[ENGLISH]

Hello Cloud Gurus and welcome to this lecture,

which is part two of my EC2 summary.

So let's begin with Route 53

and remember, Route 53 is Amazon's DNS service

and it allows you to map a domain name

to an EC2 instance,

an elastic load balancer or S3 bucket.

And if you remember, we created a web server running httpd,

we added an application load balancer

and then we configured our hosted zone

with a new alias and an A record

to map our domain name

to the DNS address of our application load balancer

and this is one of the most common use cases for Route 53.

And just be aware of the following Route 53 terminology.

So we used a hosted zone,

which is a container for DNS records for our domain

and alias is what allows you to route traffic

addressed to the zone apex

or the top of the DNS namespace,

for example, ilovecloud.com

and send it to a resource within AWS,

for example, an elastic load balancer

and an A record allows you to route traffic to a resource

such as a web server, using an IPv4 address.

Moving on to the AWS CLI,

keep in mind the principle of least privilege

and always give your users the minimum amount of access

required to do their job.

And it's best practice to use groups,

so create identity access management groups

and assign your users to groups.

Group permissions are assigned using IAM policy documents

and your users will automatically inherit

the permissions of the group.

When it comes to generating the secret access key,

remember that you will only see this once

and if you lose it,

then you can delete the access key ID

and secret access key and regenerate them.

And if you do that, you will need to run AWS configure again

and provide the new credentials.

Don't share your key pairs

and each developer should have their own access key ID

and secret access key

and just like passwords,

they should never be shared.

And the AWS CLI supports Linux, Windows and MacOS,

so you can install it on your local machine

and you can also use it on EC2 instances as well.

And of course, when we launch a Linux to EC2 instance,

you get the AWS CLI pre-installed.

Onto using roles with EC2

and we can use a role to give an EC2 instance

access to AWS resources like S3.

And to do that, we need to create

an Identity access management role with S3 access.

Create an EC2 instance

and attach the role that we just created

and then we should be able to access S3

from our EC2 instance.

And using roles in this way is the preferred option

from a security perspective.

And it allows you to avoid hard coding your credentials,

so roles allow you to provide access

without having to manage access keys

and secret access keys

and manually configure them on every single EC2 instance.

We use IAM policies to control a roles permissions

and you can update a policy attached to a role

and it will take immediate effect.

And you can attach and detach roles

to running EC2 instances

without having to stop or terminate the instances

And if you remember, we tried that with our own EC2 instance

and the effect was immediate.

Onto RDS or Relational Database Service.

So we've got a few different RDS database types

there's SQLServer, Oracle,

MySQL, PostgreSQL,

MariaDB and Amazon Aurora

and RDS is for online transaction processing workloads.

So this is great for processing lots of small transactions

like customer orders, banking transactions,

payments and booking systems

and It is not suitable for online analytics processing.

So instead, we should use RedShift for OLAP

and data warehousing type tasks

like analyzing large amounts of data,

reporting and sales forecasting.

And it's important to remember the differences

between automated backups

and database snapshots

and they are also known as manual snapshots.

So with automated backups,

these are enabled by default

and you define the backup window.

They provide point-in-time snapshots

plus transaction logs.

You can define a retention period of up to 35 days

and they can be used to recover your database

to any point in time within the retention period.

Whereas database snapshots are user-initiated and ad-hoc.

They provide point-in-time snapshots only,

there's no retention period

and they are stored indefinitely

until you delete them

and they can be used to backup your database instance

to a known state

and restore to that specific state at any time,

for example, you might run a database snapshot

before making a change to the database.

And when it comes to encrypting your RDS database,

you will need to enable encryption at creation time.

So you cannot enable it later

and when you encrypt your RDS database

it includes all the underlying storage,

automated backups,

snapshots, logs and read replicas as well.

So it's going to encrypt everything.

And RDS integrates with KMS,

so it uses the AWS Key Management Service

for AES-256 bit encryption.

And if you have any existing RDS instance

which you need to encrypt,

of course you cannot add encryption later

but what you can do is take a snapshot of your database

then encrypt the snapshot

and create a new RDS instance

from your encrypted snapshot.

Moving on to multi-AZ and read replica

and it's important to understand the differences

between these two technologies.

So with multi-AZ, you get an exact copy

of your production database

in another availability zone.

These are used for disaster recovery,

so in the event of a failure,

RDS will automatically failover to the standby instance.

Whereas a read replica is a read-only copy

of your primary database

and this can be in the same AZ,

cross-AZ or cross-region.

And read replicas are used to increase

or scale read performance.

And a read replica is great for read-heavy workloads

because it takes the load off your primary database

for read-only workloads,

for example, if you need to run

a business intelligence reporting jobs.

Another way to improve read performance for RDS

is to use ElastiCache.

And if you remember, ElastiCache is an in-memory cache

designed to improve read performance

for read-heavy databases

and there are two options available.

Firstly, we've got memcached

and this is an in-memory, key-value data store.

And this is the one to use

if object caching is your primary goal

and you want to keep things as simple as possible,

you don't need persistence or multi-AZ

and you don't need support

for advanced data types or data sorting.

Whereas with redis, that is also an in-memory

key-value data store

but this is the one to use

if you are performing data sorting and ranking,

such as in gaming leaderboards.

And it's also a great choice

if you have advanced data types

such as lists and hashes

and if you need data persistence and multi-AZ.

So memcached is your basic option

whereas redis provides enterprise features

like persistence and multi-AZ.

And finally, we have parameter store,

which is a service that allows you

to store confidential information

such as passwords, database connection strings

and licensed codes, et cetera.

You can store your values as plain text or encrypt them

and reference them using the parameter name,

for example, in a bootstrap script.

And parameter store is integrated

with lots of different AWS services.

So you can use it with EC2, CloudFormation, Lambda

and it also supports CodeBuild, CodePipeline

and CodeDeploy.

So that's it for this lecture,

if you have any questions please let me know.

Otherwise I will see you in the next lecture.

Thank you.

[SPANISH]

Hola Cloud Gurus y bienvenidos a esta conferencia,

que es la segunda parte de mi resumen de EC2.

Así que comencemos con la Ruta 53

y recuerda, Route 53 es el servicio DNS de Amazon

y te permite mapear un nombre de dominio

a una instancia EC2,

un balanceador de carga elástico o un cubo S3.

Y si recuerdas, creamos un servidor web que ejecuta httpd ,

agregamos un balanceador de carga de aplicaciones

y luego configuramos nuestra zona hospedada

con un nuevo alias y un registro A

para mapear nuestro nombre de dominio

a la dirección DNS de nuestro balanceador de carga de aplicaciones

y este es uno de los casos de uso más comunes para la Ruta 53.

Y tenga en cuenta la siguiente terminología de la Ruta 53.

Así que usamos una zona alojada,

que es un contenedor de registros DNS para nuestro dominio

y alias es lo que le permite enrutar el tráfico

dirigido al vértice de la zona

o la parte superior del espacio de nombres DNS,

por ejemplo, ilovecloud.com

y enviarlo a un recurso dentro de AWS,

por ejemplo, un equilibrador de carga elástico

y un registro A le permite enrutar el tráfico a un recurso

como un servidor web, utilizando una dirección IPv4.

Pasando a la CLI de AWS,

tener en cuenta el principio de privilegio mínimo

y siempre brinde a sus usuarios la cantidad mínima de acceso

requeridos para hacer su trabajo.

Y es una buena práctica usar grupos,

así que cree grupos de administración de acceso de identidad

y asigne sus usuarios a grupos.

Los permisos de grupo se asignan mediante documentos de políticas de IAM

y tus usuarios heredarán automáticamente

los permisos del grupo.

Cuando se trata de generar la clave de acceso secreta,

recuerda que solo verás esto una vez

y si lo pierdes,

entonces puede eliminar la identificación de la clave de acceso

y clave de acceso secreta y regenerarlas.

Y si lo hace, deberá ejecutar AWS configure nuevamente

y proporcionar las nuevas credenciales.

No comparta sus pares de claves

y cada desarrollador debe tener su propia ID de clave de acceso

y clave de acceso secreta

y al igual que las contraseñas,

deben ser compartidos.

Y la AWS CLI es compatible con Linux, Windows y MacOS ,

para que pueda instalarlo en su máquina local

y también puede usarlo en instancias EC2.

Y, por supuesto, cuando lanzamos una instancia de Linux a EC2,

obtiene la CLI de AWS preinstalada.

Sobre el uso de roles con EC2

y podemos usar un rol para dar una instancia EC2

acceso a recursos de AWS como S3.

Y para hacer eso, necesitamos crear

un rol de administración de acceso a la identidad con acceso S3.

Crear una instancia EC2

y adjunte el rol que acabamos de crear

y luego deberíamos poder acceder a S3

de nuestra instancia EC2.

Y usar roles de esta manera es la opción preferida.

desde una perspectiva de seguridad.

Y le permite evitar codificar sus credenciales,

por lo que los roles le permiten proporcionar acceso

sin tener que gestionar claves de acceso

y claves de acceso secretas

y configurarlos manualmente en cada instancia de EC2.

Usamos políticas de IAM para controlar los permisos de roles

y puede actualizar una política adjunta a un rol

y tendrá efecto inmediato.

Y puede adjuntar y separar roles

para ejecutar instancias EC2

sin tener que detener o terminar las instancias

Y si recuerdas, lo intentamos con nuestra propia instancia EC2

y el efecto fue inmediato.

En RDS o servicio de base de datos relacional.

Así que tenemos algunos tipos diferentes de bases de datos RDS

hay Servidor SQL , Oracle,

MySQL, PostgreSQL ,

Maria DB y Amazon Aurora

y RDS es para cargas de trabajo de procesamiento de transacciones en línea.

Así que esto es genial para procesar muchas transacciones pequeñas.

como pedidos de clientes, transacciones bancarias,

de pagos y reservas

y No es adecuado para el procesamiento de análisis en línea.

Entonces, en su lugar, deberíamos usar RedShift para OLAP

y tareas de tipo almacenamiento de datos

como analizar grandes cantidades de datos,

elaboración de informes y previsión de ventas.

Y es importante recordar las diferencias.

entre copias de seguridad automatizadas

e instantáneas de la base de datos

y también se conocen como instantáneas manuales.

Entonces, con las copias de seguridad automatizadas,

estos están habilitados por defecto

y usted define la ventana de copia de seguridad.

Proporcionan instantáneas de un punto en el tiempo

además de registros de transacciones.

Puede definir un período de retención de hasta 35 días

y se pueden utilizar para recuperar su base de datos

a cualquier punto en el tiempo dentro del período de retención.

Mientras que las instantáneas de la base de datos son iniciadas por el usuario y ad-hoc.

Solo proporcionan instantáneas de un punto en el tiempo,

hay período de retención

y se almacenan indefinidamente

hasta que los borres

y se pueden usar para hacer una copia de seguridad de su instancia de base de datos

a un estado conocido

y restaurar a ese estado específico en cualquier momento,

por ejemplo, puede ejecutar una instantánea de la base de datos

antes de realizar un cambio en la base de datos.

Y cuando se trata de cifrar su base de datos RDS,

deberá habilitar el cifrado en el momento de la creación.

Entonces no puedes habilitarlo más tarde

y cuando cifra su base de datos RDS

incluye todo el almacenamiento subyacente,

automatizadas ,

instantáneas , registros y réplicas de lectura también.

Así que va a cifrar todo.

Y RDS se integra con KMS,

por lo que utiliza el Servicio de administración de claves de AWS

para el cifrado AES-256 bits.

Y si tiene alguna instancia RDS existente

que necesitas encriptar,

por supuesto, no puede agregar cifrado más tarde

pero lo que puede hacer es tomar una instantánea de su base de datos

luego cifrar la instantánea

y crea una nueva instancia de RDS

de su instantánea cifrada.

Pasar a multi-AZ y leer réplica

y es importante entender las diferencias

entre estas dos tecnologías.

Entonces, con multi-AZ, obtienes una copia exacta

de su base de datos de producción

en otra zona de disponibilidad.

Estos se utilizan para la recuperación ante desastres,

por lo que en caso de falla,

RDS conmutará por error automáticamente a la instancia en espera.

Mientras que una réplica de lectura es una copia de solo lectura

de su base de datos primaria

y esto puede estar en la misma AZ,

entre AZ o entre regiones.

Y las réplicas de lectura se utilizan para aumentar

o escalar el rendimiento de lectura.

Y una réplica de lectura es ideal para cargas de trabajo de lectura intensiva.

porque quita la carga de su base de datos principal

para cargas de trabajo de solo lectura,

por ejemplo, si necesita ejecutar

un trabajo de informes de inteligencia de negocios.

Otra forma de mejorar el rendimiento de lectura para RDS

es usar ElastiCache .

Y si recuerda, ElastiCache es un caché en memoria

diseñado para mejorar el rendimiento de lectura

para bases de datos de lectura pesada

y hay dos opciones disponibles.

En primer lugar, tenemos memcached

y este es un almacén de datos clave-valor en memoria.

Y este es el que hay que usar

si el almacenamiento en caché de objetos es su objetivo principal

y quieres mantener las cosas lo más simples posible,

necesitas persistencia o multi-AZ

y no necesitas apoyo

para tipos de datos avanzados o clasificación de datos.

Mientras que con redis , eso también es en memoria

almacén de datos de clave-valor

pero este es el que hay que usar

si está realizando clasificación y clasificación de datos,

como en las tablas de clasificación de juegos.

Y también es una gran opción.

si tiene tipos de datos avanzados

como listas y hashes

y si necesita persistencia de datos y multi-AZ.

Así que memcached es tu opción básica

mientras redis proporciona funciones empresariales

como persistencia y multi-AZ.

Y finalmente, tenemos el almacén de parámetros,

que es un servicio que te permite

para almacenar información confidencial

como contraseñas, cadenas de conexión de bases de datos

y códigos de licencia, etcétera.

Puede almacenar sus valores como texto sin formato o cifrarlos

y haga referencia a ellos usando el nombre del parámetro,

por ejemplo, en un script de arranque.

Y el almacén de parámetros está integrado

con muchos servicios diferentes de AWS.

Entonces puede usarlo con EC2, CloudFormation , Lambda

y también es compatible con CodeBuild , CodePipeline

y CodeDeploy .

Así que eso es todo por esta conferencia,

si tiene alguna pregunta, por favor hágamelo saber.

De lo contrario, te veré en la próxima conferencia.

Gracias.