Reporte de práctica de laboratorio

Practica 10: Control de versiones (CodeCommit)

30 de marzo de 2019

Maestría en Sistemas Computacionales

*Aplicaciones y Servicios en la Nube*

Prof. Mtro. Rodolfo Luthe Ríos

Aldo Orozco Gómez Serrano

[ms723300](mailto:mi11729@iteso.mx)@iteso.mx

# Introducción

En esta práctica manejaremos la plataforma de control de versiones de AWS CodeCommit y haremos una comparación con respecto de Github y CodeCommit. En el fondo, utilizaremos Git, una herramienta de control de versiones, para gestionar los cambios de este reporte.

Configuraremos Git con nuestra cuenta de ITESO con parámetros globales. Despues crearemos una carpeta llamada git y dentro de la misma crearemos un repositorio de git usando *git init*. Añadiremos un archivo con cierto contenido y lo modificaremos para observar los cambios que se generan.

# Marco Teórico

El uso de control de versiones es ampliamente utilizado en la actualidad y se considera una buena práctica para el desarrollo de software. Aparte de permitir la colaboración ágil entre colaboradores, el control de versiones permite colaboración en tiempo real usando la mínima cantidad de recursos [1].

En las últimas casi tres décadas desde su invención, Git y Github han mejorado radicalmente la capacidad de programadores de colaborar en complejos proyectos de software en tiempo real con decenas de otras personas alrededor del mundo sin preocuparse del caos que esto generaba.

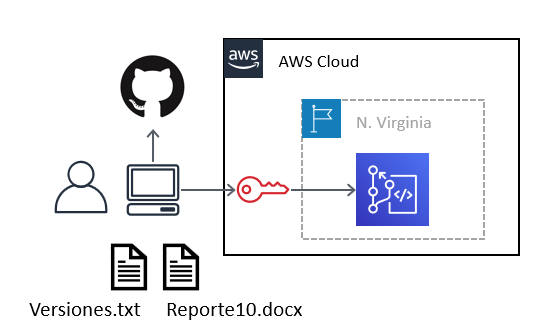
En años mas recientes se ha descubierto que estas herramientas no solo están limitadas al control de software, sino que también resultan sumamente útiles cuando de documentos se habla. Esto debido a que facilita la colaboración y el compartir cambios de un documento a otro colaborador o investigador, sin necesidad de tener un historial aparte con los cambios generados a los documentos.

GitHub en particular es una plataforma que permite visualizar los distintos cambios en una interfaz grafica web mucho mas amigable e intuitiva que la línea de comandos que, aunque mas eficiente, complica en ocasiones la revisión de ciertos cambios.

AWS CodeCommit es un sistema de control de versiones automatizado basado en Git que permite tener múltiples repositorios de código. Facilita la colaboración en un sistema altamente escalable y elimina la necesidad de operar el sistema de control de versiones, así como la escalabilidad de la infraestructura [2].

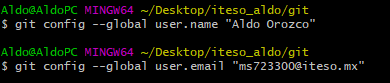
Entre los beneficios que ofrece se encuentran la alta disponibilidad, debido a que es administrado por Amazon, la alta disponibilidad y seguridad ya que se puede controlar el acceso utilizando los servicios ya conocidos por Amazon como IAM para determinar accesos a usuarios específicos.

# Diagrama

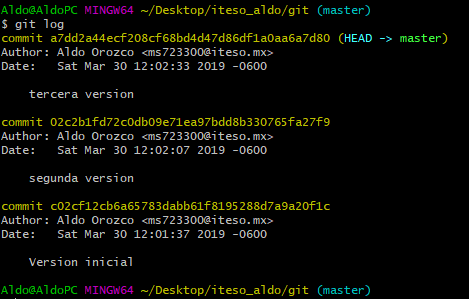


# Desarrollo de la Práctica.

Primero nos descargaremos Git en la computadora, siguiendo las instrucciones de la pagina oficial. Una vez instalado, abrimos el servicio de git bash y configuramos el nombre y el correo electrónico con configuración global.

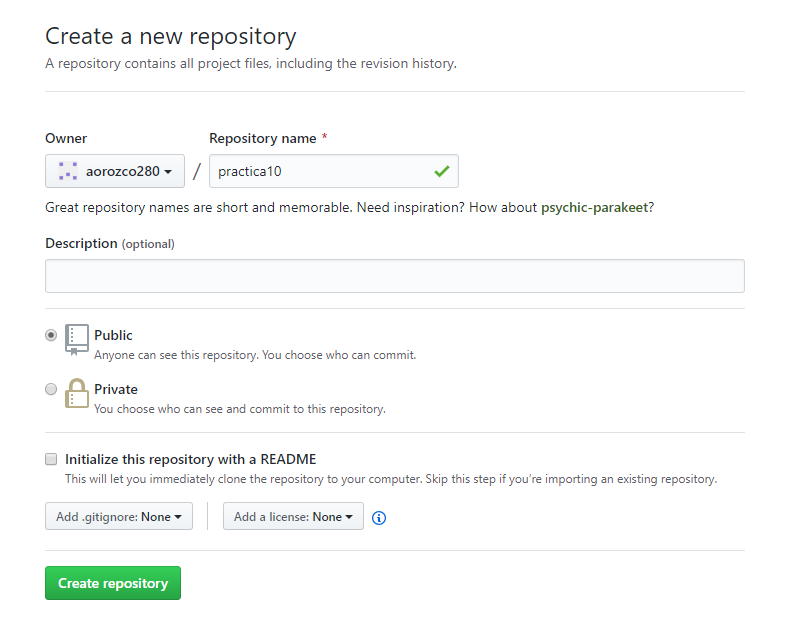


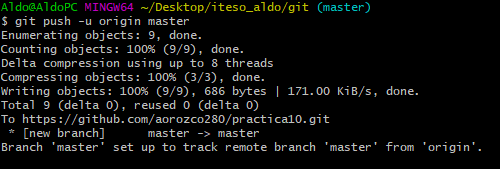
Ahora creamos un nuevo repositorio con nombre *git* y creamos un archivo llamado versiones.txt que contenga el texto *Version 1.* En la consola de git, añadimos ese archivo y creamos un primer commit. Ahora cambiamos el texto a *Version 2* y creamos un segundo commit indicando el cambio. Repetimos el procedimiento para la *Version 3*.



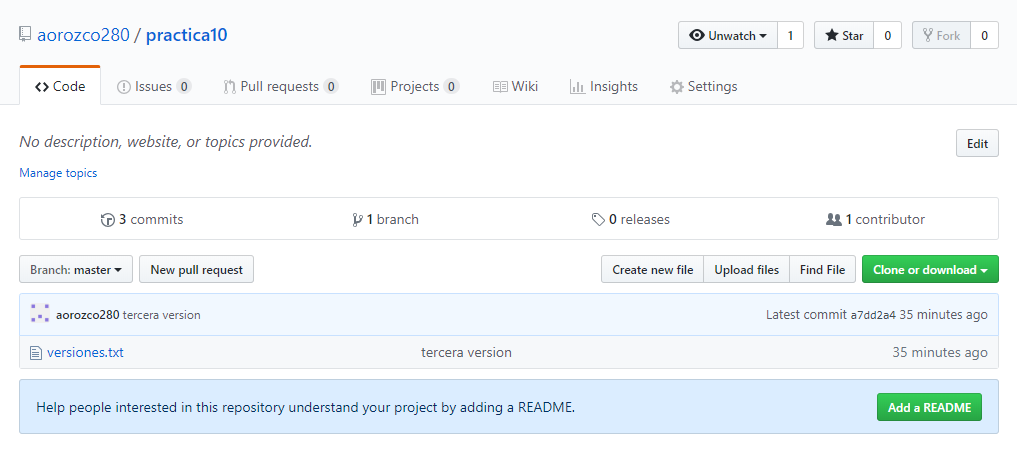
Ahora procedemos a crear una cuenta nueva en GitHub, siguiendo las instrucciones en la pagina principal. Crearemos un nuevo repositorio publico con nombre *Practica 10* y configuraremos el repositorio local para que apunte al recién creado repositorio, ejecutando los siguientes comandos:

* git remote add origin <https://github.com/aorozco280/practica10.git>
* git push -u origin master





Observamos que el archivo versiones.txt aparece en el repositorio en Github.

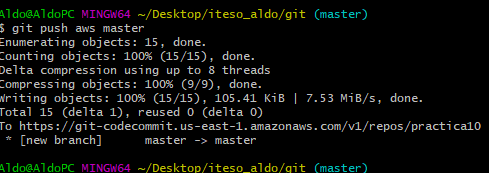


Ahora nos dirigimos a la consola de AWS y nos loguemos como programador administrador. Nos dirigimos al servicio de IAM y seleccionamos este usuario; en la opción de Security Credentials se crearán llaves para HTTPS CodeCommit. Esto nos descargara un CSV con las credenciales correspondientes.

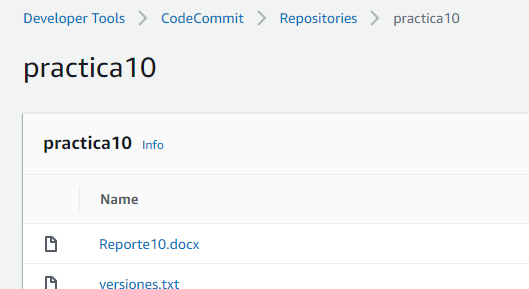
Ahora nos dirigimos al servicio de CodeCommit. Crearemos un nuevo repositorio llamado *Practica 10* y guardaremos la URL.

En la consola, añadimos la URL como un remoto llamado aws. Para ello ejecutamos los siguientes comandos:

* git remote add aws <https://git-codecommit.us-east-1.amazonaws.com/v1/repos/practica10>
* git push aws master



Ahora observamos el repositorio en CodeCommit y contiene los archivos de versiones.txt y el reporte de esta práctica.



Finalmente creamos el tracking de este reporte en el repositorio de Git. Por cada sección iremos añadiendo un nuevo commit y documentándolo en los comentarios.

# Problemas y Soluciones

Se presentaron dos problemas. El primero fue que no pude dar de alta una cuenta con mi usuario de ITESO. Al parecer, existe algún chequeo en la interfaz web que no permite que la arroba sea antecedida por un número, puesto que si añadía una letra entre el ultimo digito de mi expediente y la arroba si lo permitía. Terminé utilizando mi correo personal y no tuve que hacer ninguna modificación a mi repositorio.

El segundo problema fue que configure Git, al momento de dar de alta el remoto de AWS, con el nombre del usuario de AWS y sus credenciales, pero me dio un error y ya no logre sacarlo de ese error. Encontré una solución siguiendo este link:

* <https://stackoverflow.com/questions/15381198/remove-credentials-from-git>

Al final me percate que las credenciales son las que había descargado en CSV en la creación de credenciales HTTPS para CodeCommit y altere las ya existentes para ajustarse a estos valores y funciono.

# Experimentos y Resultados.

# Costo

# Conclusiones

# Bibliografía

[1] – H. Daniel; N. Chong, “Creative Coding with Social Version-Control: Enhancing Collaboration, Iteration & Transparency”, August 2018

[2] – “Amazon CodeCommit”, <https://aws.amazon.com/codecommit/>