Reporte de práctica de laboratorio

*Control de versiones (CodeCommit)*

*04 de Abril del 2021*

Maestría en Sistemas Computacioneales

*Aplicaciones y Servicios en la Nube*

Prof. Mtro. Rodolfo Luthe Ríos

Priscila Campos Velázquez

[ms727309@iteso.mx](mailto:mi11729@iteso.mx)

# Introducción

Desde mi perspectiva, el manejo de control de versiones distribuido, una vez conocido y dominado se convierte en una necesidad no sólo para seguir los cambios de código, sino para el manejo de archivos. Existen documentos en Excel a los que diariamente mi equipo de trabajo y yo hacemos actualizaciones. Cualquier cambio es sumamente importante, ya que existe una consolidación de toda la información en donde se proyectan métricos, y un error puede llevar a conclusiones que pudieran llevar a un resultado en gastos económicos, recursos y esfuerzos innecesarios.

Los dos manejadores de versiones utilizados en la practica son GitHub y CodeCommit. Entre las diferencias más relevantes está el hecho de que en GitHub la información que cargues vive en los servidores compartidos de GitHub, y la configuración del almacenamiento puede ser pública o privada. CodeCommit siempre considera la información que cargues como privada, y requiere de configuraciones de usuarios y control de acceso. Además de esto siempre existe la posibilidad de pedir el almacenamiento de la información en un servidor privado.

Los objetivos de esta practica:

* Utilizar un cliente de control de versiones
* Configurar servicios de control de versions centralizados
* Controlar las versiones de un documento

Las actividades principales de la practica fueron:

* Configurar un repositorio local
* Crear un repositorio en GitHub
* Crear un reporsitorio en CodeCommit
* Controlar las versiones de un documento

# Marco Teórico

Los conceptos más destacados para la buena comprensión de la practica fueron:

* ***DVCS (Distributed Version Control System):***El DVCS es un tipo de sistema de control de versiones en donde se encuentra la base de la información de un documento o código encluyendo todo su historial, y este es espejeado en todos los centros de trabajo de los que tienen acceso para editar dicha información. [[1](#_Bibliografia_1)]
* ***GitHub:*** Es un sistema de control de versiones distribuido donde los clientes no sólo pueden ver la última foto de los archivos, sino que también pueden ver el repositorio de todas sus versiones. De esta forma, si algún servidor falla y hubo sistemas colaborando a través de dicho servidor, cualquier repositorio del cliente puede ser copiado de nuevo al servidor para restaurar la información. Cada clon es un respaldo total de la información. [[2](#_Bibliografia_1)]
* ***AWS CodeCommit:*** es un servicio completamente administrado de control de código fuente que aloja repositorios basados en Git seguros. Simplifica la colaboración en el código por parte de los equipos, en un ecosistema seguro y con alta escalabilidad. Con CodeCommit no necesita utilizar su propio sistema de control de código fuente ni preocuparse por el escalado de la infraestructura de dicho sistema. CodeCommit, que funciona perfectamente con las herramientas de Git existentes, se puede utilizar para almacenar de forma segura cualquier elemento, ya sea código fuente o binario. [[3](#_Bibliografia_1)]

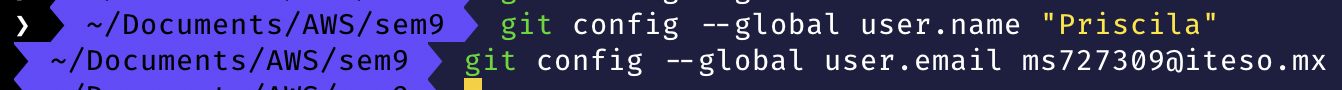
# Diagrama

# Graphical user interface, application, Teams Description automatically generated

# Desarrollo de la Práctica.

**Configurar repositorio local**

* 1. Instalar el cliente de git en una máquina.
     1. https://git-scm.com
  2. Configurar git con su cuenta de iteso.
     1. git config --global user.name "John Doe".
     2. git config --global user.email johndoe@example.com.

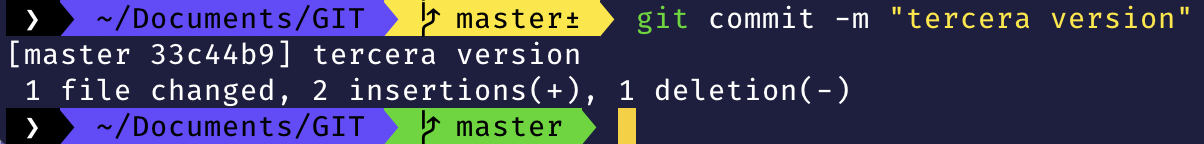


* 1. Configurar el repositorio.
     1. Crear en Mis Documentos una carpeta llamada git.
     2. Desde consola cambiarse a ese directorio (o en Windows click derecho sobre la carpeta y “Git Bash here”), git init.
  2. Control de versiones:
     1. Crear en el directorio git un archivo: versiones.txt con el contenido: Versión 1

A picture containing text

Description automatically generated

* + 1. Añadir el archivo al seguimiento de git, desde la consola en la carpeta: git add .
       1. Es importante el “.” para incluir todos los archivos en el tracking
    2. Hacer un commit de los cambios: git commit –poner mensaje versión inicial.Text

       Description automatically generated with medium confidence
    3. Editar el archivo y cambiar el contenido a: Versión 2.
       1. Hacer add y commit de los cambios:
          1. git add .
          2. git commit -m “segunda version”
    4. Hacer lo mismo para la Versión 3
    5. Consultar historial de vesiones: git log. Text

       Description automatically generated

**Configurar repositorio en GitHub**

* 1. Crear una cuenta con su correo de iteso en GitHub Graphical user interface, text, application, email

     Description automatically generated
  2. Crear un repositorio en GitHub
  3. Consultar el URL del repositorio Graphical user interface, text, application, email

     Description automatically generated
  4. Añadir al repositorio local el repositorio de GitHub, desde la carpeta de git:
     1. git remote add Hub <URL>
     2. git push Hub master

Graphical user interface, text

Description automatically generated

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

Graphical user interface, application, Teams

Description automatically generated

**Configurar repositorio en AWS CodeCommit**

* 1. Crear credemnnciales para CodeCommit.
     1. Desde la consola de IAM y seleccionar el usuario deseado.
     2. En Security Credentials generar para: HTTPS para CodeCommit.

Application

Description automatically generated with medium confidence

* 1. Desde la consola de CodeCommit crear un repositorio.Graphical user interface, text, application

     Description automatically generated
  2. Consultar el URL del repositorio.

Graphical user interface, text, application

Description automatically generated

* 1. Añadir al repositorio local el repostorio de CodeCommit, desde la carpeta de git:
     1. git remote add AWS <URL>.
     2. git push AWS master.

Text

Description automatically generated

* 1. Consultar el repositorio de CodeCommit.

Graphical user interface, text, application, Word

Description automatically generated

**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 Teorico, 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.

# Problemas y Soluciones

Cuando intenté hacer el push AWS master, marcaba error 403. Se solucionó agregando al usuario el permiso de AWSCodeCommitFullAccess.



Graphical user interface, text, application, email

Description automatically generated

# Costo

El costo de este servicio es nulo cuando administra hasta cinco usuarios.

Graphical user interface, text, application

Description automatically generated

# Conclusiones

La necesidad de un manejador de versiones distribuido siempre ha existido, aunque no sea una herramienta conocida por todos, es utilizada por la mayoría detrás de las herramientas para trabajar en conjunto en un sólo archivo o código, como Google Drive y OneDrive. Sin embargo, poder ahora mantener la base de la información con el histórico de cambios de manera completamente privada, a través de AWS, es un gran impulso para las empresas que desean mantener segura su información. Además de la ventaja de poder contar con completo control del repositorio, este servicio en AWS es más barato que GitHub y con muchas ventajas, como los repositorios ilimitados, y miles de cambios solicitados al mes por cada usuario por la módica cantidad de 1 USD al mes.

# Bibliografia

[1] <https://www.perforce.com/blog/vcs/what-dvcs-anyway>

[2]<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

[3] <https://aws.amazon.com/es/codecommit/>