Reporte de práctica de laboratorio

**Práctica 10: Control de versiones (CodeCommit)**

Fecha 04/Abril/2021

Maestría en Sistemas Computacioneales

*Aplicaciones y Servicios en la Nube*

Prof. Mtro. Rodolfo Luthe Ríos

Nombre\_alumno Elva Alejandra Rodríguez Rodríguez

[correo\_electrónico](mailto:mi11729@iteso.mx) elva.rodriguez@iteso.mx

# Introducción

El objetivo de la práctica es aprender a utilizar un controlador de versiones de archivos con el objetivo de observar las ventajas. En este caso se seleccionó Git como la herramienta para trabajar con esto además de conectarlas con diferentes plataformas como AWS utilizando CodeCommit y GitHub para poder visualizar las versiones y los cambios en cada una.

Parte del aprendizaje de la práctica se basa en aprender el uso del versionador antes mencionado en caso de no haberlo utilizado en el pasado. Y el control de estas.

# Marco Teórico

**Git** [1]

Hoy en día, Git es, con diferencia, el sistema de control de versiones moderno más utilizado del mundo. Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, el famoso creador del kernel del sistema operativo Linux, en 2005. Un asombroso número de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto. Los desarrolladores que han trabajado con Git cuentan con una buena representación en la base de talentos disponibles para el desarrollo de software, y este sistema funciona a la perfección en una amplia variedad de sistemas operativos e IDE (entornos de desarrollo integrados).

Git, que presenta una arquitectura distribuida, es un ejemplo de DVCS (sistema de control de versiones distribuido, por sus siglas en inglés). En lugar de tener un único espacio para todo el historial de versiones del software, como sucede de manera habitual en los sistemas de control de versiones antaño populares, como CVS o Subversion (también conocido como SVN), en Git, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios.

**Controlador de versiones** [2]

¿Qué es un control de versiones, y por qué debería importarte? Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante. Aunque en los ejemplos de este libro usarás archivos de código fuente como aquellos cuya versión está siendo controlada, en realidad puedes hacer lo mismo con casi cualquier tipo de archivo que encuentres en una computadora.

Si eres diseñador gráfico o de web y quieres mantener cada versión de una imagen o diseño (es algo que sin duda vas a querer), usar un sistema de control de versiones (VCS por sus siglas en inglés) es una decisión muy acertada. Dicho sistema te permite regresar a versiones anteriores de tus archivos, regresar a una versión anterior del proyecto completo, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que pueda estar causando problemas, ver quién introdujo un problema y cuándo, y mucho más. Usar un VCS también significa generalmente que, si arruinas o pierdes archivos, será posible recuperarlos fácilmente. Adicionalmente, obtendrás todos estos beneficios a un costo muy bajo.

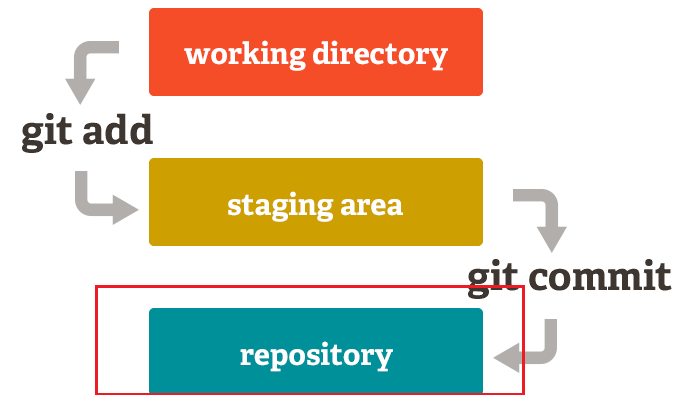
**GitHub** [3]

GitHub es una compañía sin fines de lucro que ofrece un servicio de hosting de repositorios almacenados en la nube. Esencialmente, hace que sea más fácil para individuos y equipos usar Git como la versión de control y colaboración.

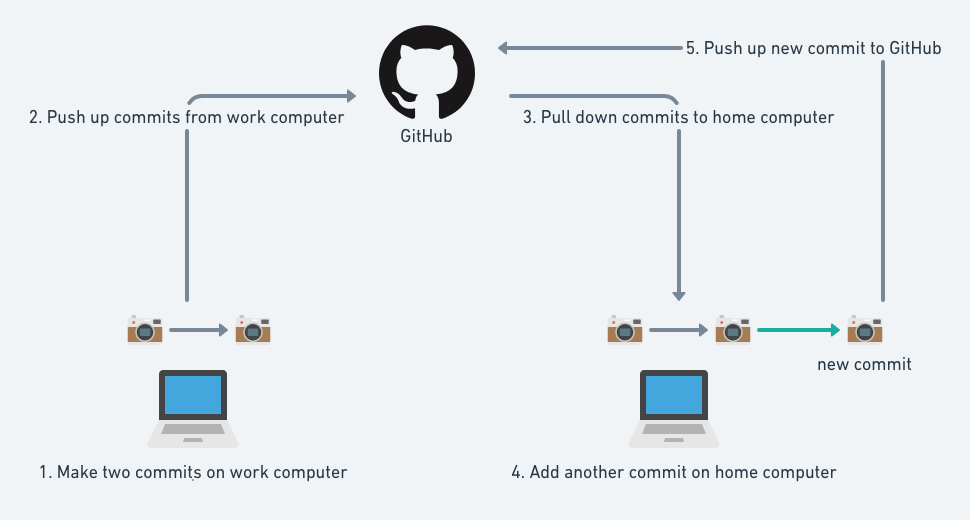
La interfaz de GitHub es bastante fácil de usar para el desarrollador novato que quiera aprovechar las ventajas del Git. Sin GitHub, usar un Git generalmente requiere de un poco más de conocimientos de tecnología y uso de una línea de comando.

# Diagrama

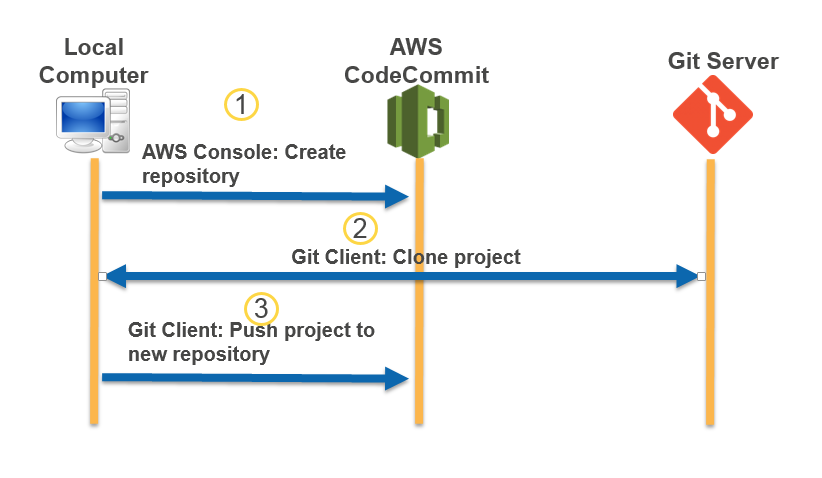
**Workflow to add and versioning a file**



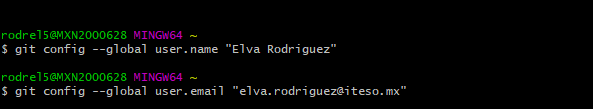
**Git to GitHub**



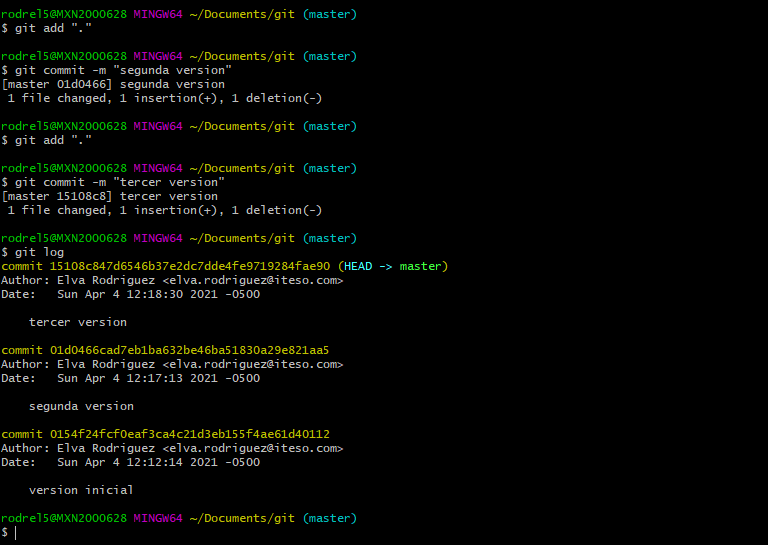
**AWS CodeCommit**

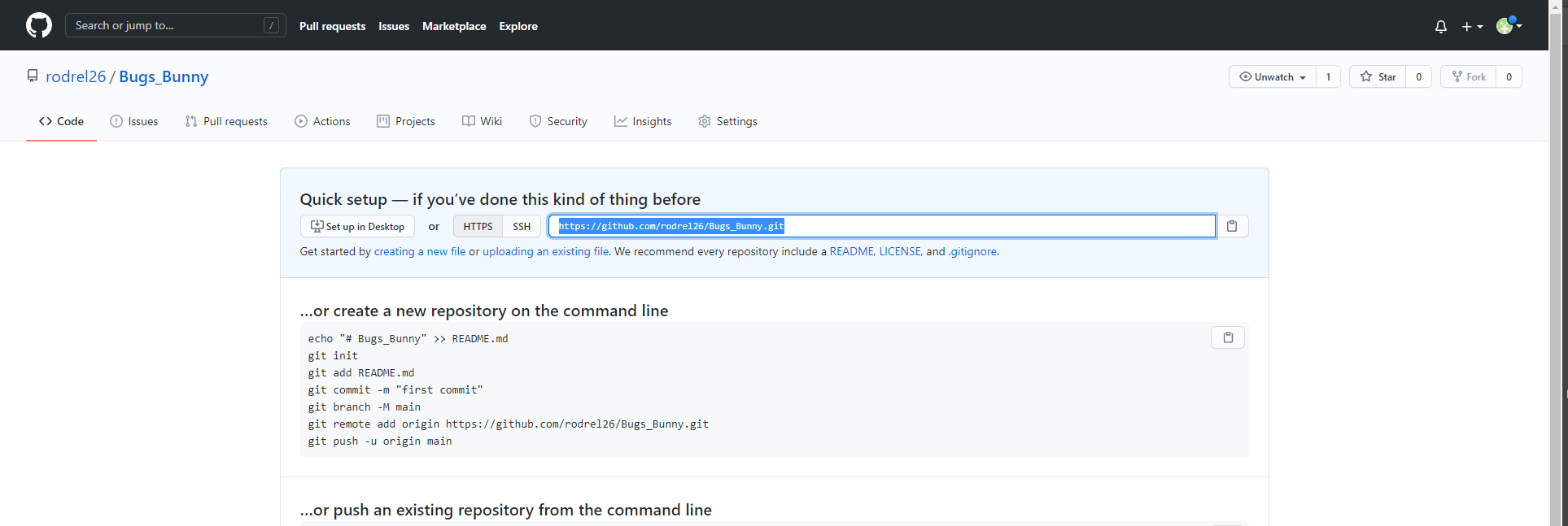


# Desarrollo de la Práctica.

**1. Configurar git con su cuenta de iteso**

**2. Configurar el repositorio**

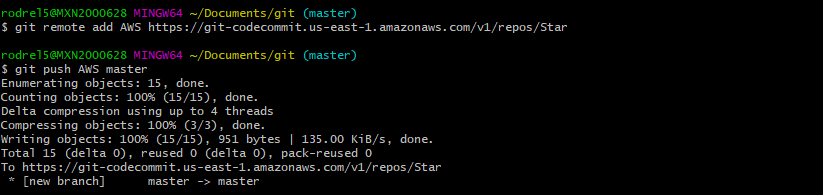
**3. Control de versiones**

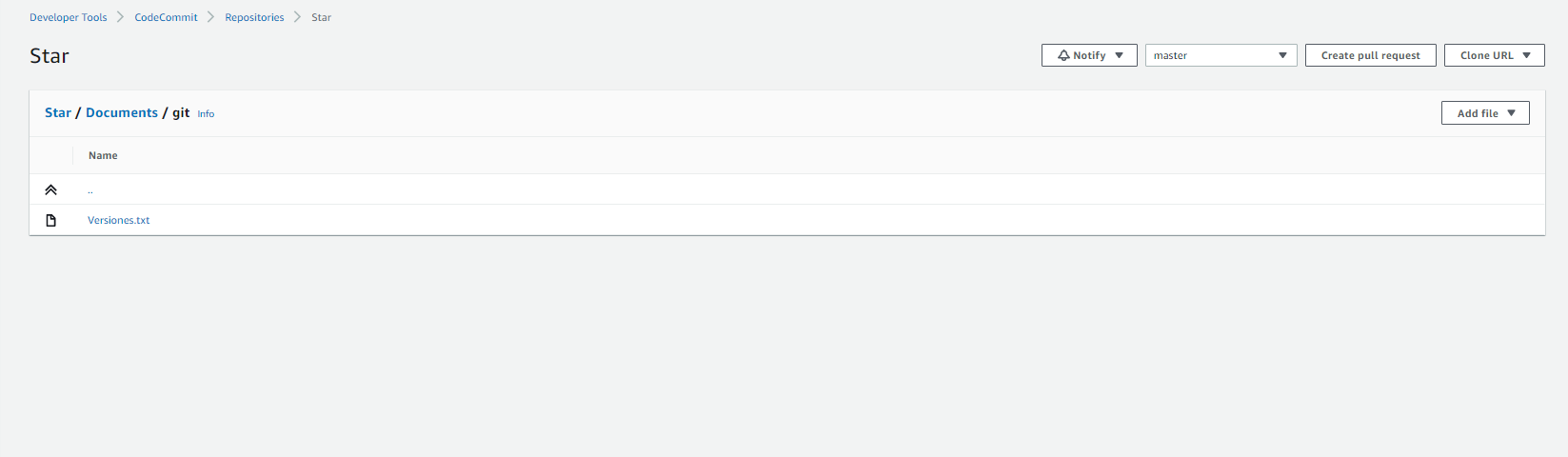
**4. Configurar repositorio en GitHub**

# 

# 

**5. Configurar repositorio en AWS CodeCommit**





# Problemas y Soluciones

El problema que se presento fue al inicio tratando de entender como funcionaba la función de “add” y “commit” en git.

La manera en la que solucioné el problema fue utilizando la información de los comandos básicos que encontré en el enlace adjunto abajo:

<https://rubygarage.org/blog/most-basic-git-commands-with-examples>

Fue de bastante utilidad para poder trabajar con lo que se pedía en la práctica.

# Experimentos y Resultados.

Los experimentos realizados fueron a través de la consola de git la que permitió utilizar el versiona miento. Además del uso y experimento de repositorios en GitHub y AWS.

Los resultados obtenidos de los experimentos realizados en la práctica fueron mantener el historial de los archivos que con los que se trabajó y visualizarlos en las consolas de AWS utilizando CodeCommit y GitHub.

# Costo

**Code Commit Costo**

# Cualquier persona que disponga de una cuenta de AWS puede comenzar a usar AWS CodeCommit de forma gratuita. Su cuenta incluye 5 usuarios activos por mes sin cargo (con ciertos límites). Deberá pagar 1,00 USD por usuario activo adicional al mes. No es necesario pagar tarifas iniciales ni asumir compromisos.

# Esta oferta de la capa gratuita de AWS para AWS CodeCommit se encuentra disponible para clientes de AWS nuevos y existentes de forma indefinida, es decir, no vence al finalizar el período estándar de 12 meses de la capa gratuita.

# **Primeros 5 usuarios activos**\*

# 0,00 USD

# Recibe:

# Repositorios ilimitados

# 50 GB/mes de almacenamiento

# 10 000 solicitudes Git al mes

# **A partir del sexto usuario activo\***

# 1,00 USD al mes

# Recibe:

# Repositorios ilimitados

# 10 GB/mes de almacenamiento por usuario activo

# 2 000 solicitudes Git/mes por usuario activo

**Plan de GitHub utilizado en la practica**

Gratis:

Ilimitados repositorios públicos / privados

Colaboradores ilimitados

2,000 acciones minutos / mesgratis para repositorios públicos

500 MB de almacenamiento de paquetes GitHubGratis para repositorios públicos

Soporte comunitario

# Conclusiones

Las conclusiones deberán ser un trabajo de reflexión que nos presente el conocimiento adquirido desde los experimentos, los resultados y el marco teórico presentado. No son conclusiones válidas: “¡aprendí mucho!”, “me gustó mucho la práctica”, “todo funcionó correctamente”. (1 o 2 párrafos).

# Bibliografia

En formato de la IEEE