Reporte de práctica de laboratorio

Control de versiones (CodeCommit)

03/28/2020

Maestría en Sistemas Computacioneales

*Aplicaciones y Servicios en la Nube*

Prof. Mtro. Rodolfo Luthe Ríos

Carlos Alberto Cordero Robles

ms197686@iteso.mx

# Introducción

Esta práctica está diseñada para crear y manejar un repositorio usando la herramienta de control de versiones de GIT.

Hay varios proveedores de servicio que permiten almacenar y administrar el repositorio en nube, en esta práctica se trabajara con dos github y codecommit. GitHub es un proveedor ya conocido con una interfaz web y opciones muy trabajadas por muchos desarrolladores. CodeCommit es particularmente de AWS y aunque su alcance no es tan amplio como el de GitHub su diferenciador es que puede interactuar junto con los otros servicios de AWS.

# Marco Teórico

Control de versiones

Son sistemas que ayudan al equipo de desarrollo a manejar los cambios en el código. Las herramientas de control de versiones llevan un registro de cada modificación en una base de datos y si por ejemplo se encuentra algún error en el funcionamiento del código se puede analizar dicho registro para saber a partir de que cambio se hizo presente dicho error y corregirlo sin afectar el resto del trabajo del equipo. Con estas herramientas el equipo puede paralelizar su trabajo ya que cada desarrollador puede realizar sus cambios en el conocido “Branch” y posterior mente integrar los cambios a un “branch” que será común para todos.

URL

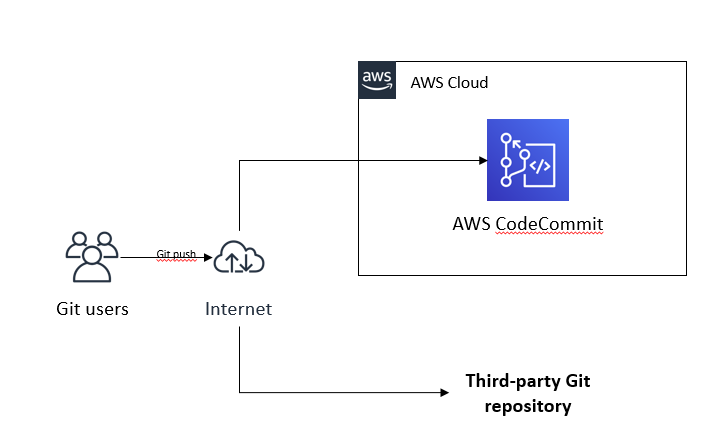
Son las siglas para Uniform Resource Locator. Son una serie de caracteres que sirven para indentificar un recurso en internet. También puede ser visto como una dirección a dicho recurso que va a estar disponible de manera virtual en internet.

GIT

Git es una herramienta de open source para el control de versiones diseñado para manejar toda clase de proyectos que involucren código. Resulta ser una herramienta bastante rápida y eficiente. Git es un sistema d eocntrol de versiones distribuido y a diferencia de los sistemas centralizados, los sistemas distribuidos tienen un manejo remoto y un manejo local de los archivos y branches y es necesario actualizar o sincronizar los cambios locales con los remotos.

Git fue desarrollado por Linus Torblads (desarrollador de Linux) y fue diseñado para el manejo de un gran número de archivos de código.

# Diagrama



# Desarrollo de la Práctica.

**Configurar repositorio local**

1. Instalar el cliente de git en una maquina

a. https://git-scm.com

2. Configurar git con su cuenta de iteso

a. git config --global user.name "John Doe"

b. git config --global user.email johndoe@example.com

3. Configurar el repositorio

a. Crear en Mis Documentos una carpeta llamada git

b. Desde consola cambiarse a ese directorio (o en Windows click derecho sobre la

carpeta y “Git Bash here”)

i. git init

4. Control de versiones

a. Crear en el directorio git un archivo: versiones.txt con el contenido: Versión 1

b. Añadir el archivo al seguimiento de git, desde la consola en la carpeta: git add .

i. Es importante el “.” para incluir todos los archivos en el tracking

c. Hacer un commit de los cambios: git commit –poner mensaje versión inicial

d. Editar el archivo y cambiar el contenido a: Versión 2

i. Hacer add y commit de los cambios:

1. git add .

2. git commit -m “segunda version”

e. Hacer lo mismo para la Versión 3

f. Consultar historial de vesiones: git log

En mi caso yo ya tengo GIT instalado así que pasare a configurar git.

En el explorador de Windows, botón derecho -> Git bash here



Figure 1Bash de GIT

Configuramos usuario y mail con las líneas de comando

git config –global user.name “<user\_name>”

git config –global user.email <email>

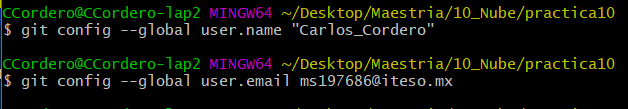


Figure 2Configuracion de usuario y mail

Ahora creamos una carpeta en donde estaremos trabajando el repositorio e iniciamos el repositorio ahí con el comando:

git init

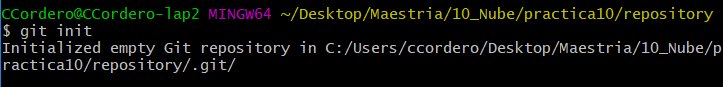


Figure 3Creacion de repositoro vacio de git

Se debe de crear un archivo .git que es donde git almacena todos los cambios de dicho repositorio pero es probable que no se vea. Para hacerlo visible es necesario cambiar las propiedades de folder para hacer visibles todos los archivos. Presionando botón derecho -> quitar el check de Hidden resolverá la situación.

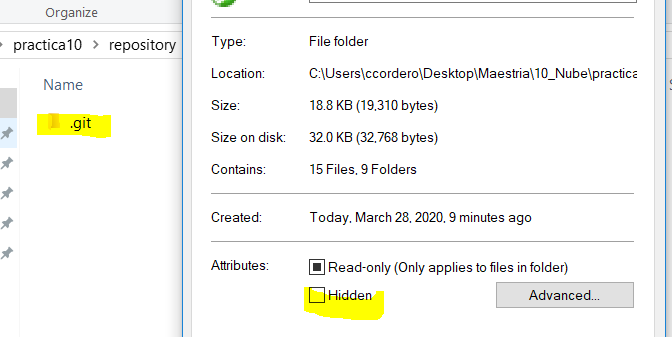


Figure 4folder .git

Ahora crearemos un archivo dentro de la carpeta del repositorio el archivo será dummy pero que nos servirá para versionarlo, se llamará versiones.txt y tendrá el texto Version 1.

Lo agregaremos el archivo al repositorio y le daremos commit al branch local con el mensaje versión inicial.

git add .

git commit -m "version inicial"

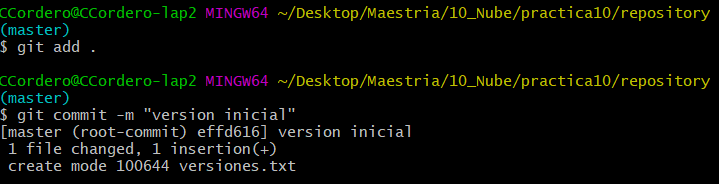


Figure 5archivo versiones.txt agregado y commit aplicado

Aplicaremos un cambio y le daremos commit otra vez, el cambio será ponerle Version 2 al texto y el mensaje del commit será seguda versión.

git add .

git commit -m "segunda version"

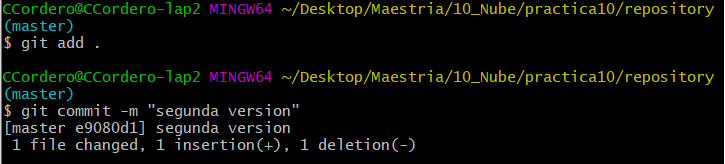


Figure 6archivo modificado y versionado a segunda versión

Repetimos nuevamente para ahora tener una tercera versión.

Usando una interfaz gráfica como tortoiseGiT podemos ver gráficamente los commits realizados.

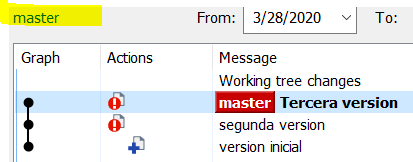


Figure 7Commits del archivo versiones

Ahora para consultar el historial de dicho branch que en este caso es master aplicamos git log. Deberán aparecer la información de usuario y mail cargados más los mensajes de commit.

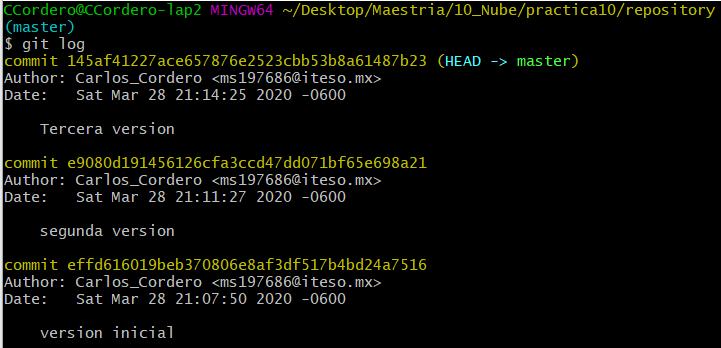


Figure 8log de cambios en branch master

**Configurar repositorio en GitHub**

1. Crear una cuenta con su correo de iteso en GitHub

2. Crear un repositorio en GitHub

3. Consultar el URL del repositorio

4. Añadir al repositorio local el repostorio de GitHub, desde la carpeta de git:

a. git remote add Hub <URL>

b. git push Hub master

5. Refrescar la pantalla del repositorio en GitHub ¿Qué cambio observan?

En mi caso yo ya tengo una cuenta de github, por lo tanto, pasare a crear un repositorio

En la página de github y dentro de nuestra cuenta le damos en New.



Figure 9Creación e nuevo repositorio

Configuramos el repositorio.

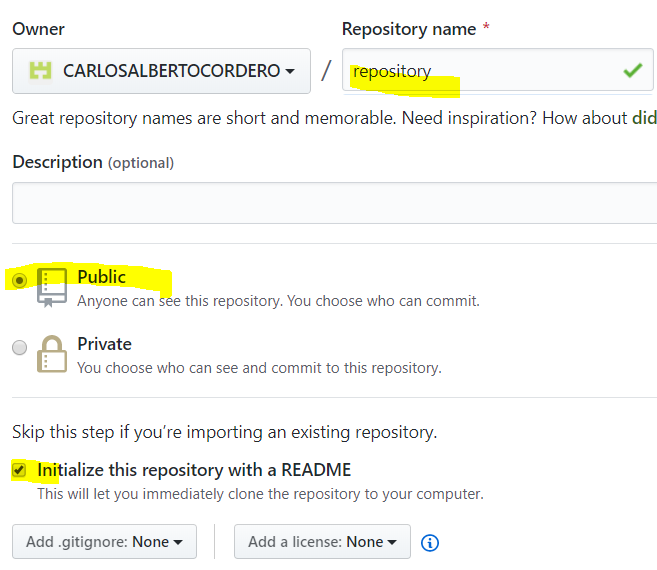


Figure 10Configuración de repositorio en github

Para consultar el URL del repositorio lo que tenemos que hacer es presinar el botón de Clone or download y desplegara el URL.

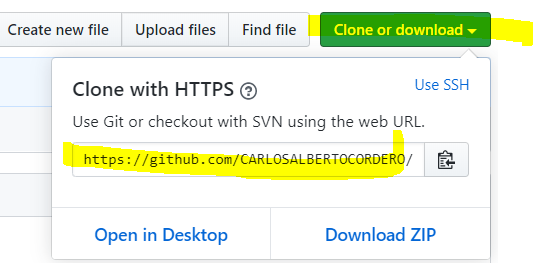


Figure 11URL de repositorio

Ahora vincularemos nuestro repositorio local ya creado con el repositorio en github y luego subiremos los cambios del repositorio local al remoto.

Nota: Como se creó el repositorio remote con un Readme en mi caso es posible que sea necesario primero hacer un pull para tener los cambios sincronizados

git pull --progress -v --no-rebase --allow-unrelated-histories "GitHub" master

git remote add Hub <URL>

git push Hub master

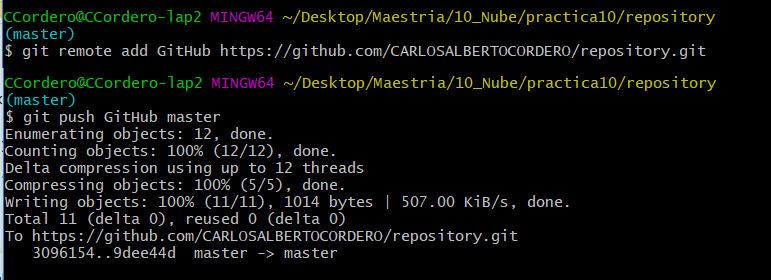
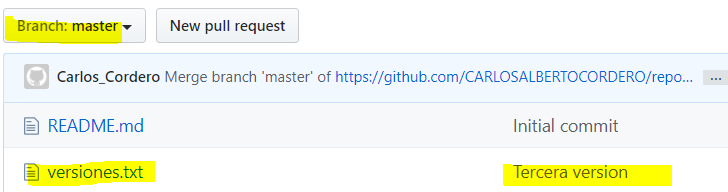


Figure 12Vinculacion de repositorio remoto y local y subida de archivos

Después de subir los cambios veremos en la interfaz de github que para el branch master aparecerá el archivo versiones.txt y el mensaje de commit asignado.



**Configurar repositorio en AWS CodeCommit**

1. Crear credemnnciales para CodeCommit

a. Desde la consola de IAM y seleccionar el usuario deseado

b. En Security Credentials generar para: HTTPS para CodeCommit

2. Desde la consola de CodeCommit crear un repositorio

3. Consultar el URL del repositorio

4. Añadir al repositorio local el repostorio de CodeCommit, desde la carpeta de git:

a. git remote add AWS <URL>

b. git push AWS master

5. Consultar el repositorio de CodeCommit

Primero voy a empezar por crear un usuario, así como se hizo en la práctica 1.

Services -> IAM -> Add user

Luego seleccionamos el usuario y nos movemos a la pestaña de Security credentials.

En esa pestaña es necesario generar las credenciales para HTTPS Git credentials for AWS CodeCommit, Download credentials.

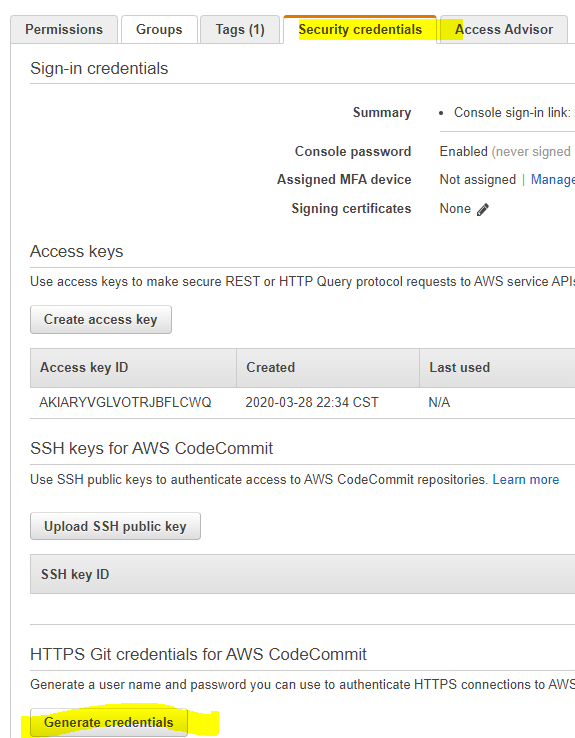


Figure 13Generación de credenciales para GIT

Ahora vamos al servicio de CodeCommit para crear un repositorio remoto y obtener su URL.

Services -> CodeCommit -> Create repository, le damos nombre y presionamos créate.

Así como en github le damos en clone URL. El URL quedara copiado en el clipboard.

Ahora usaremos los mismos comandos para vincular y subir los cambios a AWS.

git remote add AWS10 <URL>

git push Hub master

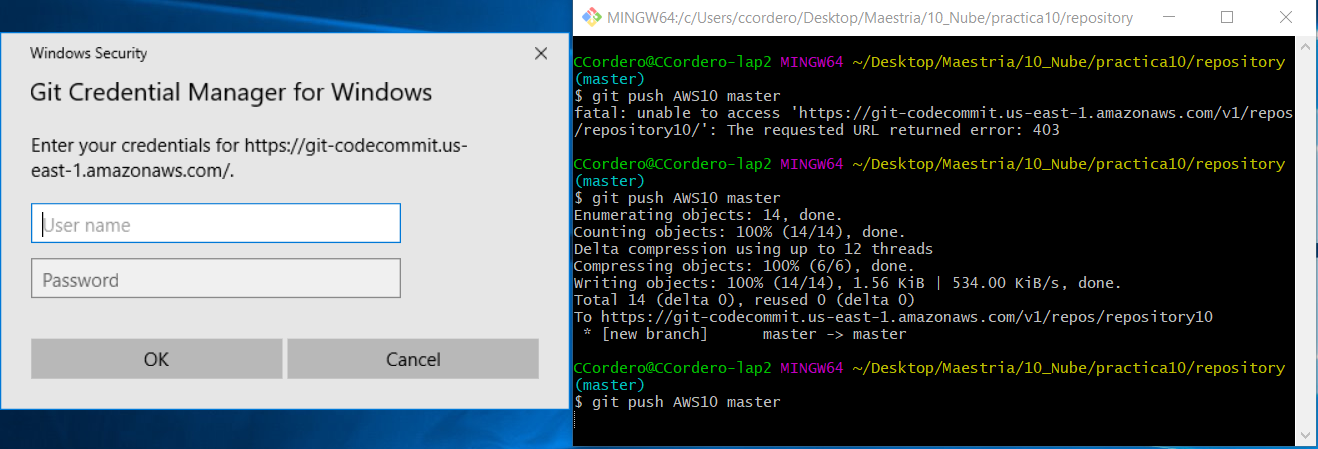


Figure 14Subiendo archivos a CodeCommit

Nos pedirá las credenciales que generamos para CodeCommit, después de ponerlas empezará la descarga.

Veremos en la página de CodeCommit que aparecen los archivos.

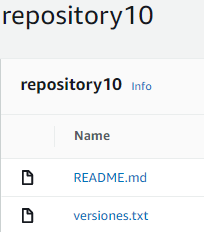


Figure 15CodeCommit con archivos subidos

**Controlar las versiones del entregable de la práctica 10**

1. Crear el documento del reporte, y añadirlo para tracking

2. Hacer add y commit al terminar cada sección del reporte -Marco, Diagrama, etc- con el

comentario correspondiente

3. Mostrar log de los cambios al documento

4. Subir PDF del reporte a Moodle, compartir el URL del documento en CodeCommit

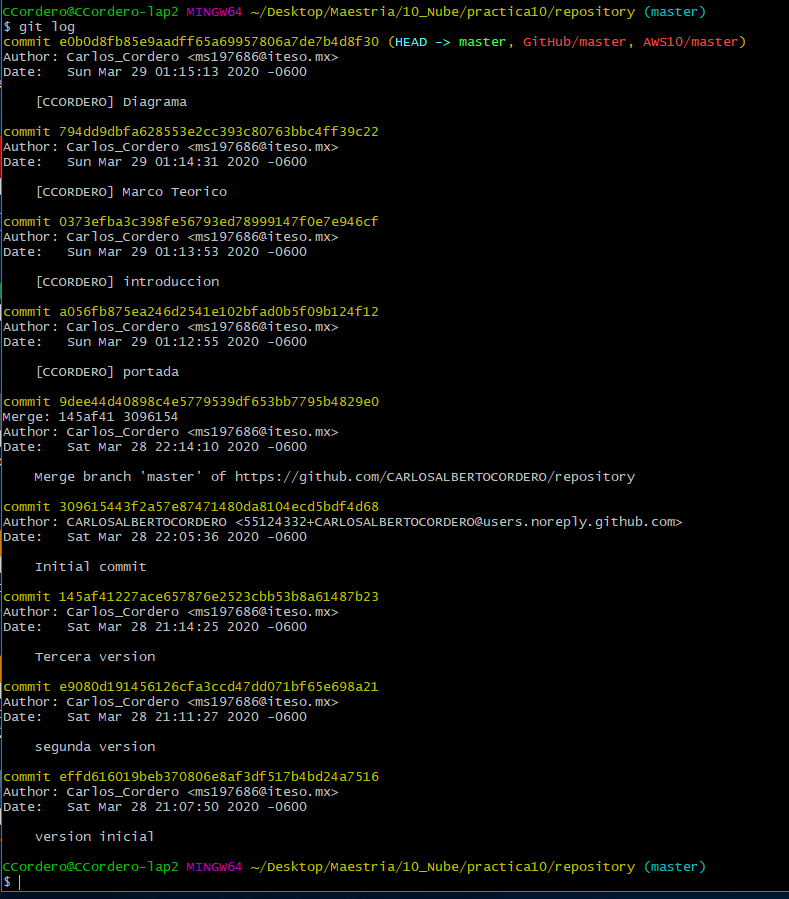


Figure Log de commits a práctica

URL de práctica: <https://console.aws.amazon.com/codesuite/codecommit/repositories/repository10/browse/refs/heads/master/--/L010_CORDERO_ROBLES_CARLOS_ALBERTO_ms197686.docx?region=us-east-1>

# Problemas y Soluciones

Como estamos creando de manera separada en repositorio local y el remoto es necesario hacer un merge de los historiales, hay dos modos o a la hora de crearel remoto lo creamos totalmete vacio o la otra es aplicar el comando git pull --progress -v --no-rebase --allow-unrelated-histories "GitHub" master para hacer el merge de historiales.

Para esta práctica es mejor usar la cuenta personal de AWS y no la educate para evitar problemas de credenciales de usuarios.

Si te equivocas a la hora de pasar las credenciales en la sección de CodeCommit es necesario configurar git para que pregunte otra vez como esta en la referencia [5].

# Experimentos y Resultados.

Lo primero que se hace en la práctica es hacer un repositorio local y aplicar algunos commits.

Posteriormente se crea un repositorio remoto en github y se suben los cambios. Se repiten los pasos paro ahora con CodeCommit que es el servicio que proporciona AWS para repositorios de control de versiones.

Por último, subimos los cambios de cada sección de la práctica.

# Costo

Al parecer esta práctica es gratis ya que ni github ni AWS me cobran por los archivos que subí.

Para que github cobre tendría que subir más de 1GB de información.

Para AWS los primeros 5 usuarios tienen el beneficio de repositorios ilimitados, 50GB y 10000 solicitudes de git al mes.

# Conclusiones

\* Las herramientas de control de versiones facilitan al equipo de desarrollo para trabajar en paralelo y también llevar un historial de los cambios.

\* Existen más de un proveedor de servicios para almacenamiento de repositorio.

\* GIT es un Sistema de control de versiones distribuido por lo cual se pueden manejar branches locales y remotos.

\* Para poder subir los cambios de los repositorios locales a los remotos primero hay que vincularlo y luego subirlos con push.

# Bibliografia

[1] «What is version control». [En línea]. Disponible en:

<https://www.atlassian.com/git/tutorials/what-is-version-control> [Accedido: 29-Mar-2020].

[2] «Definicion de URL». [En línea]. Disponible en:

<https://definicion.de/url/> Accedido: 29-Mar-2020].

[3] «git--distributed-is-the-new-centralized». [En línea]. Disponible en:

<https://git-scm.com/> Accedido: 29-Mar-2020].

[4] «Qué Es Git». [En línea]. Disponible en:

<https://codigofacilito.com/articulos/que-es-git> Accedido: 29-Mar-2020].

[5] «Git push The requested URL returned error: 403». [En línea]. Disponible en:

<https://es.stackoverflow.com/questions/201496/git-push-the-requested-url-returned-error-403> Accedido: 29-Mar-2020].