**Instalando Git en Linux**

Cada distribución de Linux tiene un comando especial para instalar herramientas y actualizar el sistema.

En las distribuciones derivadas de Debian (como Ubuntu) el comando especial es apt-get, en Red Hat es yum y en ArchLinux es pacman. Cada distribución tiene su comando especial y debes averiguar cómo funciona para poder instalar Git.

Antes de hacer la instalación, debemos hacer una actualización del sistema. En nuestro caso, los comandos para hacerlo son sudo apt-get update y sudo apt-get upgrade.

Con el sistema actualizado, ahora sí podemos instalar Git y, en este caso, el comando para hacerlo es sudo apt-get install git. También puedes verificar que Git fue instalado correctamente con el comando git --version.

# Git reset vs. Git rm

Git reset y git rm son comandos con utilidades muy diferentes, pero aún así se confunden muy fácilmente.

## git rm

Este comando nos ayuda a eliminar archivos de Git sin eliminar su historial del sistema de versiones. Esto quiere decir que si necesitamos recuperar el archivo solo debemos “viajar en el tiempo” y recuperar el último commit antes de borrar el archivo en cuestión.

Recuerda que git rm no puede usarse así nomás. Debemos usar uno de los flags para indicarle a Git cómo eliminar los archivos que ya no necesitamos en la última versión del proyecto:

* git rm --cached: Elimina los archivos del área de Staging y del próximo commit pero los mantiene en nuestro disco duro.
* git rm --force: Elimina los archivos de Git y del disco duro. Git siempre guarda todo, por lo que podemos acceder al registro de la existencia de los archivos, de modo que podremos recuperarlos si es necesario (pero debemos usar comandos más avanzados).

## git reset

Este comando nos ayuda a volver en el tiempo. Pero no como git checkout que nos deja ir, mirar, pasear y volver. Con git reset volvemos al pasado sin la posibilidad de volver al futuro. Borramos la historia y la debemos sobreescribir. No hay vuelta atrás.

Este comando es **muy peligroso** y debemos usarlo solo en caso de emergencia. Recuerda que debemos usar alguna de estas dos opciones:

Hay dos formas de usar git reset: con el argumento --hard, borrando toda la información que tengamos en el área de staging (y perdiendo todo para siempre). O, un poco más seguro, con el argumento --soft, que mantiene allí los archivos del área de staging para que podamos aplicar nuestros últimos cambios pero desde un commit anterior.

* git reset --soft: Borramos todo el historial y los registros de Git pero guardamos los cambios que tengamos en Staging, así podemos aplicar las últimas actualizaciones a un nuevo commit.
* git reset --hard: Borra todo. Todo todito, absolutamente todo. Toda la información de los commits y del área de staging se borra del historial.

**¡Pero todavía falta algo!**

* git reset HEAD: Este es el comando para sacar archivos del área de Staging. No para borrarlos ni nada de eso, solo para que los últimos cambios de estos archivos no se envíen al último commit, a menos que cambiemos de opinión y los incluyamos de nuevo en staging con git add, por supuesto.

# ¿Por qué esto es importante?

Imagina el siguiente caso:

Hacemos cambios en los archivos de un proyecto para una nueva actualización. Todos los archivos con cambios se mueven al área de staging con el comando git add. Pero te das cuenta de que uno de esos archivos no está listo todavía. Actualizaste el archivo pero ese cambio no debe ir en el próximo commit por ahora.

¿Qué podemos hacer?

Bueno, todos los cambios están en el área de Staging, incluido el archivo con los cambios que no están listos. Esto significa que debemos sacar ese archivo de Staging para poder hacer commit de todos los demás.

¡Al usar git rm lo que haremos será eliminar este archivo completamente de git! Todavía tendremos el historial de cambios de este archivo, con la eliminación del archivo como su última actualización. Recuerda que en este caso no buscábamos eliminar un archivo, solo dejarlo como estaba y actualizarlo después, no en este commit.

En cambio, si usamos git reset HEAD, lo único que haremos será mover estos cambios de Staging a Unstaged. Seguiremos teniendo los últimos cambios del archivo, el repositorio mantendrá el archivo (no con sus últimos cambios pero sí con los últimos en los que hicimos commit) y no habremos perdido nada.

**Conclusión**: Lo mejor que puedes hacer para salvar tu puesto y evitar un incendio en tu trabajo es conocer muy bien la diferencia y los riesgos de todos los comandos de Git.

git init inicio

Git add

DIRECTORIO DE

TRABAJO

PREPARACION

O

STAGING

REPOSITORIO LOCAL

Entorno de desarrollo personal

Archivo. HTML “Tracked” o rastreado

Git add

Archivo.CSS “Untracked” o no rastreado

Git add

DIRECTORIO DE

TRABAJO

PREPARACION

O

STAGING

REPOSITORIO LOCAL

Entorno de desarrollo personal

Después que tienes datos en el staging se manda commit

Entorno de desarrollo personal

Git commit

DIRECTORIO DE

TRABAJO

PREPARACION

O

STAGING

REPOSITORIO LOCAL

GitHub, GitLab, BitBucket,

**Flujo de trabajo básico con un repositorio remoto**

Por ahora, nuestro proyecto vive únicamente en nuestra computadora. Esto significa que no hay forma de que otros miembros del equipo trabajen en él.

Para solucionar esto están los **servidores remotos**: un nuevo estado que deben seguir nuestros archivos para conectarse y trabajar con equipos de cualquier parte del mundo.

Estos servidores remotos pueden estar alojados en GitHub, GitLab, BitBucket, entre otros. Lo que van a hacer es guardar el mismo repositorio que tienes en tu computadora y darnos una URL con la que todos podremos acceder a los archivos del proyecto para descargarlos, hacer cambios y volverlos a enviar al servidor remoto para que otras personas vean los cambios, comparen sus versiones y creen nuevas propuestas para el proyecto.

Esto significa que debes aprender algunos nuevos comandos:

* **git clone url\_del\_servidor\_remoto**: Nos permite descargar los archivos de la última versión de la rama principal y todo el historial de cambios en la carpeta .git.
* **git push**: Luego de hacer git add y git commit debemos ejecutar este comando para mandar los cambios al servidor remoto.
* **git fetch**: Lo usamos para traer actualizaciones del servidor remoto y guardarlas en nuestro repositorio local (en caso de que hayan, por supuesto).
* **git merge**: También usamos el comando git fetch con servidores remotos. Lo necesitamos para combinar los últimos cambios del servidor remoto y nuestro directorio de trabajo.
* **git pull**: Básicamente, git fetch y git merge al mismo tiempo.

Algunos comandos que pueden ayudar cuando colaboren con proyectos muy grandes de github:

1. git log --oneline - Te muestra el id commit y el título del commit.
2. git log --decorate- Te muestra donde se encuentra el head point en el log.
3. git log --stat - Explica el número de líneas que se cambiaron brevemente.
4. git log -p- Explica el número de líneas que se cambiaron y te muestra que se cambió en el contenido.
5. git shortlog - Indica que commits ha realizado un usuario, mostrando el usuario y el titulo de sus commits.
6. git log --graph --oneline --decorate y
7. git log --pretty=format:"%cn hizo un commit %h el dia %cd" - Muestra mensajes personalizados de los commits.
8. git log -3 - Limitamos el número de commits.
9. git log --after=“2018-1-2” ,
10. git log --after=“today” y
11. git log --after=“2018-1-2” --before=“today” - Commits para localizar por fechas.
12. git log --author=“Name Author” - Commits realizados por autor que cumplan exactamente con el nombre.
13. git log --grep=“INVIE” - Busca los commits que cumplan tal cual está escrito entre las comillas.
14. git log --grep=“INVIE” –i- Busca los commits que cumplan sin importar mayúsculas o minúsculas.
15. git log – index.html- Busca los commits en un archivo en específico.
16. git log -S “Por contenido”- Buscar los commits con el contenido dentro del archivo.
17. git log > log.txt - guardar los logs en un archivo txt

Servidor Remoto

**REPOSITORIO REMOTO**

Git Push

Git commit

Git add

Entorno de desarrollo personal

DIRECTORIO DE

TRABAJO

PREPARACION

O

STAGING

REPOSITORIO LOCAL