execute our state machine.

But just in case you wanted to run this job twice,

then you can just copy these input values here.

Just copy to your clipboard,

and then you'll be able to run everything a second time

if you wish.

So now, I'm just going to select Open In A New Browser Tab

and select Start Execution.

And then if we scroll down,

we can see the progress of our job,

and we can see that we've got a task in progress here.

And the task in progress is shown as blue.

And we've also got a task that has completed as well.

And if a task has successfully completed,

it's going to be shown as green,

and then any task that fails are going to be shown as red.

And if you select one of your tasks,

you can see the details over here.

So you can see the input and output.

And then once everything is complete,

it should all show as green.

And then if we scroll down here,

we can see the execution event history.

So we can see all of our Lambda functions being scheduled,

and started, and succeeded.

And then if we scroll back up,

we can see that everything has completed successfully.

So that is Step Functions.

And for my exam tips, just remember

that Step Functions are a great way to visualize

your serverless applications.

They automatically trigger

and track each step of your workflow or your state machine.

And the output of 1 step

is often the input into the next.

Step Functions log the state of each step,

so if something goes wrong,

then you can track what went wrong and where.

And, if you remember in the console,

we could see all the events in the execution event history,

and it also gave us a visual workflow

showing us the status of each step.

So that's it for Step Functions.

If you have any questions, please let me know.

Otherwise, I will see you in the next lesson.

Thank you.

[SPANISH]

Hola, Cloud Gurus, y bienvenidos a esta lección.

que va a ser una demostración de "Step Functions".

Y comenzaremos con una descripción general rápida de las funciones de paso

para que entiendas lo que son

y la razón para usarlos.

Revisaremos los diferentes tipos de flujo de trabajo.

que están disponibles con Step Functions,

flujos de trabajo secuenciales, flujos de trabajo paralelos,

y ramificación de flujos de trabajo.

Luego tendremos una demostración rápida usando un proyecto de muestra.

y terminar con mis consejos de examen.

Ahora, Step Functions proporciona una interfaz visual

para aplicaciones sin servidor,

permitiéndole crear y ejecutar aplicaciones sin servidor

como una serie de pasos.

Cada paso en su aplicación

se ejecuta en el orden definido por su lógica de negocios,

y la salida de 1 paso

puede ser la entrada en el siguiente paso.

Entonces, Step Functions proporciona orquestación,

y gestionan la lógica de su aplicación.

Y si piensas en una aplicación

que consta de muchas funciones Lambda diferentes,

tiene sentido que necesitarás

de alguna manera orquestar todo,

incluida la secuenciación, el manejo de errores y la lógica de reintento,

para que todo se ejecute

en el orden correcto como esperas.

Y Step Functions también registra el estado de cada paso.

Así que cuando las cosas van mal,

esto te va a ayudar a diagnosticar

y depurar problemas muy rápido

porque podrás identificar

qué paso del proceso falló.

Ahora, tengo algunos ejemplos del tipo de flujos de trabajo

que podemos gestionar usando Step Functions.

Y el primero es un flujo de trabajo secuencial.

Y aquí es donde los pasos suceden uno tras otro.

Y en este flujo de trabajo, comenzamos por decodificar algunos datos.

Luego generamos algunas estadísticas,

procesar los datos eliminando caracteres especiales,

tokenizamos el resultado y damos cuenta,

y entonces ese es el final del flujo de trabajo.

Así que todos estos pasos suceden uno tras otro,

y la salida de 1 paso es la entrada para el siguiente paso.

Y cuando consideramos un flujo de trabajo como este,

cada uno de estos pasos se va a cumplir

por una función Lambda diferente.

Ahora, las funciones de paso consisten en máquinas de estado

y tareas,

y una máquina de estado es simplemente el propio flujo de trabajo,

y una tarea es un solo paso o una sola unidad de trabajo

que conforma el flujo de trabajo.

Ahora, en mi siguiente ejemplo,

tenemos un ejemplo de un flujo de trabajo paralelo,

y comenzamos procesando una foto,

y luego el flujo de trabajo se divide en 3 tareas paralelas.

Estamos extrayendo metadatos, cambiando el tamaño de la imagen,

y luego realizar algún reconocimiento facial.

Y todas estas actividades están sucediendo en paralelo.

por funciones Lambda separadas.

Y finalmente, las 3 tareas paralelas convergen

y con guardar algunos datos en una base de datos.

Ese es el flujo de trabajo paralelo.

Y una vez más, el resultado de cada paso

está alimentando el siguiente paso.

Y recuerda, cada uno de estos pasos se maneja

por una función Lambda diferente,

que hay que coordinar.

Y luego todo está coordinado por Step Functions.

Y luego, en mi último ejemplo, tenemos un flujo de trabajo ramificado,

y comienza como un flujo de trabajo secuencial.

Por lo tanto, está activando la compilación de código para completar una compilación.

A continuación, obtenemos los resultados de nuestra prueba.

Y luego, si todas las pruebas pasan,

tenemos 1 curso de acción,

así notificamos éxito.

Y si alguna de las pruebas falla,

entonces tenemos un curso de acción diferente.

Así que va a notificar el fallo.

Entonces, en este flujo de trabajo, en realidad estamos tomando una decisión

en cuanto a qué camino vamos a tomar.

Y una vez más, todos estos diferentes pasos o tareas

van a ser administrados por diferentes funciones de Lambda,

y luego todo está coordinado por Step Functions.

Ahora, para el examen,

solo necesitas saber qué son las funciones de paso

y por qué los usaríamos.

Pero quiero darte una demostración rápida

solo porque hace que sea mucho más fácil de recordar

una vez que lo hayas visto en acción.

Entonces, si desea unirse a mí en la consola de AWS,

nos ensuciaremos las manos con Step Functions.

Así que aquí estoy en la consola,

y en el cuadro de búsqueda,

Quiero que busque funciones de paso.

Y luego abre el menú de la izquierda.

y seleccione Máquinas de estado.

Vamos a crear una máquina de estado,

y vamos a ejecutar un proyecto de muestra.

Ahora, bajo el proyecto de ejemplo,

Quiero que te desplaces hacia abajo hasta que encuentres este,

Trabajo Polar, y seleccione eso.

Y esto va a crear una aplicación de muestra usando

lambda. Desplácese hacia abajo hasta el final,

y aquí es donde podemos ver la definición

de nuestra máquina de estado o flujo de trabajo.

Ahora, las máquinas de estado están definidas

utilizando el lenguaje de estados de Amazon.

Y no te preocupes,

no es necesario aprender este idioma para el examen.

Y luego, en el lado derecho,

también está este genial diagrama.

Y aquí es donde obtienes tu representación visual.

de su máquina de estado.

Ahora, este es un flujo de trabajo bastante simple,

y comienza enviando un trabajo,

luego espera unos segundos,

luego obtiene el estado del trabajo.

Si el trabajo no está completo,

va a dar la vuelta en este bucle.

Así que esperará de nuevo,

y luego intentará obtener el estado del trabajo nuevamente,

y va a hacer eso por un máximo de 3 intentos.

Y si después de 3 intentos, el trabajo aún no está completo,

nos va a decir que el trabajo falló.

Así que es un flujo de trabajo bastante simple.

Y ahora podemos desplazarnos hacia abajo,

y seleccione Siguiente,

y desplegar recursos.

Ahora, va a construir todo para nosotros.

utilizando CloudFormation.

Y solo sé consciente

que normalmente tarda unos 10 minutos en completarse.

Entonces, en este punto, puedes simplemente tomar un descanso,

Vuelve en 10 minutos y deberíamos estar listos para irnos.

Así que ahora, han pasado unos minutos.

Creo que fue un poco menos de 10 minutos, en realidad,

y todo lo tiene completo.

Y si todo ha tenido éxito,

entonces esto es lo que deberías ver.

Y estamos listos para ejecutar nuestra máquina de estado.

Pero en caso de que quisiera ejecutar este trabajo dos veces,

entonces puede simplemente copiar estos valores de entrada aquí.

Simplemente copie en su portapapeles,

y luego podrás ejecutar todo por segunda vez

si lo desea.

Así que ahora, solo voy a seleccionar Abrir en una nueva pestaña del navegador

y seleccione Iniciar ejecución.

Y luego, si nos desplazamos hacia abajo,

podemos ver el progreso de nuestro trabajo,

y podemos ver que tenemos una tarea en progreso aquí.

Y la tarea en curso se muestra en azul.

Y también tenemos una tarea que también se ha completado.

Y si una tarea se ha completado con éxito,

se va a mostrar como verde,

y luego cualquier tarea que falle se mostrará en rojo.

Y si selecciona una de sus tareas,

Puedes ver los detalles aquí.

Para que pueda ver la entrada y la salida.

Y luego, una vez que todo esté completo,

todo debería mostrarse en verde.

Y luego, si nos desplazamos hacia abajo aquí,

podemos ver el historial de eventos de ejecución.

Entonces podemos ver todas nuestras funciones de Lambda siendo programadas,

y comenzó, y tuvo éxito.

Y luego, si retrocedemos hacia arriba,

podemos ver que todo se ha completado con éxito.

Así que eso es funciones de paso.

Y para mis consejos de examen, solo recuerda

que las funciones de pasos son una gran manera de visualizar

sus aplicaciones sin servidor.

Se disparan automáticamente

y realice un seguimiento de cada paso de su flujo de trabajo o de su máquina de estado.

Y la salida de 1 paso

es a menudo la entrada en el siguiente.

Las funciones de paso registran el estado de cada paso,

así que si algo sale mal,

entonces puede rastrear qué salió mal y dónde.

Y, si recuerdas en la consola,

pudimos ver todos los eventos en el historial de eventos de ejecución,

y también nos dio un flujo de trabajo visual

mostrándonos el estado de cada paso.

Así que eso es todo para funciones de paso.

Si tiene alguna pregunta, por favor hágamelo saber.

De lo contrario, te veré en la próxima lección.

Gracias.