[ENGLISH]

Hello, Cloud Gurus, and welcome

to the Serverless Summary - Part 2.

So let's jump straight in, beginning with Lambda and VPCs.

So you can enable Lambda to access resources

in a private VPC, and to do that, you need to provide

the VPC config information to your Lambda function,

so that will be the private subnet ID

and the security group ID.

And then Lambda will use the VPC information

to configure an elastic network interface

using an IP address from the private subnet CIDR range.

So let's say our private subnet CIDR range is 10.0.0.0/24.

Then Lambda will configure the elastic network interface

using an IP address from that range,

and it could be something like 10.0.0.5.

And the security group then allows your function

to access resources within your private VPC.

Onto step functions, and just remember

that step functions are used to visualize

and orchestrate your serverless applications,

and they automatically trigger and track each step

of the state machine or workflow,

and the output of one step is often the input to the next.

And step functions will log the state of each step

so if something goes wrong, then you can track

what went wrong and where.

Now for the exam, it's important to understand

the differences between the types of workflow

available with step functions.

So we've got standard workflows,

designed for long-running processes

that take up to a year to complete.

They run at most once and are non-idempotent.

So if you run an identical task, it would always

cause a change in state, so this is workflows

that you only want to run one time.

Then there's express workflows designed

for short-lived processes, up to 5 minutes.

They run at least once, so they can run multiple times,

and they are idempotent, and this is for workflows

that allow you turn an identical task multiple times,

without causing a change in state each time,

and there are 2 different types of express workflow.

We've got synchronous and asynchronous.

So a synchronous express workflow is one

that must complete before the next step begins.

So think about a customer placing an order

on your website and they need to create an account

on your website before they can complete their order.

And then with asynchronous express workflows,

other tasks are not dependent on completion

of the workflow, so think about a messaging system

where we're sending out messages and then move on

to the next task without waiting

for any acknowledgement.

Moving on to X-Ray, and X-Ray helps developers

to analyze and debug distributed applications,

and it provides you with a service map

which is a visual representation of all the components

that make up your application.

And if you would like to run X-Ray on your EC2 instance,

then the X-Ray agent or daemon must be installed

and we use the X-Ray SDK to instrument

your application or configure it to send traces into X-Ray.

And just to remind you, this is what the X-Ray

Service Map looks like, and it provides

an end-to-end view of requests as they travel

through your application.

So it will provide information about latency,

HTTP status codes, and error messages as well,

and this kind of information can be used

to troubleshoot connectivity and performance issues.

And X-Ray is integrated with lots of different AWS services

like DynamoDB, Lambda, and API Gateway.

And you can also instrument your own applications

or configure them to send data into X-Ray,

for example data about incoming HTTP requests

into your application.

And applications could be running on EC2,

Elastic Beanstalk, on-premises systems,

and even in Elastic Container Service.

And if you are running X-Ray in Elastic Container Service,

then it's best practice to run the X-Ray daemon

in its own Docker image,

running alongside your application,

so you don't install it on the same Docker container

running your application.

And we use X-Ray annotations

to add user-defined key-value pairs

to your X-Ray data, and that will allow you to filter,

index, and search the data, for example,

searching for a specific game

using the game name or game ID.

Onto API Gateway caching, and API Gateway caching

of course improves the performance of your APIs

by caching the output of API calls

to avoid calling your backend application every time.

Responses are cached for the TTL period,

and the default TTL or time to live is 300 seconds,

or 5 minutes.

And it reduces the number of API calls

made to your application because API Gateway

returns the cached response instead of making

a new request to your application.

And this improves performance and reduces latency

for your end users.

And then another way that API Gateway

manages performance of your applications

is by using throttling,

and API Gateway has a default limit of 10,000

requests per second and 5,000 concurrent requests,

and these are per-region limits.

If you exceed the limit, then API Gateway

returns an error, and it's a 429 error,

and you might also see the message, "Too many requests."

And API Gateway uses throttling to prevent your API

from being overwhelmed by too many requests.

And just remember, these are default limits,

so you can make a request to AWS Support

to increase your limits.

And then finally, we have Advanced API Gateway.

So just remember that you can import

your own APIs using external definition files

and the supported format is OpenAPI,

which is formerly known as Swagger.

And when dealing with legacy applications

which use the SOAP protocol, you can configure

API Gateway as a web service passthrough,

or you can use API Gateway to convert

the XML response that you receive from the application

to JSON format.

So that's it for this lecture.

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

al Resumen sin servidor - Parte 2.

Así que saltemos directamente, comenzando con Lambda y VPC.

Para que pueda habilitar Lambda para acceder a los recursos

en una VPC privada, y para hacerlo, debe proporcionar

la información de configuración de VPC a su función Lambda,

entonces esa será la ID de subred privada

y el ID del grupo de seguridad.

Y luego Lambda usará la información de VPC

para configurar una interfaz de red elástica

usando una dirección IP del rango CIDR de la subred privada.

Así que digamos que nuestro rango de CIDR de subred privada es 10.0.0.0/24.

Luego, Lambda configurará la interfaz de red elástica.

usando una dirección IP de ese rango,

y podría ser algo así como 10.0.0.5.

Y el grupo de seguridad luego permite su función

para acceder a los recursos dentro de su VPC privada.

Sobre las funciones de paso, y solo recuerda

que las funciones de paso se utilizan para visualizar

y organice sus aplicaciones sin servidor,

y activan y rastrean automáticamente cada paso

de la máquina de estado o flujo de trabajo,

y la salida de un paso es a menudo la entrada del siguiente.

Y las funciones de paso registrarán el estado de cada paso

así que si algo sale mal, entonces puedes rastrear

qué salió mal y dónde.

Ahora para el examen, es importante entender

las diferencias entre los tipos de flujo de trabajo

disponible con funciones de paso.

Así que tenemos flujos de trabajo estándar,

diseñado para procesos de larga duración

que tardan hasta un año en completarse.

Se ejecutan como máximo una vez y no son idempotentes.

Entonces, si ejecuta una tarea idéntica, siempre

provocar un cambio de estado, por lo que se trata de flujos de trabajo

que sólo desea ejecutar una vez.

Luego están los flujos de trabajo exprés diseñados

para procesos de corta duración, hasta 5 minutos.

Se ejecutan al menos una vez, por lo que pueden ejecutarse varias veces,

y son idempotentes, y esto es para flujos de trabajo

que le permiten realizar una tarea idéntica varias veces,

sin causar un cambio de estado cada vez,

y hay 2 tipos diferentes de flujo de trabajo express.

Tenemos sincrónicos y asincrónicos.

Entonces, un flujo de trabajo express sincrónico es uno

que debe completarse antes de que comience el siguiente paso.

Así que piensa en un cliente que hace un pedido

en su sitio web y necesitan crear una cuenta

en su sitio web antes de que puedan completar su pedido.

Y luego, con flujos de trabajo rápidos asíncronos,

otras tareas no dependen de la finalización

del flujo de trabajo, así que piense en un sistema de mensajería

donde estamos enviando mensajes y luego seguir adelante

a la siguiente tarea sin esperar

para cualquier reconocimiento.

Pasando a X-Ray, y X-Ray ayuda a los desarrolladores

para analizar y depurar aplicaciones distribuidas,

y te proporciona un mapa de servicios

que es una representación visual de todos los componentes

que conforman su solicitud.

Y si desea ejecutar X-Ray en su instancia EC2,

entonces el agente o daemon de X-Ray debe estar instalado

y usamos el X-Ray SDK para instrumentar

su aplicación o configurarla para enviar trazas a X-Ray.

Y solo para recordarles, esto es lo que la radiografía

Service Map se ve y proporciona

una vista de extremo a extremo de las solicitudes a medida que viajan

a través de su aplicación.

Por lo tanto, proporcionará información sobre la latencia,

Códigos de estado HTTP y mensajes de error también,

y este tipo de información se puede utilizar

para solucionar problemas de conectividad y rendimiento.

Y X-Ray está integrado con muchos servicios diferentes de AWS

como DynamoDB, Lambda y API Gateway.

Y también puedes instrumentar tus propias aplicaciones

o configurarlos para enviar datos a X-Ray,

por ejemplo, datos sobre solicitudes HTTP entrantes

en su aplicación.

Y las aplicaciones podrían ejecutarse en EC2,

Elastic Beanstalk, sistemas locales,

e incluso en Elastic Container Service.

Y si está ejecutando X-Ray en Elastic Container Service,

entonces es una buena práctica ejecutar el demonio X-Ray

en su propia imagen de Docker,

ejecutándose junto con su aplicación,

para que no lo instale en el mismo contenedor Docker

ejecutando su aplicación.

Y usamos anotaciones de rayos X

para agregar pares clave-valor definidos por el usuario

a sus datos de rayos X, y eso le permitirá filtrar,

indexar y buscar los datos, por ejemplo,

buscando un juego especifico

utilizando el nombre del juego o ID del juego.

En el almacenamiento en caché de API Gateway y el almacenamiento en caché de API Gateway

por supuesto mejora el rendimiento de sus API

almacenando en caché la salida de las llamadas a la API

para evitar llamar a su aplicación backend cada vez.

Las respuestas se almacenan en caché durante el período TTL,

y el TTL predeterminado o tiempo de vida es de 300 segundos,

o 5 minutos.

Y reduce el número de llamadas a la API

hecho a su aplicación porque API Gateway

devuelve la respuesta almacenada en caché en lugar de hacer

una nueva solicitud a su aplicación.

Y esto mejora el rendimiento y reduce la latencia.

para sus usuarios finales.

Y luego otra forma en que API Gateway

gestiona el rendimiento de sus aplicaciones

es mediante el uso de estrangulamiento,

y API Gateway tiene un límite predeterminado de 10,000

solicitudes por segundo y 5.000 solicitudes simultáneas,

y estos son límites por región.

Si supera el límite, API Gateway

devuelve un error, y es un error 429,

y es posible que también vea el mensaje "Demasiadas solicitudes".

Y API Gateway utiliza la limitación para evitar que su API

de estar abrumado por demasiadas solicitudes.

Y recuerde, estos son límites predeterminados,

para que pueda realizar una solicitud a AWS Support

para aumentar tus límites.

Y finalmente, tenemos Advanced API Gateway.

Así que recuerda que puedes importar

sus propias API utilizando archivos de definición externos

y el formato admitido es OpenAPI,

que antes se conocía como Swagger.

Y cuando se trata de aplicaciones heredadas

que utilizan el protocolo SOAP, puede configurar

API Gateway como paso de servicio web,

o puede usar API Gateway para convertir

la respuesta XML que recibe de la aplicación

al formato JSON.

Así que eso es todo por esta conferencia.

Si tiene alguna pregunta, por favor hágamelo saber.

De lo contrario, siéntase libre de pasar a la siguiente lección.

¡Gracias!