[ENGLISH]

Hello,

Cloud Gurus and welcome to this lecture which is part 1 of the summary

of the serverless section, including all of my exam tips.

So let's begin with serverless 101.

And serverless enables you to build scalable applications quickly without

managing any servers.

It is extremely low cost and serverless applications are events driven.

So you are only charged when your code is executed.

And AWS handles the heavy lifting.

So you can focus on writing code and building your application instead

of configuring servers. On to Lambda and Lambda,

of course, is extremely cost-effective. You pay only when your code executes.

It scales automatically, and it is event driven.

So Lambda functions are triggered by an event or action.

They are also independent and each event will trigger a single function.

And Lambda is, of course,

serverless along with other technologies like API Gateway, DynamoDB,

S3, SNS, and SQS.

And do be aware of the kind of services that can be used to trigger a

Lambda function. And here are just a few examples.

So we've got DynamoDB, Kinesis, SQS,

Application Load Balancer, API Gateway, CloudFront,

S3, SNS, SES, CloudFormation,

CloudWatch, CodeCommit, and CodePipeline.

And that's not an exhaustive list.

It's just the ones that we'll be talking about in this course.

Moving on to API Gateway. And remember,

an API is like the front door to your application.

And API Gateway provides an endpoint to your applications running in

AWS.

It is serverless so it is low cost and scales automatically,

and it provides throttling.

So you can throttle API Gateway to prevent your application from being

overloaded by too many requests.

And everything is logged to CloudWatch. For example, API calls,

latencies, and errors as well. And if you remember,

we built our own serverless website and we use Lambda to run some

Python code.

We used API Gateway to provide an endpoint that we used it to trigger our

Lambda function on demand,

and we used S3 to host our website files.

And then when we accessed the website, the code in our index.html

made a role to the API Gateway endpoint, which triggered the Lambda function.

And the response returned by Lambda was displayed on our web page.

On to version control with Lambda.

And $latest is always the latest version of the code you uploaded to

Lambda,

and we can use Lambda versioning and aliases to point our applications

to a specific version of our Lambda code if we don't want to

use $latest. For example,

we can create an alias to point to the version of our Lambda code that we

want to use in production.

And this is an example of an alias.

However, if we want to use $latest,

then that will be the latest version of our code.

But just remember that if your application is using an alias

instead of $latest,

then remember that it will not automatically use the new code when you upload

it. So you will need to update your alias if you want it to reference a new

version of code that you have just uploaded.

On to the Lambda concurrent executions

limit.

And it's important to understand that a limit exists and the default limit

is 1000 concurrent executions per second.

And if you are running a serverless website like we do at A Cloud Guru,

then it is likely that you are going to hit the limit at some point.

And if you do hit the limit,

you will start to see invocations being rejected.

And you'll see a 429 HTTP status code,

which means there are too many requests coming in and you have hit the

concurrent executions limit. And this is only a default limit.

So the remedy is to get the limit raised by AWS support.

And if you're running a critical application,

then you can use reserved concurrency to guarantee a set number of

concurrent executions are always available for your critical function.

So that is the end of part 1.

And if you'd like to join me for part 2 of the serverless summary,

then I will see you in the next lecture. Thank you.

[SPANISH]

hola ,

Cloud Gurus y bienvenidos a este conferencia cual es parte 1 de el resumen

de el sin servidor sección , incluyendo todo de mi examen consejos \_

Así que vamos comenzar con sin servidor 101.

y sin servidor permite tú a construir escalable aplicaciones rápidamente sin

gerente cualquier servidor.

Él es extremadamente bajo costo y sin servidor las aplicaciones son eventos conducido \_

Entonces eres solo \_ cargado cuando su código es ejecutado \_

Y manijas de AWS el trabajo pesado.

Para que puedas concentrarte en escribiendo codigo y construccion su solicitud en cambio

de configurar servidores. En a Lambda y Lambda,

de supuesto , es extremadamente rentable \_ Tú pagar solo cuando su código ejecuta \_

Él escamas automáticamente , y es evento conducido \_

Entonces se activan las funciones de Lambda por un evento o acción \_

ellos también son independiente y cada uno evento voluntad activar una sola función .

Y Lambda es , de curso ,

sin servidor a lo largo de con otro tecnologías como API Gateway, DynamoDB ,

S3, SNS y SQS.

Y sé consciente de el amable de servicios que se puede utilizar a desencadenar un

función lambda . Y aquí hay solo algunos \_ ejemplos \_

Así que hemos consiguió DynamoDB , Kinesis , SQS,

Equilibrador de carga de aplicaciones , API Gateway, CloudFront ,

S3, SNS, SES, Formación en la nube ,

CloudWatch , CodeCommit y CodePipeline .

y eso es no un exhaustivo lista \_

Es justo el unos eso estaremos hablando \_ acerca de esto curso \_

Moviente en a la puerta de enlace API. Y recuerda ,

una API es como el frente puerta a su aplicación \_

Y API Gateway proporciona un punto final a su aplicaciones que se ejecutan en

AWS.

Él es sin servidor por lo que es bajo costo y escalas automáticamente ,

y eso proporciona estrangulamiento \_

Por lo tanto, puede acelerar API Gateway para prevenir su solicitud de ser

sobrecargado por también muchos solicitudes \_

y todo es registrado a Reloj en la nube . Para ejemplo , llamadas API ,

latencias y errores también . \_ y si tú recuerda ,

nosotros construido nuestro propio sin servidor sitio web y usamos Lambda para ejecutar algunos

Código Python .

Nosotros usó API Gateway para proporcionar un punto final eso nosotros usado él a desencadenar nuestro

función lambda en demanda ,

y nosotros usamos S3 para alojar nuestro archivos del sitio web .

y luego cuando nosotros accedido el sitio web , el código en nuestro index.html

hizo un papel para el punto final de API Gateway , que motivado la función Lambda .

Y la respuesta volvió por Lambda fue desplegado en nuestra página web.

En a Control de versiones con Lambda.

Y $ último es siempre el el último versión de el código tú subido a

lambda,

y podemos usar versiones y alias de Lambda para punto nuestro aplicaciones

a un especifico versión de nuestro código lambda si nosotros no desear a

use $ último . Para ejemplo ,

podemos crear \_ un alias para punto a el versión de nuestro código lambda eso nosotros

desear para usar en la producción .

y esto es un ejemplo de un alias

Sin embargo , si nosotros desear para usar $ último ,

entonces eso será el \_ el último versión de nuestro código \_

Pero justo recordar eso si su solicitud es usando un alias

en cambio de $ más reciente ,

entonces recordar eso él voluntad no usar automáticamente el nuevo código cuando tú subir

eso \_ entonces tu voluntad necesidad a actualizar tu alias si tú desear él a hacer referencia a un nuevo

versión de código eso tú tener justo subido \_

En a la lambda concurrente ejecuciones

límite \_

y es importante a entender que un limite existe y el límite predeterminado

es 1000 concurrente ejecuciones por segundo .

y si está ejecutando un servidor sin servidor sitio web como hacemos en A Cloud Guru ,

entonces él es probable eso te vas \_ golpear el \_ límite en algunos punto \_

y si golpeas el \_ límite ,

tú voluntad comenzar a ver invocaciones ser rechazado \_

y lo harás ver un código de estado HTTP 429 ,

cual medio también hay \_ muchos peticiones entrando y tu haber golpeado el

concurrente ejecuciones límite \_ y esto es sólo un límite predeterminado .

Entonces el recurso es a conseguir el límite aumentó por el soporte de AWS .

y si estás ejecutando un crítico aplicación ,

entonces puedes usar reservado simultaneidad a garantizar un número fijo de

concurrente Las ejecuciones son siempre disponible para su crítico función \_

Entonces eso es el fin de parte 1

y si tú lo harías como a únete a mí para parte 2 de el sin servidor resumen ,

entonces lo haré ver tu en el próximo conferencia \_ Agradecer tu \_