Reporte de práctica de laboratorio

**Control de versiones (CodeCommit)**

Fecha: 07 de abril de 2021

Maestría en Sistemas Computacionales

*Aplicaciones y Servicios en la Nube*

Prof. Mtro. Rodolfo Luthe Ríos

Juan Carlos Morales Ruiz

[carlos.morales@iteso.mx](mailto:mi11729@iteso.mx)

# Introducción

**Objetivos**

• Utilizar un cliente de control de versiones

• Configurar servicios de control de versions centralizados

• Controlar las versiones de un documento

**Escenario**

* Git
* Github
* Code Commit AWS

**Resultados esperados**

• Configurar un repositorio local

• Crear un repositorio en GitHub

• Crear un reporsitorio en CodeCommit

• Controlar las versiones de un documento

# Marco Teórico

**Control de versiones**

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.

**Git**

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

# Diagrama

![Interfaz de usuario gráfica

Descripción generada automáticamente]()

# 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 "Carlos Morales"

b. git config --global user.email “carlos.morales@iteso.mx”

3. Configurar el repositorio

a. Crear en Mis Documentos una carpeta llamada git

b. Desde consola cambiarse a ese directorio

(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

**Configurar repositorio en GitHub**

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

a) https://github.com/

2. Crear un repositorio en GitHub

3. Consultar el URL del repositorio

a) https://github.com/jkmorales/awsPractica10

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

a) git remote add origin git@github.com:jkmorales/awsPractica10.git

b) git branch -M main

c) git push -u origin main

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

**Aparece nuestro archivo versiones.txt con todo el historial de commits y versiones de nuestro archivo, su historial.**

**Configurar repositorio en AWS CodeCommit**

1. Crear credenciales para CodeCommit

a. Desde la consola de IAM 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

<https://git-codecommit.us-east-1.amazonaws.com/v1/repos/labPrac10>

b. git push AWS master

**git push AWS10 main**

5. Consultar el repositorio de CodeCommit

**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

# Problemas y Soluciones

**Afortunadamente, para esta práctica no se presentó problema alguno.**

# Experimentos y Resultados.

• ¿Ves conveniente el uso de control de versiones para documentos que no son código?

Si, definitivamente. Puede ser código, documentos u otro recurso, sin duda el uso de git para documentar avances y poder colaborar con otras personas es indispensable y recomendable.

• ¿Cómo comparas GitHub y CodeCommit? Los dos son implementaciones DVCS de git

Al fin de cuentas viene siendo lo mismo, los comandos y la tecnología terminan siendo de **GIT.**

Al igual que github, codecommit, gitlab, gitbucket terminan siendo solo el lugar en donde se almacenarán los repositorios.

El lugar ya queda a consideración, dependiendo de las necesidades.

• Evidencia de la práctica

**Instalar y configurar GIT con la cuenta de ITESO**

**![Texto

Descripción generada automáticamente]()**

**Configurar Repositorio**

**![Interfaz de usuario gráfica, Texto

Descripción generada automáticamente]()**

**Control de versiones**

**![Texto

Descripción generada automáticamente]()![Texto

Descripción generada automáticamente]()![Texto

Descripción generada automáticamente]()![Texto

Descripción generada automáticamente]()**

**Crear y añadir repositorio en GitHub**

**![Texto

Descripción generada automáticamente]()**

**Crear Repositorio en CodeCommit**

**![Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente]()**

**Añadir repositorio local a codecommit**

****

**![Texto

Descripción generada automáticamente]()**

**Consultar Repositorio en codecommit**

**![Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente]()**

**Crear el documento del reporte, y añadirlo para tracking**

**![Texto

Descripción generada automáticamente]()**

**Logs de cambios en documento**

El documento se agregó casi al terminar el reporte de la práctica, es por eso que no se pudo ir versionando cada ajuste que se le hacía.

**URL del documento en codecommit**

# Costo

Explicare el costo de la solución implementada, justificando la solución implementada con base en los costos.

# 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).

# Bibliografía

En formato de la IEEE

**[1]** *Amazon Elastic Compute Cloud Documentation,* Amazon Web Services, Inc. [En línea]. Disponible en <https://docs.aws.amazon.com/ec2/index.html#lang/en_us>

**[2]** *Pro Git Book, GIT*. [En línea]. Disponible en: <https://git-scm.com/book/en/v2>

**[3]** *What is AWS CodeCommit?,* Amazon Web Services, Inc. [En línea]. Disponible en:

https://docs.aws.amazon.com/codecommit/latest/userguide/welcome.html