**COMANDOS**

**Git init**

Inicializa el repositorio, se prepara el entorno para versionar los archivos crear el archivo .git que es donde se guardan los cambios atómicos

**Git add**

**Git add .**  añade todo los archivos que este en ese directorio

**Git add**  nombre archivo añade el archivo seleccionado

**Git commit -m “** mensaje**”**

Envía los cambios del archivo a la base de datos del sistema de control de versiones.

**Git Status**

Muestra los archivos que se están versionando y cual es el estatus en el que están

**Git show**  nombre archivo.extensión

Muestra el detalle de todos los cambios hechos en los archivos y cual es el ultimo commit, este muestra mas deralle que el comando log

**Git log**

Muestra todos los commit hechos de los archivo junto con el id del commit este número sirve cuando quieres regresar a un commit anterior

**Git log** nombre archivo Muestra los commit del archivo seleccionado

**Git rm --cached** nombre archivo.extensión borra de la memoria ram lo que hiciste.Lo que hizo fue quitar el comando add

**Git config**

Muestra como se usa las configuraciones de git

**Git config --list** la configuración por defecto de tu git

**Git config --list --show-origin** muestra donde están las configuraciones guardadas

**Git config --global user.name “**yadira**”** configura quien hará los cambios

**Git config --global user.email “**micorreo**”** configura quien hará los cambios

**Git diff**

Te dice los cambios hechos

**Git diff** id commit v1 id commit v2 para comparar versiones/commits

**Git checkout** idcommit

Regresa a un commit en espacifico

**Git checkout** idcommit nombre archivo.extensión Regresa la versión de un archivo en el commit espedifico

**Git checkout master** nombre archivo.extensión Traer la versión que esta en master

**Git reset** idcommit

permite volver a una versión del id del commit

**Git reset** idcommit **--hard** todo vuelve al estado anterior

**Git reset** idcommit **--soft** lo que hiciste add no se borro

**Git log --stat**

Muestra las lineas modificadas de cada log/commit

**Git commit -a**

Hace el git add de los cambios, solo funciona con archivos que antes ya se le había dado un add.

**Git branch** nombre rama

Se crea un rama

**Git checkout** nombre rama

Cambiar de rama

**Comandos para trabajo remoto con GIT**

**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 merge 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.

Adicionalmente, tenemos otros comandos que nos sirven para trabajar en proyectos muy grandes:

**git log --oneline**:Te muestra el id commit y el título del commit.

**git log --decorate**: Te muestra donde se encuentra el head point en el log.

**git log --stat**: Explica el número de líneas que se cambiaron brevemente.

**git log -p**: Explica el número de líneas que se cambiaron y te muestra que se cambió en el contenido.

**git shortlog**: Indica que commits ha realizado un usuario, mostrando el usuario y el título de sus commits.

**git log --graph --oneline --decorate** y

**git log --pretty=forma**t:"%cn hizo un commit %h el dia %cd": Muestra mensajes personalizados de los commits.

**git log -3**: Limitamos el número de commits.

**git log --after=**“2018-1-2”

**git log --after=**“today” y

**git log --after=**“2018-1-2” **--before=**“today”: Commits para localizar por fechas.

**git log --author**=“Name Author”: Commits hechos por autor que cumplan exactamente con el nombre.

**git log --grep**=“INVIE”: Busca los commits que cumplan tal cual está escrito entre las comillas.

**git log --grep**=“INVIE” –i: Busca los commits que cumplan sin importar mayúsculas o minúsculas.

**git log –** index.html: Busca los commits en un archivo en específico.

**git log -S** “Por contenido”: Buscar los commits con el contenido dentro del archivo.

**git log >** log.txt: guardar los logs en un archivo txt

# Linea de comandos

**Pwd** en que carpeta estas

**Ls** muestra el contenido de la carpeta/directorio

**Espacio** significa que el comando anterior es el principal

**-** que es un argumento es decir que no escribes toda la palabra

-- significa que vas a usar una palabra

**Ls -al** muestra el contenido de la carpeta/directorio, oculto y los coloque en una lista

**Ls -l** muestra el contenido de la carpeta/directorio y los coloque en una lista

**Ctrl + l** limpia la consola

**Cd U + tab** muestra lo que empieza con la letra U

**~** home

**Mkdir** nombre carpeta Crear una carpeta

**Touch** nombre archivo.extensiónCrear un archivo

. Carpeta actual

.. carpeta interior

**Cat**  nombre archivo.extensión mostrar contenido de un archivo

**History** historia de comando

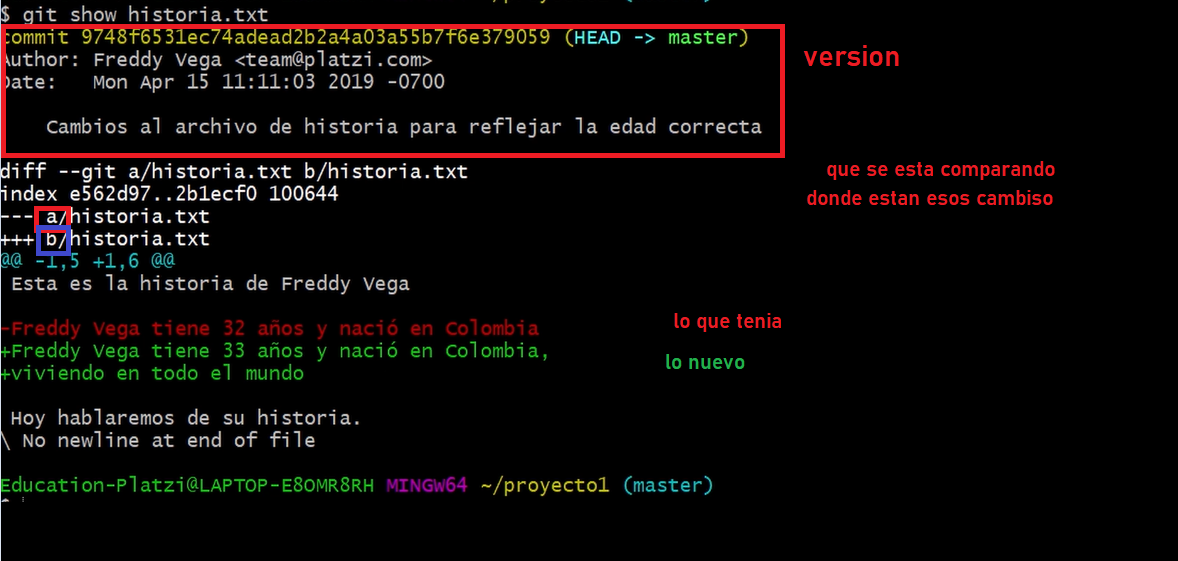
**!** numero de comando ejecuta un comando anterior

**Rm** nombre archivo.extensión borra un archivo

Comando --help muestra las opciones del comando

Notas:en Linux y Mac si importa mayúscula y minúsculas

# Como leer los logs de un archivo



# Editor Vim

**Esc + shift + zz** salir de la consola