[ENGLISH]

Hello, Cloud Gurus,

and welcome to this lecture.

And in this lecture,

we are going to get our hands dirty with CodeCommit.

So let's run through all the things

that we're going to be doing.

So firstly, we're going to create a new repository

in CodeCommit.

We're going to add a file into the repository.

We're going to create a new branch,

which is like a new working area in the repository.

We're then going to update our file.

And then finally, we're going to merge our changes

from our branch back into the master repo.

And if you would like to complete all the steps with me,

then please make sure you are working

in your own AWS account,

because some of the steps in this demo

will not work in the AWS Sandbox.

So if you'd like to get started,

please join me in the console.

So here I am in the console, and you will find CodeCommit,

it's down under the Developer Tools.

So just click on CodeCommit.

Click on Get started.

And they have recently updated the console

for CodeCommit to make it easier,

and it is much more user-friendly than it used to be.

So that's why you can see this banner up here saying,

do I want to return to the old experience?

Well, I definitely don't,

and I'm going to stick with the new experience.

So this lab uses the new console.

So the first thing we're going to do

is create our new repository,

and we're going to start by giving it a name,

and I'm just going to call this ccrepo.

You can add an optional description.

And then click Create.

So that has successfully created our repository.

And it gives you a description of the different types

of connection that you can have into your repository.

So you can have either HTTPS or SSH.

And this just describes the different connection types

from your clients.

So whenever you connect to the repository,

it's always done in a secure way,

and your connections will be encrypted in transit,

using either HTTPS or SSH.

But we're not going to configure

those kind of connections right now.

We're just going to interact with our repository,

using the console.

So I'm just going to close that section down.

And now we're going to add our very first file

into the repository.

So click on Add File, and we're going to create a new file.

So we're just going to create a really simple text file.

We need to give it a name.

Add the name of the author.

And an email address associated

with this change that we're making.

And we can also add an optional commit message

as we commit our change into the master.

And then select Commit Changes.

So that file has now been committed to the master,

and you can see that we can view the file from the console.

If we click on the name of our repository,

we can see the file is located there,

and we can see that it's located in the master branch.

And of course, the master branch,

that's our golden source or our master source of our code.

But just imagine that you needed to make a change.

You needed to update the file,

for example, if you were developing a new feature,

or if you were fixing a bug in an existing piece of code.

Well, you wouldn't want to make that change

on the master copy of your data.

So that is where branches come in.

And what you would do instead

is you would create a new branch,

and you would make those updates on your own branch.

And a new branch is simply a new working area,

and it's a copy or a clone of the original master branch.

So let's go ahead and create a new branch.

So let's click on Branches, Create Branch,

and we're going to give it a name

and we're just going to call it mybranch.

And we're going to select to

create the branch from the master.

So it's going to be a clone of the master.

So click on Create Branch.

So that's our new branch created.

And if we click on the branch,

it has exactly the same file in.

If we click on the file, it looks exactly the same.

So now we can go ahead and edit our file.

So I'm just going to click on Edit

and add another line into the file.

Once again, we need to add the author name.

So, who made the edit? What's their email address?

And we can add an optional message again.

And click on Commit Changes.

So now we've updated the file,

but we've only updated it in mybranch.

And if we go to Branches,

we've got our 2 different branches there.

And you can see

that we last commit made a commit on mybranch just now,

but the original branch is still 3 minutes out of date.

And at this stage, in practice,

you would probably want to get your updates peer reviewed.

You might want to run some tests,

say, some automated tests, preferably, or some manual tests

to make sure that you're happy with the changes

and that they're good to go and that you're confident

that you want to merge them into the master.

So once you're happy

and you're ready to merge your changes into the master,

the next thing you need to do is create a pull request.

And this is going to apply the changes

that we made in our branch into the master repository.

So it's going to merge the change we made

into that master branch.

So in order to do that, we click on Create Pull Request,

and we need to define the source and destination.

So the destination is going to be the master,

and the source is going to be mybranch.

So select your branch, and then click on Compare.

And what this does is it runs a number of tests

to make sure that there's no conflicts

between mybranch and the master.

So for example, one of the checks it will do

is it will check that the master has not been updated

by somebody else while you were preparing your own changes.

And if that has happened,

that means your own branch is going to be out of date.

And if your branch is out of date,

then you're going to have to clone the code

from the master again and add your changes

into the brand-new copy of the code.

But for us right now, there shouldn't be any conflicts

between your branch and the master.

So we should be able to go ahead

and merge our change into the master branch.

So first of all, we need to give our merge a title

to describe what we're doing.

So I'm just going to type

"Merging mybranch into the master."

And if we scroll down,

it shows you this visual representation

of the changes that we're actually making.

It's compared the 2 files,

and it's detected that we are adding this line here.

And if you've ever used GitHub before, it's very similar.

You get this same view in GitHub where it will show you

if you've deleted something,

changed something, or added something,

and it would color-code the differences

between the 2 files.

So it makes it really easy to identify the changes

that are going to be made,

especially if somebody else is reviewing your code.

So let's go ahead and click Create,

and that will create our pull request.

So now our pull request has been created,

and the next thing we need to do is click on Merge,

and that will merge our branch into the master repository.

And it gives you the option

of deleting the source branch after the merge.

So do you want to delete mybranch

after the merge has been completed or not?

And generally you would want it to delete that source branch

after you have merged.

At a later date, if you want to do another change,

then it's best practice to just create another branch

and start again with a clean branch.

So let's go ahead and click Merge Pull Request,

and hopefully that should be successful.

So it says mybranch has been merged into master.

And if we click through these tabs,

it just gives you an overview

of the events that have happened.

So it will show you, you've created our pull request.

We merged the pull request.

We can take a look at the changes.

And we can also view the commits as well.

And if we go back to our Branches,

you can see that mybranch has now been deleted.

So we only have the master branch.

If we click on the master branch, click on the file,

and you can see our file has been updated.

So it's very similar to GitHub if you ever used that before.

And in addition to working in the console,

you can also work with CodeCommit from your local machine,

and you can interact with the repository

using the standard Git commands.

And you don't really need to know that for the exam.

You don't need to memorize any Git commands

or anything like that.

The main thing you need to know for the exam

is that CodeCommit is a source control system.

It enables collaboration between different teams

and different users working on the same project.

And it's secure,

allowing you to connect using HTTPS and SSH.

So your data is always protected in transit.

So really, just remember for the exam what CodeCommit is

and what it is used for.

And there's just one last thing

that I want to show you on CodeCommit,

which is under Settings.

And you can actually configure notifications

using SNS and CloudWatch to send notifications

to subscribers when certain events happen

in your CodeCommit repository.

So for example, you can send a notification

when somebody does a pull request or closes a pull request

or when somebody comments on a pull request as well.

And this is where you would set it up.

So you can set up a CloudWatch event rule

and also an SNS topic

for people to subscribe to the notifications.

So that is the end of this lecture.

If you have any questions, please let me know.

If not, feel free to move on to the next lecture. Thank you.

[SPANISH]

Hola, gurús de la nube,

y bienvenidos a esta conferencia.

Y en esta conferencia,

nos vamos a ensuciar las manos con CodeCommit .

Así que repasemos todas las cosas

que vamos a estar haciendo.

En primer lugar, vamos a crear un nuevo repositorio.

en CodeCommit .

Vamos a añadir un archivo al repositorio.

Vamos a crear una nueva sucursal,

que es como una nueva área de trabajo en el repositorio.

Entonces vamos a actualizar nuestro archivo.

Y finalmente, vamos a fusionar nuestros cambios

desde nuestra rama de vuelta al repositorio principal.

Y si quieres completar todos los pasos conmigo,

entonces por favor asegúrese de que está trabajando

en su propia cuenta de AWS,

porque algunos de los pasos de esta demostración

funcionará en AWS Sandbox.

Entonces, si desea comenzar,

por favor, únase a mí en la consola.

Así que aquí estoy en la consola, y encontrarás CodeCommit ,

está debajo de las Herramientas para desarrolladores.

Así que simplemente haga clic en CodeCommit .

Haga clic en Comenzar.

Y han actualizado recientemente la consola.

para CodeCommit para hacerlo más fácil,

y es mucho más fácil de usar de lo que solía ser.

Es por eso que puedes ver esta pancarta aquí arriba que dice:

Quiero volver a la vieja experiencia?

Bueno, definitivamente no,

y voy a seguir con la nueva experiencia.

Este laboratorio usa la nueva consola.

Así que lo primero que vamos a hacer

es crear nuestro nuevo repositorio,

y vamos a empezar por darle un nombre,

y solo voy a llamar a esto ccrepo .

Puede agregar una descripción opcional.

Y luego haga clic en Crear.

Así que ha creado con éxito nuestro repositorio.

Y te da una descripción de los diferentes tipos.

de conexión que puede tener en su repositorio.

Entonces puede tener HTTPS o SSH.

Y esto solo describe los diferentes tipos de conexión.

de sus clientes.

Entonces, siempre que se conecte al repositorio,

se hace de forma segura,

y sus conexiones serán encriptadas en tránsito,

utilizando HTTPS o SSH.

Pero no vamos a configurar

ese tipo de conexiones en este momento.

Solo vamos a interactuar con nuestro repositorio,

utilizando la consola.

Así que voy a cerrar esa sección.

Y ahora vamos a agregar nuestro primer archivo

en el repositorio.

Haga clic en Agregar archivo y vamos a crear un nuevo archivo.

Así que vamos a crear un archivo de texto realmente simple.

Tenemos que darle un nombre.

Añade el nombre del autor.

Y una dirección de correo electrónico asociada

con este cambio que estamos haciendo.

Y también podemos agregar un mensaje de confirmación opcional

a medida que comprometemos nuestro cambio en el maestro.

Y luego seleccione Confirmar cambios.

Entonces ese archivo ahora se ha enviado al maestro,

y puede ver que podemos ver el archivo desde la consola.

Si pulsamos sobre el nombre de nuestro repositorio,

podemos ver que el archivo se encuentra allí,

y podemos ver que está ubicado en la rama maestra.

Y por supuesto, la rama maestra,

esa es nuestra fuente dorada o nuestra fuente maestra de nuestro código.

Pero imagina que necesitas hacer un cambio.

Necesitabas actualizar el archivo,

por ejemplo, si estuviera desarrollando una nueva característica,

o si estaba arreglando un error en una pieza de código existente.

Bueno, no querrías hacer ese cambio.

en la copia maestra de sus datos.

Así que ahí es donde entran las ramas.

Y lo que harías en su lugar

es que crearías una nueva rama,

y haría esas actualizaciones en su propia sucursal.

Y una nueva sucursal es simplemente una nueva área de trabajo,

y es una copia o un clon de la rama maestra original.

Así que sigamos adelante y creemos una nueva rama.

Hagamos clic en Sucursales, Crear sucursal,

y le vamos a poner un nombre

y simplemente lo llamaremos mybranch .

Y vamos a seleccionar para

crear la rama desde el maestro.

Así que va a ser un clon del maestro.

Así que haga clic en Crear sucursal.

Así que esa es nuestra nueva sucursal creada.

Y si hacemos clic en la rama,

tiene exactamente el mismo archivo en.

Si hacemos clic en el archivo, se ve exactamente igual.

Así que ahora podemos continuar y editar nuestro archivo.

Así que voy a hacer clic en Editar

y agregue otra línea en el archivo.

Una vez más, necesitamos agregar el nombre del autor.

Entonces, ¿quién hizo la edición? ¿Cuál es su dirección de correo electrónico?

Y podemos agregar un mensaje opcional nuevamente.

Y haga clic en Confirmar cambios.

Así que ahora hemos actualizado el archivo,

pero solo lo hemos actualizado en mybranch .

Y si vamos a Sucursales,

Tenemos nuestras 2 sucursales diferentes allí.

y puedes ver

que hicimos la última confirmación hizo una confirmación en mybranch hace un momento,

pero la rama original todavía tiene 3 minutos de desactualización.

Y en esta etapa, en la práctica,

probablemente querrá que sus actualizaciones sean revisadas por pares .

Es posible que desee realizar algunas pruebas,

digamos , algunas pruebas automatizadas, preferiblemente, o algunas pruebas manuales

para asegurarse de que está satisfecho con los cambios

y que están listos para ir y que tienes confianza

que desea fusionarlos en el maestro.

Así que una vez que estés feliz

y está listo para fusionar sus cambios en el maestro,

lo siguiente que debe hacer es crear una solicitud de extracción.

Y esto va a aplicar los cambios.

que hicimos en nuestra sucursal en el repositorio principal.

Así que va a fusionar el cambio que hicimos

en esa rama maestra.

Entonces, para hacer eso, hacemos clic en Crear solicitud de extracción,

y necesitamos definir la fuente y el destino.

Así que el destino va a ser el maestro,

y la fuente va a ser mybranch .

Así que seleccione su sucursal y luego haga clic en Comparar.

Y lo que esto hace es ejecutar una serie de pruebas

para asegurarse de que no haya conflictos

entre mybranch y el maestro.

Entonces, por ejemplo, uno de los controles que hará

es comprobará que el maestro no se ha actualizado

por otra persona mientras preparaba sus propios cambios.

Y si eso ha pasado,

eso significa que su propia sucursal estará desactualizada.

Y si tu sucursal está desactualizada,

entonces vas a tener que clonar el código

del maestro nuevamente y agregue sus cambios

en la nueva copia del código.

Pero para nosotros en este momento, no debería haber ningún conflicto.

entre su rama y el maestro.

Así que deberíamos ser capaces de seguir adelante

y fusionar nuestro cambio en la rama principal.

Entonces, antes que nada, debemos darle a nuestra fusión un título

para describir lo que estamos haciendo.

Así que solo voy a escribir

"Fusionando mi rama con el maestro".

Y si nos desplazamos hacia abajo,

te muestra esta representación visual

de los cambios que realmente estamos haciendo.

Se comparan los 2 archivos,

y se detecta que estamos agregando esta línea aquí.

Y si alguna vez has usado GitHub antes, es muy similar.

Obtienes esta misma vista en GitHub donde te mostrará

si has borrado algo,

cambió algo, o agregó algo,

y codificaría por colores las diferencias

entre los 2 archivos.

Así que hace que sea realmente fácil identificar los cambios.

que se van a hacer,

especialmente si alguien más está revisando su código.

Así que sigamos adelante y hagamos clic en Crear,

y eso creará nuestra solicitud de extracción.

Así que ahora se ha creado nuestra solicitud de extracción,

y lo siguiente que debemos hacer es hacer clic en Fusionar,

y eso fusionará nuestra rama en el repositorio principal.

Y te da la opción

de eliminar la rama de origen después de la fusión.

Entonces, ¿quieres eliminar mybranch ?

después de que la fusión se haya completado o no?

Y, en general, le gustaría que elimine esa rama de origen

después de que te hayas fusionado.

En una fecha posterior, si desea hacer otro cambio,

entonces es una buena práctica simplemente crear otra rama

y empezar de nuevo con una rama limpia.

Así que sigamos adelante y hagamos clic en Merge Pull Request,

y con suerte eso debería tener éxito.

Entonces dice que mybranch se ha fusionado con master.

Y si hacemos clic en estas pestañas,

solo te da una visión general

de los hechos que han sucedido.

Entonces le mostrará que ha creado nuestra solicitud de extracción.

Fusionamos la solicitud de extracción.

Podemos echar un vistazo a los cambios.

Y también podemos ver los compromisos también.

Y si volvemos a nuestras Sucursales,

puede ver que mybranch ahora se ha eliminado.

Así que solo tenemos la rama maestra.

Si hacemos clic en la rama maestra, haga clic en el archivo,

y puede ver que nuestro archivo ha sido actualizado.

Entonces es muy similar a GitHub si alguna vez lo usaste antes.

Y además de trabajar en la consola,

puede trabajar con CodeCommit desde su máquina local,

y puedes interactuar con el repositorio

utilizando los comandos estándar de Git .

Y realmente no necesitas saber eso para el examen.

No necesitas memorizar ningún comando de Git

o algo por el estilo.

Lo principal que debes saber para el examen.

es que CodeCommit es un sistema de control de código fuente.

Permite la colaboración entre diferentes equipos.

y diferentes usuarios trabajando en el mismo proyecto.

Y es seguro,

permitiéndole conectarse usando HTTPS y SSH.

Por lo tanto, sus datos siempre están protegidos en tránsito.

De verdad, recuerda para el examen qué es CodeCommit

y para que sirve.

Y solo hay una última cosa

que quiero mostrarles en CodeCommit ,

que está en Configuración.

Y puedes configurar notificaciones

usando SNS y CloudWatch para enviar notificaciones

a los suscriptores cuando suceden ciertos eventos

en su repositorio de CodeCommit .

Entonces, por ejemplo, puede enviar una notificación

cuando alguien hace una solicitud de extracción o cierra una solicitud de extracción

o cuando alguien comenta sobre una solicitud de extracción también.

Y aquí es donde lo configurarías.

Para que pueda configurar una regla de eventos de CloudWatch

y también un tema SNS

para que la gente se suscriba a las notificaciones.

Así que ese es el final de esta conferencia.

Si tiene alguna pregunta, por favor hágamelo saber.

Si no, siéntase libre de pasar a la siguiente lección. Gracias.