Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática

Ingeniería de Software

CI-0126

Laboratorio 1

Uso básico de GitHub

Repositorio del laboratorio
`https://github.com/MarioCordero/my-labs-inge-soft-CI-0126`

Profesora: Msc. Rebeca Obando Vásquez

Estudiante: Mario Cordero

Carnet: C22306

26 de agosto de 2025

1. Comandos Avanzados de Git

1.1. Rebase and Merge

¿Qué es rebase and merge?

Es una estrategia de integración que combina las ideas de **rebase** y **merge**. Primero, Git toma los commits de la rama de trabajo y los aplica sobre la rama principal como si se hubieran creado directamente allí desde un inicio (**rebase**). Luego, en vez de hacer un commit de merge, sólo se va a la rama principal.

¿Cuándo usarlo?

- Se usa cuando queremos integrar cambios en nuestro proyecto de manera clara pero sin mezclar commits de más, así no se pierde detalle.

1.2. Squash and Merge

¿Qué es squash and merge?

Esta estrategia combina todos los cambios o commits de una rama en un solo commit a la rama donde se va a hacer merge, entonces en la rama donde se hiz merge aparece solo un commit, aunque la rama "hija"haya tenido muchísimos más commits.

¿Cuándo usarlo?

- Se usa cuandop hicimos un montón de commits irrelevantes en una rama y se puede resumir en un solo commit, para evitar saturar de commits la rama a la que se la va a hacer merge.

1.3. Git Reset

¿Qué hace?

El comando git reset sirve para mover el puntero **HEAD** a un commit anterior y decidir qué hacer con los cambios que estaban después de ese punto.

¿Qué es HEAD?
Marcador que apunta al commit actual en el que se está trabajando.

Modos disponibles:

- `-soft`: Mueve HEAD a un commit anteriror y mantiene cambios en staging.
- `-mixed`: Mueve HEAD a un commit anteriror, se sacan los cambios de staging pero se mantiene en los archivos (Working directory).
- `-hard`: Mueve HEAD a un commit anteriror y descarta absolutamente todos los cambios. El trabajo no guardado se pierde para siempre.

1.4. Git Revert

¿Qué hace?

Crea un nuevo commit que deshace cambios previos sin borrar el historial, entonces tenemos una versión "nuevaçon los cambios revertidos, pero si se necesita se puede acceder a esos cambios.

¿Cuándo usarlo?

- Se usa para deshacer cambios que ya están en el repositorio remoto y que otras personas pueden estar usando.

1.5. Diferencia entre Git Reset y Git Revert

La diferencia entre Git Reset y Git Revert está en cómo afectan el historial del repositorio. Cuando usamos **reset**, lo que hacemos es reescribir el historial, o sea, movemos el puntero de la rama hacia un commit anterior y descartar los que vinieron después.

En cambio, **revert** no borra commits ni altera el historial existente sino que crea un nuevo commit que “revierte” los cambios de uno anterior y así se puede usar sin problemas en ramas compartidas, ya que mantiene el camino o un historial de lo ocurrido y permite deshacer errores sin perder información. Por eso, mientras reset es más agresivo y se usa principalmente de forma local, revert es la opción adecuada cuando se trabaja en equipo.