[ENGLISH]

Hello, Cloud Gurus and welcome to this lesson,

which is part two of the Developer Theory Summary.

CodeArtifact is an artifact repository that makes it easy

for developers to find the software packages that they need.

A package is a bundle of software used

in your software development processes

and developers can use CodeArtifact

to find approved packages and also publish their own.

And we can also use CodeArtifact

to access packages from a public repository.

And to do that, create an upstream repository

with an external connection to pull packages

from the external public repository.

And this is what it looks like.

So we create our upstream repository,

configure the external connection to the public repository

and then we can pull down packages

that exist within the public repository

and allow our users to access them

from our own CodeArtifact Repository.

CodePipeline is a continuous integration

and continuous delivery service,

orchestrating your end-to-end software release process based

on a workflow that you define.

Code Pipeline will automatically trigger your pipeline

as soon as a change is detected

in your source code repository.

And it integrates with loads of different AWS services

and third-party tools as well.

So it integrates with things like CodeCommit,

CodeBuild and CodeDeploy.

And you can also use third-party tools like GitHub

for your source control and Jenkins for building your code.

And it also works with Elastic Beanstalk, CloudFormation,

Lambda, Elastic Container Service, and EC2 as well.

And speaking of Elastic Container Service, remember

that containers are like a virtual operating environment

with everything that your software needs to run,

including libraries, system tools, code and runtime.

This allows applications to be built

using independent stateless components on microservices

running in multiple containers.

Where can we run our containers?

On Elastic Container Service, also known as ECS.

So ECS can run your containers on clusters

of virtual machines.

There's also the serverless option called Fargate,

which means you can run your containers without worrying

about the underlying EC2 instances.

However, if you'd like more control over your environment,

let's say you wanna control the installation,

the configuration, and the management

of your compute environment,

then go for the EC2 option within ECS.

And then we have ECR, or Elastic Container Registry,

and this is where you store your container images.

Onto CloudFormation.

And CloudFormation allows you to manage, configure

and provision AWS infrastructure

as YAML or JSON code.

And you will need to be familiar

with the different sections in a CloudFormation template.

So first of all, we've got parameters which allows you

to input custom values like the name of an SSH key pair.

The conditions section allows you to provision resources

according to the environment.

For instance, you might want to attach

a different security group

to EC2 instances in your development environment

to your production environment.

The resources section is mandatory

and it describes the AWS resources that CloudFormation

is going to create, like the EC2 instances,

S3 buckets and security groups, et cetera.

The mapping section allows you to create custom mappings,

for example, mapping the region

to a specific AMI that you want to use for that region.

And then we have the transform section.

And this allows you to reference code that is located in S3,

for example, reusable snippets of CloudFormation code.

And it can also be used to specify the use

of the serverless application model

to deploy serverless resources.

The serverless application model is used to define

and provision serverless applications using CloudFormation.

We use sam package to package the application

and upload it to S3.

And sam deploy is used to deploy a serverless application

using CloudFormation.

CloudFormation nested stacks allow you

to reuse CloudFormation code

so that you don't need to copy and paste it

every time you want to reuse it.

And it's really useful for frequently used configurations,

like load balances or web or application servers.

And to define a nested stack,

just create the CloudFormation template

that you want to reuse and store it in S3.

Then reference it using the resources section

of a new template.

And the resource type is going to be stack.

CloudFormation allows you to export values

from one stack and import them into another.

And we use the output section of the stack

to export the values.

And in this example, we're exporting the subnet ID

and the security group from this stack.

And then in a subsequent stack,

we can then import these values

using the ImportValue function.

And let's say, for example,

we wanted to create an EC2 instance

and associate the subnet

and security group that was created by the previous stack,

we can use the ImportValue function to do that.

Amplify is a service that allows front end developers

to create full stack-web and mobile applications on AWS.

It has libraries that integrate with Cognito, S3, Lambda

and API Gateway to create a reliable backend

for your application.

And the two different Amplify services

that you need to know are firstly Amplify Hosting,

which provides static website hosting that integrates

with your code repository for CI/CD functionality.

And then there's also Amplify Studio.

And this is a visual tool that's used

to configure your front end

and the backend of your application.

The Cloud Development Kit enables you to deploy applications

and AWS resources using a programming language

of your choice, like TypeScript, Python, Java, .NET or Go.

With CDK, you define your application using apps,

stacks and constructs.

An app is a container for one or more stacks.

A stack is a unit of deployment

like a CloudFormation stack,

and a construct is used to define the AWS resources.

After defining everything using the programming language

of your choice, we then use the CDK

to synthesize a CloudFormation template.

So we use the CDK CLI to create our CloudFormation template

from our code

and then we can go ahead and deploy it.

And to do that, first of all, we run cdk init

in our empty directory to create a new CDK project.

We compile the app using npm run build.

Use cdk synth to create our CloudFormation template

and then deploy the stack using cdk deploy

to deploy our stack using CloudFormation.

So that is it for this lesson.

If you have any questions, please let me know.

Otherwise, I will see you in the next one.

Thank you.

[SPANISH]

Hola, Cloud Gurus y bienvenidos a esta lección,

que es la segunda parte del Resumen de la teoría del desarrollador.

CodeArtifact es un repositorio de artefactos que facilita

para que los desarrolladores encuentren los paquetes de software que necesitan.

Un paquete es un paquete de software utilizado

en sus procesos de desarrollo de software

y los desarrolladores pueden usar CodeArtifact

para encontrar paquetes aprobados y también publicar los suyos propios.

Y también podemos usar CodeArtifact

para acceder a paquetes desde un repositorio público.

Y para hacer eso, crea un repositorio upstream

con una conexión externa para tirar de paquetes

del repositorio público externo.

Y esto es lo que parece.

Entonces creamos nuestro repositorio upstream,

configurar la conexión externa al repositorio público

y luego podemos desplegar paquetes

que existen dentro del repositorio público

y permitir que nuestros usuarios accedan a ellos

de nuestro propio repositorio de CodeArtifact .

CodePipeline es una integración continua

y servicio de entrega continua,

orquestando su proceso de lanzamiento de software de extremo a extremo basado

en un flujo de trabajo que defina.

Code Pipeline activará automáticamente su canalización

tan pronto como se detecte un cambio

en su repositorio de código fuente.

Y se integra con un montón de diferentes servicios de AWS

y herramientas de terceros también.

Entonces se integra con cosas como CodeCommit ,

CodeBuild y CodeDeploy .

Y también puedes usar herramientas de terceros como GitHub

para su control de fuente y Jenkins para construir su código.

Y también funciona con Elastic Beanstalk, CloudFormation ,

Lambda, Elastic Container Service y EC2 también.

Y hablando de Elastic Container Service, recuerda

que los contenedores son como un entorno operativo virtual

con todo lo que su software necesita para funcionar,

incluyendo bibliotecas, herramientas del sistema, código y tiempo de ejecución.

Esto permite crear aplicaciones

usando componentes sin estado independientes en microservicios

funcionando en varios contenedores.

¿Dónde podemos ejecutar nuestros contenedores?

En Elastic Container Service, también conocido como ECS.

Entonces ECS puede ejecutar sus contenedores en clústeres

de máquinas virtuales.

También existe la opción sin servidor llamada Fargate ,

lo que significa que puede ejecutar sus contenedores sin preocuparse

sobre las instancias EC2 subyacentes.

Sin embargo, si desea tener más control sobre su entorno,

digamos que quiere controlar la instalación,

la configuración y la gestión

de su entorno informático,

luego vaya a la opción EC2 dentro de ECS.

Y luego tenemos ECR, o Elastic Container Registry,

y aquí es donde almacena las imágenes de su contenedor.

En CloudFormation .

Y CloudFormation le permite administrar, configurar

y aprovisionar la infraestructura de AWS

como código YAML o JSON.

Y tendrás que estar familiarizado

con las diferentes secciones en una plantilla de CloudFormation .

En primer lugar, tenemos parámetros que le permiten

para ingresar valores personalizados como el nombre de un par de claves SSH.

La sección de condiciones le permite provisionar recursos

según el entorno.

Por ejemplo, es posible que desee adjuntar

un grupo de seguridad diferente

a instancias EC2 en su entorno de desarrollo

a su entorno de producción.

La sección de recursos es obligatoria

y describe los recursos de AWS que CloudFormation

va a crear, como las instancias EC2,

S3 cubos y grupos de seguridad, etcétera.

La sección de mapeo le permite crear mapeos personalizados,

por ejemplo, mapear la región

a una AMI específica que desee utilizar para esa región.

Y luego tenemos la sección de transformación.

Y esto le permite hacer referencia al código que se encuentra en S3,

por ejemplo, fragmentos reutilizables del código de CloudFormation .

Y también se puede utilizar para especificar el uso.

del modelo de aplicación sin servidor

para implementar recursos sin servidor .

El modelo de aplicación sin servidor se utiliza para definir

y aprovisione aplicaciones sin servidor utilizando CloudFormation .

Usamos sam package para empaquetar la aplicación

y subirlo a S3.

Y sam deployment se usa para implementar una aplicación sin servidor

usando Formación de nubes .

Las pilas anidadas de CloudFormation le permiten

para reutilizar el código de CloudFormation

para que no tengas que copiar y pegar

cada vez que quieras reutilizarlo.

Y es realmente útil para configuraciones de uso frecuente,

como balances de carga o servidores web o de aplicaciones.

Y para definir una pila anidada,

simplemente cree la plantilla de CloudFormation

que desea reutilizar y almacenarlo en S3.

Luego haga referencia a él usando la sección de recursos.

de una nueva plantilla.

Y el tipo de recurso va a ser pila.

CloudFormation le permite exportar valores

de una pila e importarlos a otra.

Y usamos la sección de salida de la pila.

para exportar los valores.

Y en este ejemplo, estamos exportando la ID de subred

y el grupo de seguridad de esta pila.

Y luego, en una pila posterior,

podemos importar estos valores

utilizando la función ImportValue .

Y digamos, por ejemplo,

queríamos crear una instancia EC2

y asociar la subred

y grupo de seguridad que fue creado por la pila anterior,

podemos usar la función ImportValue para hacer eso.

Amplify es un servicio que permite a los desarrolladores front-end

para crear aplicaciones móviles y web de pila completa en AWS.

Tiene bibliotecas que se integran con Cognito , S3, Lambda

y API Gateway para crear un backend confiable

para su aplicación.

Y los dos servicios diferentes de Amplify

que necesita saber son, en primer lugar, Amplify Hosting,

que proporciona alojamiento de sitios web estáticos que integran

con su repositorio de código para la funcionalidad de CI/CD.

Y luego también está Amplify Studio.

Y esta es una herramienta visual que se utiliza

para configurar su interfaz

y el backend de su aplicación.

El kit de desarrollo en la nube le permite implementar aplicaciones

y recursos de AWS utilizando un lenguaje de programación

de su elección, como TypeScript , Python, Java, .NET o Go.

Con CDK, usted define su aplicación usando aplicaciones,

pilas y construcciones.

Una aplicación es un contenedor para una o más pilas.

Una pila es una unidad de implementación.

como una pila de CloudFormation ,

y se utiliza una construcción para definir los recursos de AWS.

Después de definir todo usando el lenguaje de programación.

de su elección, luego usamos el CDK

para sintetizar una plantilla de CloudFormation .

Así que usamos la CLI de CDK para crear nuestra plantilla de CloudFormation

de nuestro código

y luego podemos seguir adelante y desplegarlo.

Y para hacer eso, antes que nada, ejecutamos cdk en eso

en nuestro directorio vacío para crear un nuevo proyecto CDK.

Compilamos la aplicación usando npm run build.

Use el sintetizador cdk para crear nuestra plantilla de CloudFormation

y luego implemente la pila usando cdk deployment

para implementar nuestra pila usando CloudFormation .

Eso es todo por esta lección.

Si tiene alguna pregunta, por favor hágamelo saber.

De lo contrario, te veré en la próxima.

Gracias.