

AWS CODECOMMIT: CREACIÓN, CONFIGURACIÓN E INTERACCIÓN CON UN REPOSITORIO UTILIZANDO COMANDOS DE GIT

En el ámbito del desarrollo de software, el control de versiones es un elemento crítico para el éxito de cualquier proyecto. AWS CodeCommit es una solución de alojamiento de repositorios de código privados totalmente administrada que forma parte de la gama de servicios de AWS (Amazon Web Services). Proporciona una plataforma segura y escalable para alojar, administrar y colaborar en el código fuente de aplicaciones y proyectos de software, simplificando la colaboración entre equipos de desarrollo.

Esta práctica se enfocará en explorar las características esenciales de AWS CodeCommit y aprender cómo utilizarlo de manera efectiva para gestionar el ciclo de vida del código fuente. Abordaremos conceptos clave, como la creación de repositorios privados, la gestión de ramas y la revisión de cambios. A lo largo de la práctica, adquirirás experiencia en la creación y administración de repositorios de código, el seguimiento de cambios y la solución de posibles conflictos.

Requerimientos:

- Disponer de acceso a los recursos de AWS a través de un *sandbox* de AWS Academy
- Disponer de acceso a un entorno con la herramienta GIT instalada y el instalador de paquetes para Python (PIP)

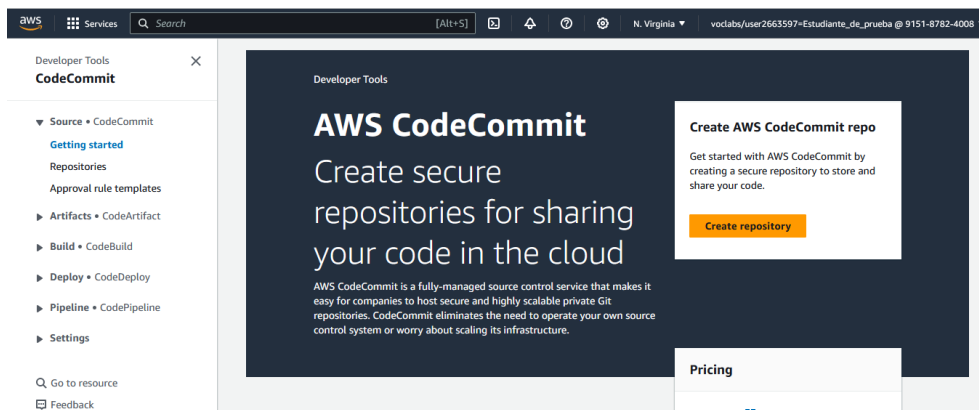
Arquitectura propuesta:



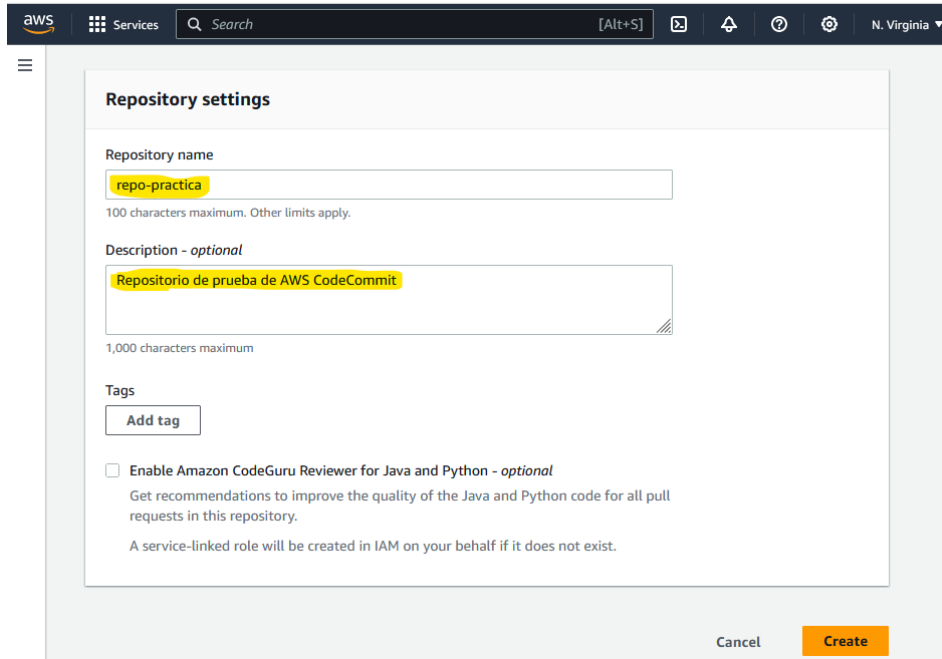
Realización:

CREACIÓN DE UN REPOSITORIO DE AWS CODECOMMIT

- 1) El primer paso para crear un repositorio es acceder a la Consola de Administración de AWS y desde allí acceder al servicio de AWS CodeCommit. Desde la siguiente ventana, se presiona el botón **Create repository**:



- 2) Desde la ventana siguiente, introducimos el nombre del repositorio. Para esta práctica, se utilizará como **Repository name** el nombre *repo-practica*. Introduciremos además en el campo **Description** el valor *Repositorio de prueba de AWS CodeCommit*:



La casilla de verificación que permite habilitar las revisiones de Amazon CodeGuru Reviewer. Este servicio realiza un análisis estático de código automatizado que utiliza aprendizaje automático (*Machine Learning*) para detectar potenciales defectos en el código que son difíciles de encontrar en códigos en Java y Python, ofreciendo sugerencias para mejorar dicho código. No obstante, el servicio Amazon CodeGuru Reviewer no está disponible en los entornos de AWS Academy Learner Labs por el momento.

Para terminar, presionamos el botón **Create**.

- 3) Una vez creado el repositorio, es necesario obtener las credenciales para el acceso. Estas credenciales pueden obtenerse de dos formas:
- Creando un **conjunto de credenciales de AWS CodeCommit** desde AWS IAM para el usuario de IAM que vaya a acceder a AWS CodeCommit. Esta solución no es viable en esta práctica, ya que el entorno de AWS Academy Learner Labs no ofrece un usuario de IAM para su uso, sino un usuario federado a partir de un proveedor de identidad externo.
 - Utilizando las **credenciales de acceso programático al AWS Academy Learner Lab**. Esta forma será la que emplearemos, ya que nuestro usuario federado dispone de permisos para gestionar y acceder al servicio de AWS CodeCommit. Utilizaremos para ello un comando de AWS CodeCommit llamado **Credential Helper**, que permitirá el acceso a los repositorios a partir de nuestra clave de acceso temporal. Sin embargo, el inconveniente que presenta esta opción es que, al tener las claves de acceso a AWS Academy Learner Labs un carácter temporal, será necesario actualizarlas cada vez que se inicie el laboratorio.

Connection steps

HTTPS | SSH | HTTPS (GRC)

Warning: You are signed in using federated access or temporary credentials. The only supported connection method for these sign-in types is to use the credential manager included with the AWS CLI, as documented below. To configure a connection using SSH or Git credentials over HTTPS, sign in as an IAM user.

Step 1: Prerequisites

You must use a Git client that supports Git version 1.7.9 or later to connect to an AWS CodeCommit repository. If you do not have a Git client, you can install one from Git downloads. [View Git downloads page](#)

You must have an AWS CodeCommit managed policy attached to your IAM user, belong to a CodeStar project team, or have the equivalent permissions. [Learn how to create and configure an IAM user for accessing AWS CodeCommit](#) | [Learn how to add team members to an AWS CodeStar Project](#)

Step 2: Set up the AWS CLI Credential Helper

Set up your connection to AWS CodeCommit repositories using the credential helper included in the AWS CLI. This is the only connection method for AWS CodeCommit repositories that does not require an IAM user, so it is the only method that supports root access, federated access, and temporary credentials. [Learn more](#)

Additional details

You can find more detailed instructions in the documentation. [View documentation](#)

ACCESO A UN REPOSITORIO DE AWS CODECOMMIT DESDE UN ENTORNO EXTERNO

- 4) Para configurar el acceso programático con las credenciales del AWS Academy Learner Lab, previamente instalaremos la AWS CLI. Para ello, podemos hacerlo desde el siguiente enlace, eligiendo la forma de instalación en función del sistema operativo:

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

- 5) Una vez instalada la AWS CLI, desde el laboratorio, presionamos el botón **AWS Details** y a continuación el botón **Show** de la opción AWS CLI, tal y como muestra la figura:

Cloud Access

AWS CLI:
Copy and paste the following into ~/.aws/credentials

```
[default]
aws_access_key_id=ASIA5CHS5FV5G2NTP3ME
aws_secret_access_key=VW3gV1rDrHaDZ13nI32q+/JCAxyEsmggIe
s6U/2m
aws_session_token=Fwo6ZIXVYXZdEKj////////wEaDE1p0Pn3VV
x5ja6/y5K9AbXTzEng5FIzIaQX8n35yDg0w1eJkHkmd9ux2ht+otzPM
FjsSb/I110FAPhuuVwLhC66HRYwvNI4ku4UbFzV1zRDfeFupB13v
JsaOq1g3eG2y/kHeXsrdwbeIbx3ozSTROJfPulW8FOCFuxg8GTU0VH+uH
8ch391u8Ya4R21HX1L17a06NEIZJz8tr0BngmoofAEH6emS5TeuXcS5V
np6EuJZ1IhwCERYDrSad0IHU0civbsHtkv485zDhCja34mpBj1tpgop1K
Ygw/AUgQ8j7uBnGc7AxP9FverpF5rFLnEXP2UQd6B8PTauH/B8xrs
```

Desde aquí, copiamos el texto con la configuración de las credenciales de acceso a corto plazo y lo pegamos en un nuevo fichero llamado *credentials*, que lo almacenaremos en la ruta relativa al directorio inicial de usuario *.aws/credentials*. Si no existiese tal ruta, hay que crearla manualmente a partir del directorio inicial de usuario en el sistema operativo:

```

[default]
aws_access_key_id=ASIA5CNS5FW5G2NTP3ME
aws_secret_access_key=VV3gVlDrHaDZI3nIJ2Q+/jCAXyEsmPgIes6U/Zm
aws_session_token=FwoGZXIvYXZlEKj////////wEaDE1p0Pn3VVx5ja6/ySK9AbXTzEngSfIZtAqQXBm35yDg0wie
jKhKmd9uxZht+otzPMFjsSb/Ill0FAPMmuVmLoHce66hRMYWvNI4ku4UbfZvzRDfeWpB13vJsAq1g3eG2ykHeXsRdb
WebIbx3ozSTR0jfPuN0F0CFuxgBGTUDVN+uH8ch391uBYa4R21HX1LI7a06NEIZJz8tr0BngmoofAEH6emSSTeuXcSyVnp
GEUjZ1IhwCERYDrSadD1HUDcWbsMtkv48SzDhCjaJ4mpBjItpgop1KYgw/AUGQBJ7WbNgc7AxP9FverpFSrfLnEXPZUQd6
Bo8PTAuH/B8xrs
~
~
~
~
~

```

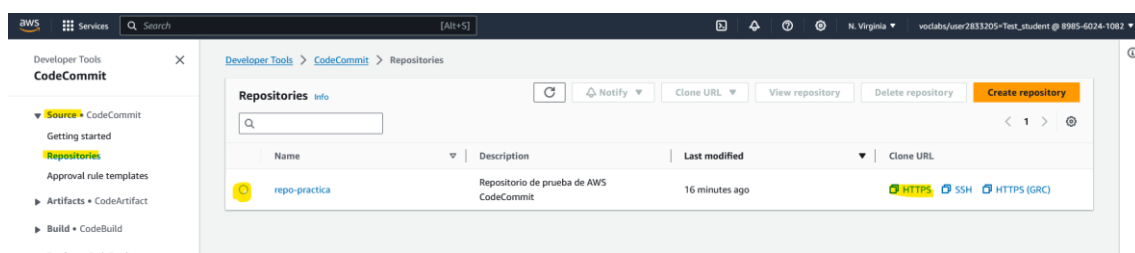
- 6) A continuación, indicaremos a Git que utilice el *Credential Helper* de AWS CodeCommit para autenticar el acceso al repositorio utilizando las credenciales configuradas. Para ello se ejecutan las órdenes:

```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```

La orden de la AWS CLI *aws codecommit credential-helper* no está pensada para ser ejecutada directamente, sino a través del comando *git config*.

- 7) Con el paso anterior, ya estaría configurado el acceso a nuestro repositorio. El acceso a un repositorio de AWS CodeCommit puede realizarse de tres maneras:
- **HTTPS.** El acceso puede realizarse mediante credenciales de AWS IAM o mediante el *Credential Helper* de AWS CodeCommit.
 - **SSH.** El acceso debe realizarse necesariamente mediante una clave privada, cargando previamente una clave pública en el usuario de AWS IAM. No es viable en esta práctica, ya disponemos de acceso muy limitado al servicio AWS IAM en AWS Academy Learner Labs.
 - **HTTPS (GRC).** Permite utilizar URIs de AWS CodeCommit para el acceso mediante HTTPS. Es necesario instalar la extensión de *Git* llamada **git-remote-codecommit**.

Ahora obtendremos la URL (HTTPS) de nuestro repositorio para poder probarlo. Para ello, volvemos a la consola de AWS CodeCommit, seleccionamos en el menú lateral la opción **Source / Repositories**, marcamos la casilla de nuestro repositorio y presionamos el enlace **HTTPS**:



Ahora sólo resta clonar nuestro repositorio mediante la orden *git clone* y pegando la URL de nuestro repositorio:

```

root@aws:~# git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/repo-practica
Cloning into 'repo-practica' ...
warning: You appear to have cloned an empty repository.
root@aws:~#

```

A partir de este momento, ya sería posible entrar en el repositorio local y utilizar las órdenes de Git para operar con él y sincronizarlo con el repositorio central en AWS CodeCommit.

- 8) Por otra parte, también es posible acceder a nuestro repositorio de AWS CodeCommit utilizando URIs personalizadas de AWS CodeCommit. Estas URIs tienen el formato siguiente:

codecommit::<región>://<nombre-repo>

Para poder operar con estas URIs mediante es necesario instalar una extensión de Git llamada **git-remote-codecommit**. Para instalar esta extensión es necesario disponer en nuestro entorno de el instalador de paquetes para Python (PIP). Ejecutamos la orden:

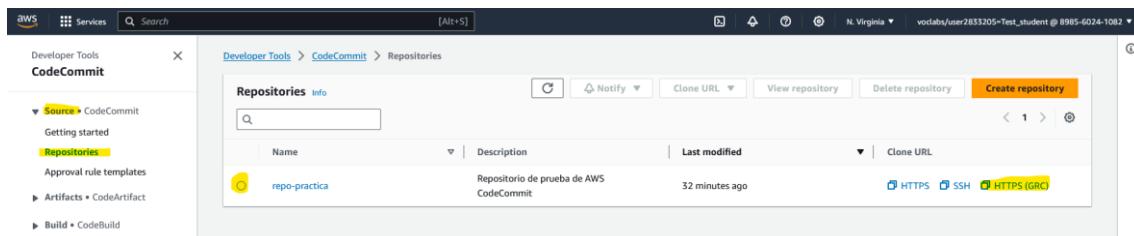
```
pip install git-remote-codecommit
```

```

root@aws:~# pip install git-remote-codecommit
Collecting git-remote-codecommit
  Using cached git_remote_codecommit-1.17-py3-none-any.whl
Requirement already satisfied: botocore>=1.17.0 in ./local/lib/python3.10/site-packages (from git-remote-codecommit) (1.29.158)
Requirement already satisfied: python-dateutil<3.0.0, >=2.1 in ./local/lib/python3.10/site-packages (from botocore>=1.17.0->git-remote-codecommit) (2.8.2)
Requirement already satisfied: jmespath<2.0.0, >=0.7.1 in ./local/lib/python3.10/site-packages (from botocore>=1.17.0->git-remote-codecommit) (1.0.1)
Requirement already satisfied: urllib3<1.27, >=1.25.4 in ./local/lib/python3.10/site-packages (from botocore>=1.17.0->git-remote-codecommit) (1.26.16)
Requirement already satisfied: six>=1.5 in ./local/lib/python3.10/site-packages (from python-dateutil<3.0.0, >=2.1->botocore>=1.17.0->git-remote-codecommit) (1.14.0)
Installing collected packages: git-remote-codecommit
Successfully installed git-remote-codecommit-1.17
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
root@aws:~#

```

- 9) Ahora obtendremos la URL (HTTPS-GRC) de nuestro repositorio para poder probarlo. Para ello, volvemos a la consola de AWS CodeCommit, seleccionamos en el menú lateral la opción **Source / Repositories**, marcamos la casilla de nuestro repositorio y presionamos el enlace **HTTPS (GRC)**:



- 10) Por último, eliminamos el repositorio creado anteriormente y clonamos de nuevo el repositorio utilizando la URI personalizada de AWS CodeCommit mediante las órdenes:

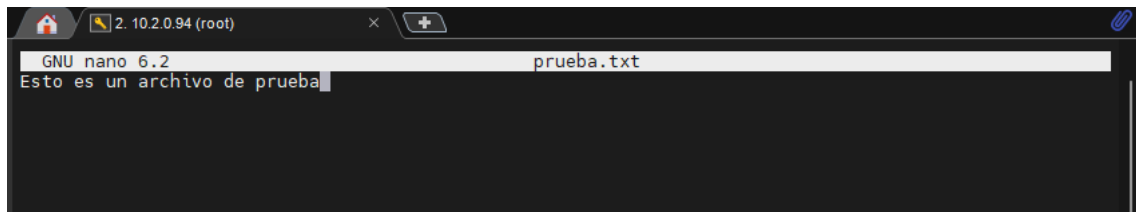
```
rm -Rf repo-practica
git clone codecommit::us-east-1://repo-practica
```

OPERACIONES SOBRE UN REPOSITORIO DE AWS CODECOMMIT

- 11) Una vez clonado el repositorio, accedemos al repositorio local y procedemos a crear la primera versión. Para ello, creamos un archivo *prueba.txt* mediante las siguientes órdenes:

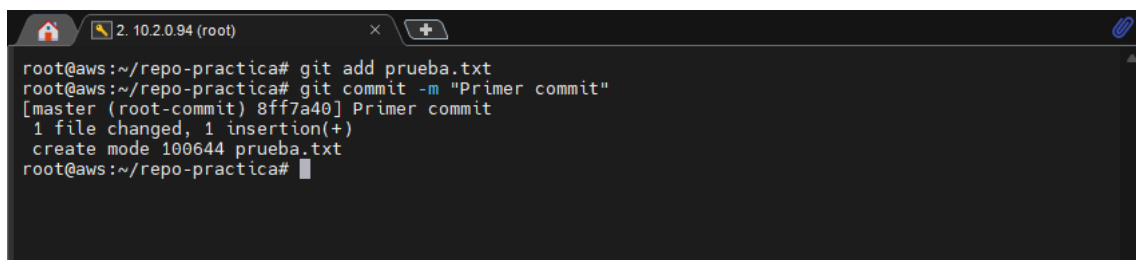
```
cd repo-practica
touch prueba.txt
nano prueba.txt
```

Introducimos contenido en el archivo *prueba.txt* y salvamos los cambios.



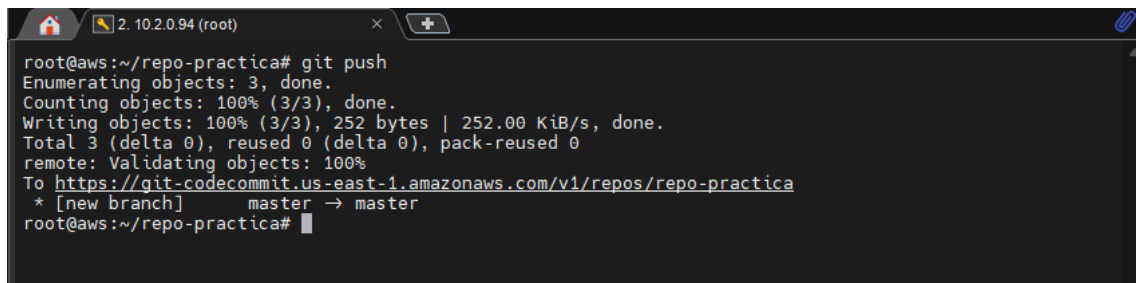
- 12) A continuación, añadimos el archivo al área de *Staging* y confirmamos los cambios con las siguientes órdenes:

```
git add prueba.txt
git commit -m "Primer commit"
```

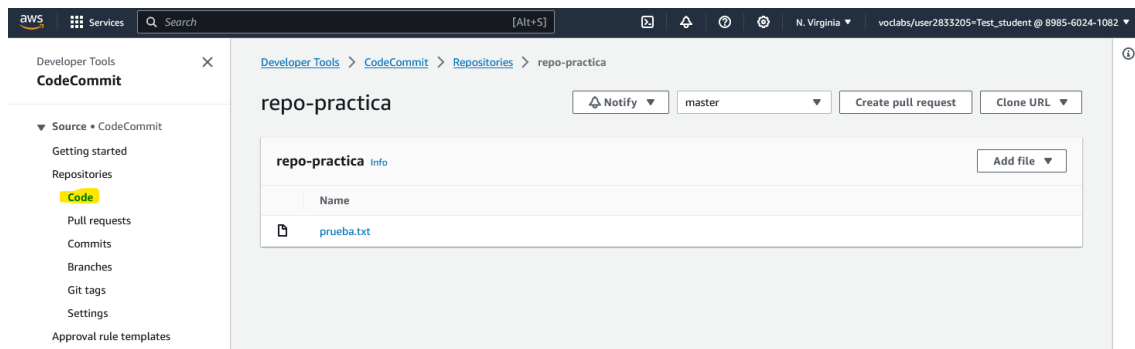


- 13) Por último, sincronizamos los cambios anteriores con el repositorio de AWS CodeCommit con la orden:

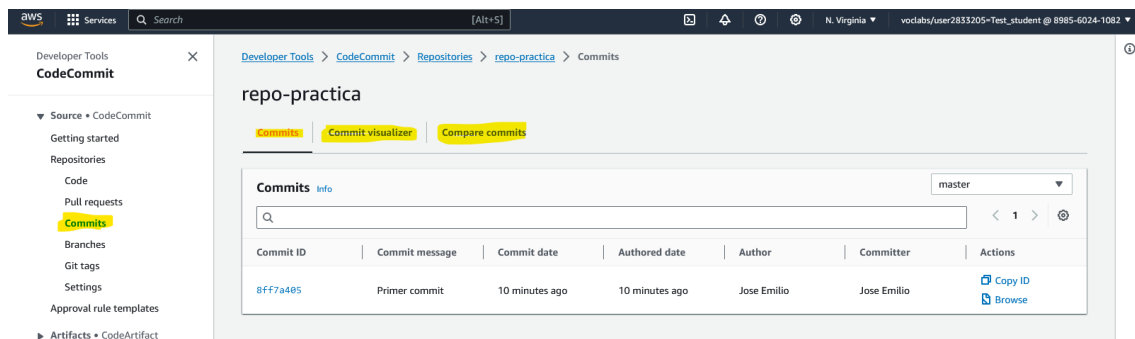
```
git push
```



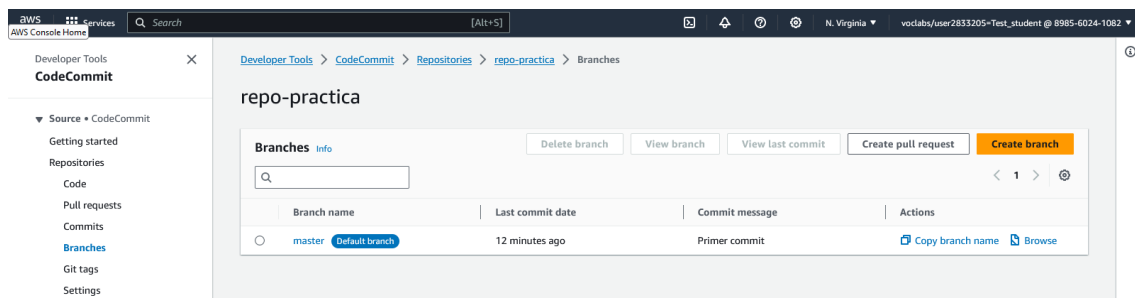
- 14) Si volvemos a la consola de AWS CodeCommit y accedemos a nuestro repositorio, podremos comprobar cómo se ha sincronizado con la nube de AWS:



- 15) Como puede comprobarse en la consola de AWS CodeCommit, podemos acceder a las diferentes versiones (*commit*) de nuestro repositorio sincronizado, accediendo a la opción **Commits** del menú lateral. Desde esta opción, no sólo podemos verificar nuestras versiones, sino tener una representación visual mediante grafos acíclicos dirigidos (GAD) de nuestras versiones (y en diferentes ramificaciones), así como poder comparar versiones en ramificaciones diferentes:



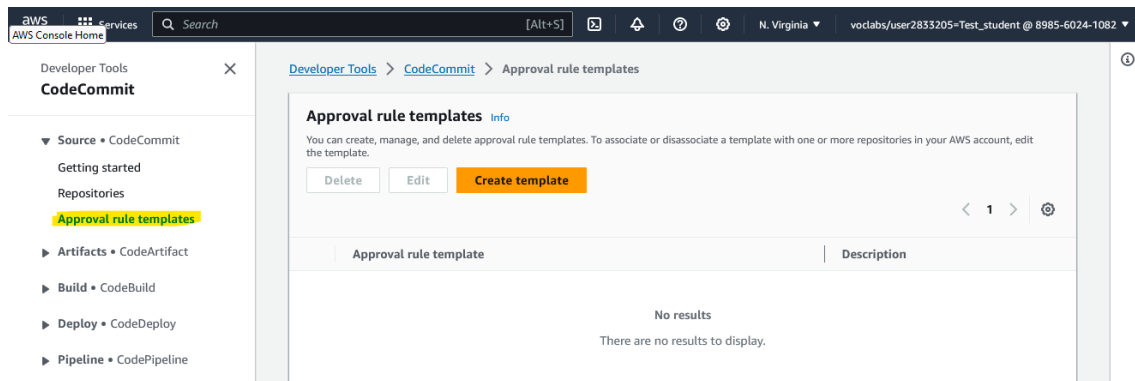
- 16) También es posible administrar las ramificaciones de nuestro repositorio accediendo a la opción **Branches** del menú lateral:



- 17) En un entorno real, los desarrolladores no disponen de permisos para fusionar los cambios realizados en una ramificación de desarrollo sobre la ramificación *master* (que, por lo general, se corresponde con la rama de producción, aunque esto depende de la estrategia de *branching* que se utilice). Es por ello por lo que, cuando se pretende crear una nueva *release* en producción debe solicitarse la fusión sobre la ramificación *master* mediante una **pull request**.

AWS CodeCommit dispone de herramientas para poder realizar una *pull request* para que sea validada y confirmada por el personal autorizado. Para crear un conjunto de usuarios autorizados para validar una *pull request* es necesario crear una **plantilla de regla de aprobación** y vincularla con

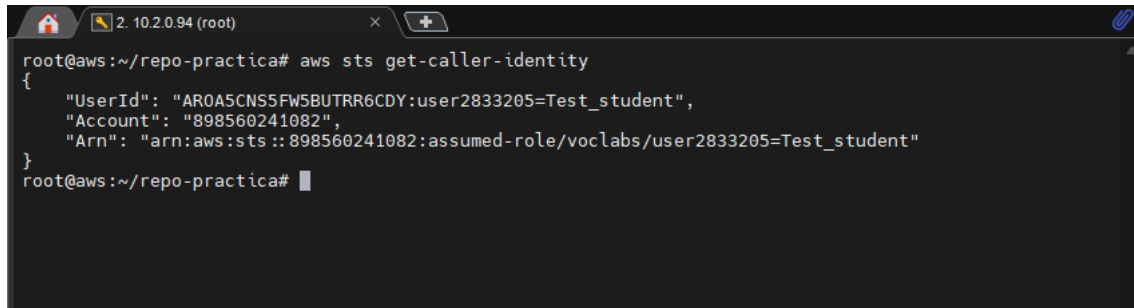
el repositorio al que aplica. Para ello, desde la consola de AWS CodeCommit, accedemos a la opción **Approval rule templates** del menú lateral y presionamos el botón **Create template**:



- 18) A continuación, se introduciría el nombre de la plantilla de reglas de aprobación. Se debería indicar también el número necesario de personas que deban aprobar una *pull request*, los miembros que estarían autorizados para realizar la aprobación y los repositorios a los que aplicaría esta regla, tal y como se muestra en la figura siguiente:

Para indicar la entidad autorizada para aprobar un *pull request* sobre nuestro repositorio, en este caso habríamos de añadir nuestro rol asumido en AWS Academy Learner Lab, para lo cual deberíamos obtener este valor ejecutando la siguiente orden en nuestro entorno y copiando el valor indicado en el atributo *Arn* del documento JSON devuelto:

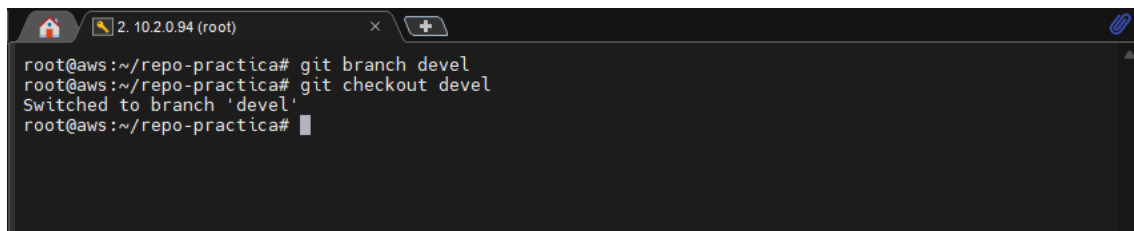
```
aws sts get-caller-identity
```



```
root@aws:~/repo-practica# aws sts get-caller-identity
{
  "UserId": "AROACSNS5FW5BUTRR6CDY:user2833205=Test_student",
  "Account": "898560241082",
  "Arn": "arn:aws:sts::898560241082:assumed-role/voclabs/user2833205=Test_student"
}
root@aws:~/repo-practica#
```

- 19) Volviendo a nuestro entorno, crearemos una nueva rama en nuestro repositorio local mediante la orden:

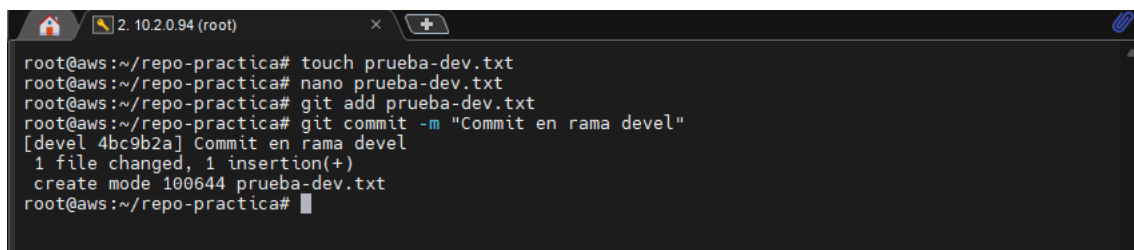
```
git branch devel
git checkout devel
```



```
root@aws:~/repo-practica# git branch devel
root@aws:~/repo-practica# git checkout devel
Switched to branch 'devel'
root@aws:~/repo-practica#
```

- 20) A continuación, creamos un nuevo archivo llamado *prueba-dev.txt* en la rama *devel*, introducimos contenido en él y creamos un nuevo *commit*, mediante las órdenes siguientes:

```
touch prueba-dev.txt
nano prueba-dev.txt
git add prueba-dev.txt
git commit -m "Commit en rama devel"
```



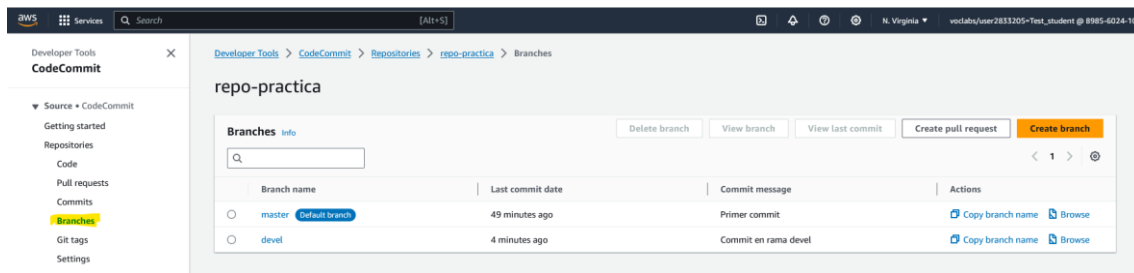
```
root@aws:~/repo-practica# touch prueba-dev.txt
root@aws:~/repo-practica# nano prueba-dev.txt
root@aws:~/repo-practica# git add prueba-dev.txt
root@aws:~/repo-practica# git commit -m "Commit en rama devel"
[devel 4bc9b2a] Commit en rama devel
1 file changed, 1 insertion(+)
create mode 100644 prueba-dev.txt
root@aws:~/repo-practica#
```

- 21) Por último, sincronizamos los cambios con el repositorio en AWS CodeCommit mediante las órdenes:

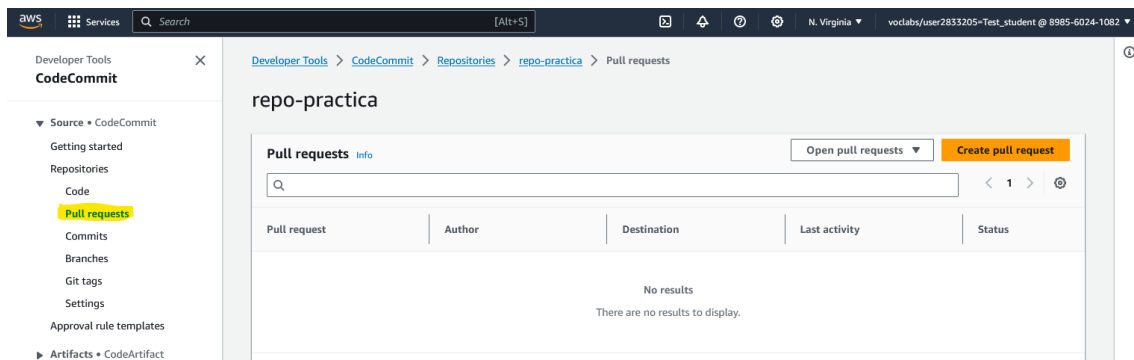
```
git push --set-upstream origin devel
```

```
root@aws:~/repo-practica# git push --set-upstream origin devel
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 323 bytes | 323.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/repo-practica
* [new branch]      devel -> devel
Branch 'devel' set up to track remote branch 'devel' from 'origin'.
root@aws:~/repo-practica#
```

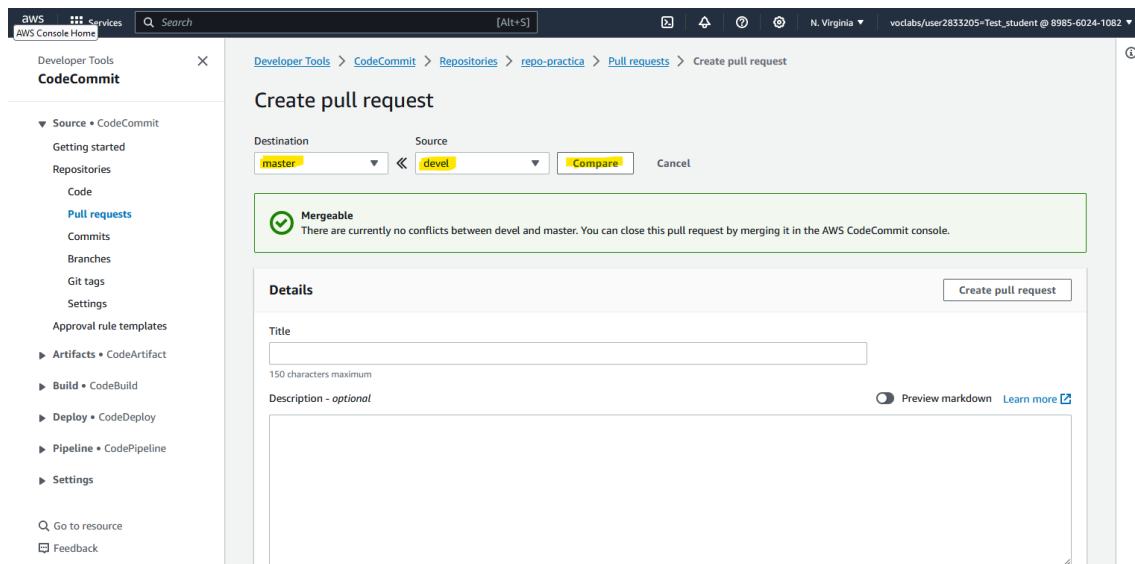
- 22) Si volvemos a la consola de administración de AWS CodeCommit, accediendo a la opción **Branches** podremos visualizar ambas ramificaciones, tanto *master* como *devel*.



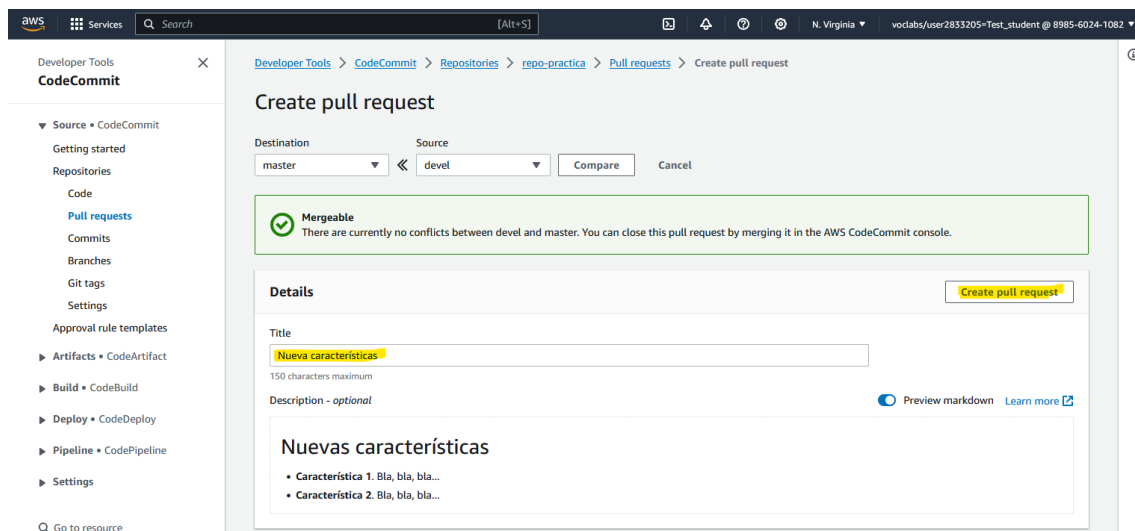
- 23) Si un desarrollador desee fusionar la ramificación *devel* sobre la ramificación *master* en un entorno real, debería crear una *pull request*. Para simular este caso, accederemos a la opción **Pull requests** del menú lateral y presionaremos el botón **Create pull request**:



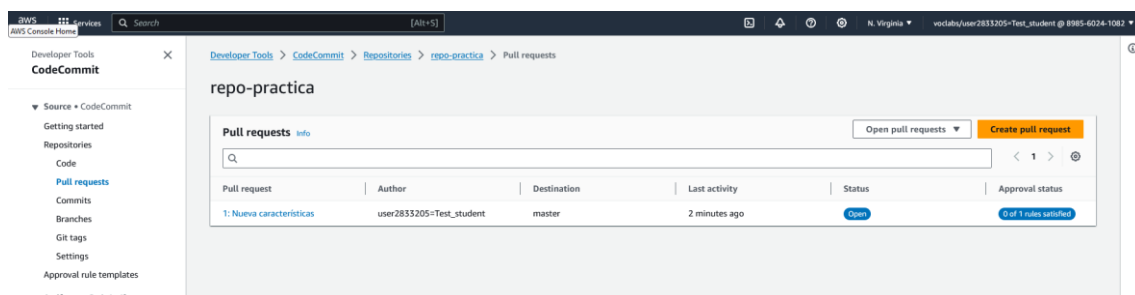
- 24) A continuación, seleccionaremos como destino la ramificación *master* y como origen la ramificación *devel*, y presionaremos el botón **Compare**:



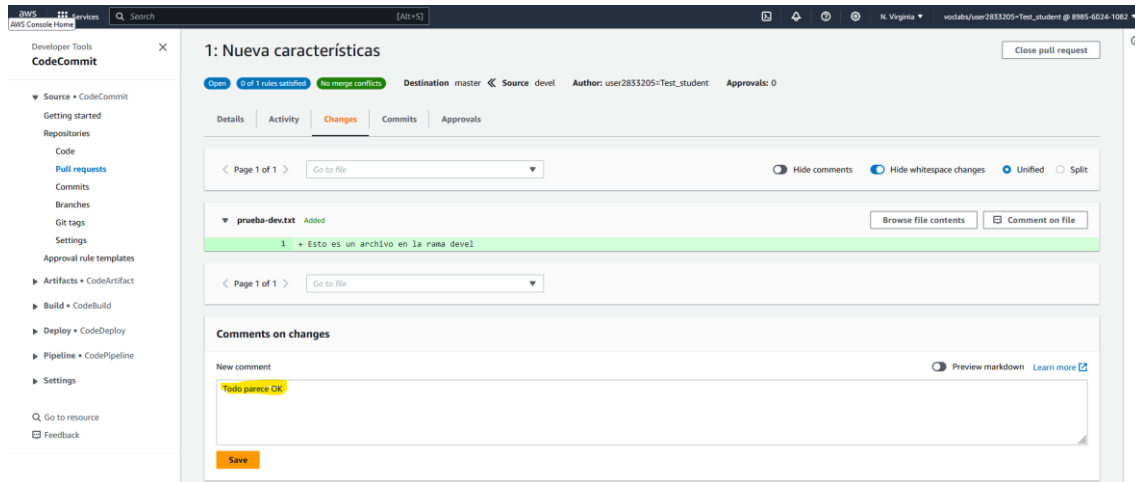
25) En el siguiente paso, introducimos el título y la descripción de nuestra *pull request* y presionamos el botón **Create pull request**:



26) Al volver de nuevo a la consola, podremos comprobar que se creó la *pull request*:



- 27) Si entramos dentro de nuestra *pull request*, deberíamos tener un botón llamado **Approve**. Sin embargo, AWS CodeCommit no permite que el creador de una *pull request* pueda aprobar su propia *pull request*. Esta es la razón por la que dicho botón no aparece en la interfaz

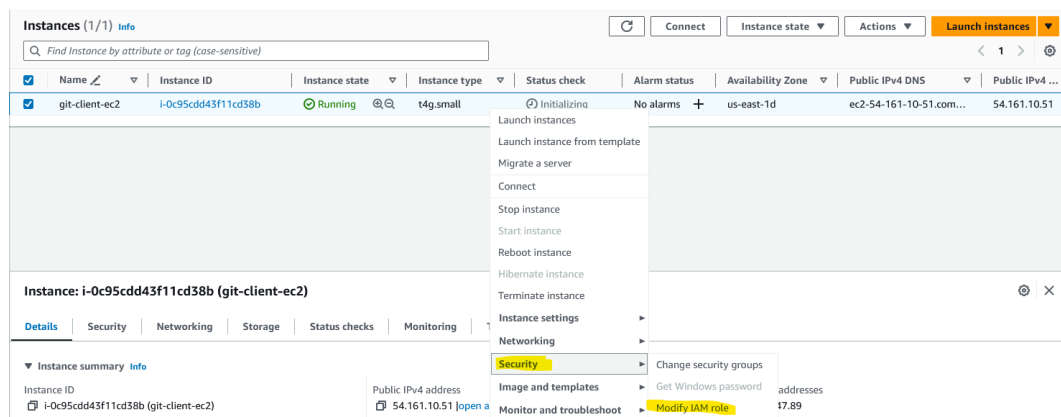


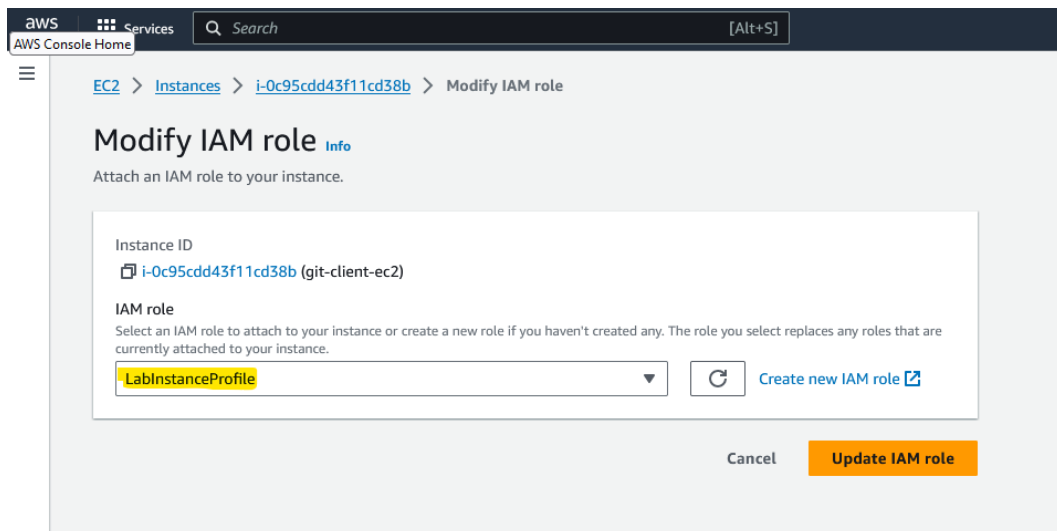
En un entorno real, el usuario o rol con potestad para aprobar las *pull request* podría realizar este proceso, para posteriormente fusionar ambas ramificaciones. En el caso de AWS Academy Learner Lab, esto no es posible, ya que no se nos permite crear nuevos usuarios o roles en nuestra cuenta.

ANEXO: ACCESO A UN REPOSITORIO DE AWS CODECOMMIT DESDE UN ENTORNO EN AWS

El principal problema que tiene el acceso a los repositorios de AWS CodeCommit con entornos externos es, precisamente, el hecho de que las credenciales obtenidas de los AWS Academy Learner Labs son temporales y, por tanto, cada vez que iniciamos el Lab se genera un nuevo conjunto de credenciales que es necesario configurar. Para evitar este inconveniente, se proponen dos opciones posibles:

- (A) Desplegar todo el entorno realizado en esta práctica en una instancia EC2 a la que se asignará el perfil de instancia *LabInstanceProfile* que está preconfigurado en AWS Academy Learner Lab y puede asignarse a las instancias EC2 para obtener los permisos indicados (entre otros, los de AWS CodeCommit) en el *LabRole*. Este perfil de instancia puede indicarse en el momento de la creación de la instancia o a posteriori, tal y como se indica en las imágenes siguientes:





(B) Utilizar el servicio de AWS Cloud9 para crear un entorno de desarrollo integrado administrado en la nube de AWS (ver práctica de AWS Cloud9). Los beneficios de utilizar esta opción son los siguientes:

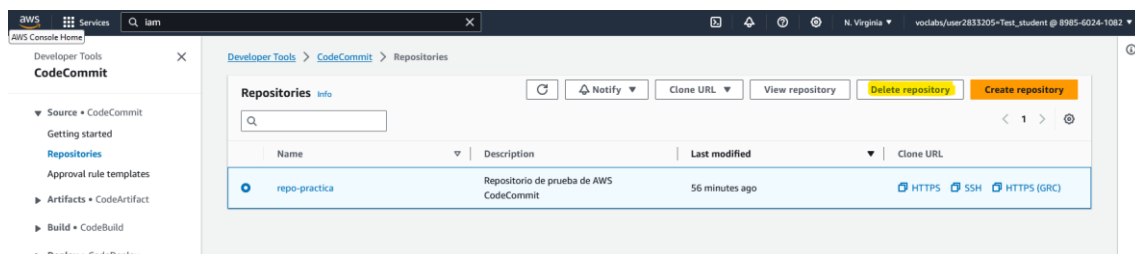
- No requiere la configuración de credenciales de AWS para operar sobre los recursos desplegados en AWS Academy Learner Lab.
- Lleva preinstalado y preconfigurado el AWS CLI
- Lleva preinstalada el sistema de control de versiones *Git*
- Lleva preinstalado la extensión *git-remote-codecommit* para operar con los repositorios de AWS CodeCommit utilizando URIs personalizadas
- Lleva preinstalado el AWS Toolkit para acceder visualmente a los recursos de AWS
- Lleva preinstalado un plugin para poder operar con repositorios *Git*

Se propone al alumno, a partir de los pasos de esta práctica, realizar las tareas necesarias para acceder a los repositorios de AWS CodeCommit utilizando tanto una instancia de Amazon EC2 como un IDE de AWS Cloud9.

Limpieza de la Práctica:

Para terminar esta práctica y liberar los recursos creados, evitando así el consumo de créditos de AWS Academy Learner Labs, simplemente debemos dar los siguientes pasos:

- Eliminar el repositorio de AWS CodeCommit. Para ello, desde la consola de AWS CodeCommit, seleccionamos el repositorio creado y presionamos el botón **Delete repository**



Recuerda también que, para hacer un uso responsable de los recursos en la nube, **el laboratorio de AWS Academy debe cerrarse** presionando el botón *End Lab* desde la plataforma.