**Comandos básicos**

Mkdir – nueva carpeta

Cat – visualizar contenido

Cd – acceder a una dirección

Cd .. – volver atrás en ruta

Cd / - home

Rm – remover

Rm -cached – remover de raíz

Pwd – ver en donde estamos

Ls – archivos donde estamos

Ls -al – archivos ocultos

**Comandos para git**

* **Validar usuario**

Git config –global user.name “Ale”

Git config –global user.email “alemau97@gmail.com”

* **Básicos**

Git init – iniciar repositorio

Code – abrir VSC

Git add – añadir archivo

Git status – ver el estado de git

Git commit – enviar cambios

Git commit -m “” – enviar cambios con un mensaje

**CUANDO SE MODIFICA UN ARCHIVO HAY QUE VOLVER A AÑADIRLO ANTES DE HACER UN COMMIT**

Git log – historial de git

Git show archivo.txt – ver historial de cambios del archivo

**ESC + Shift + Z + Z** = forzar a enviar un commit aunque no tenga un mensaje

Git diff akslhdfdf4654631619d4 a341d4fas3d5f6d4f3sd – compara versiones de commit, se copian y se pegan los números

Checkout – me traigo los cambios de una rama a mi carpeta

Merge – unir los cambios de una rama con otra

Git reset jñjdsjf64fd64s1ds4fs4d6 - -hard (más peligroso pero efectivo) VUELVE A UNA VERSION ANTERIOR

Git reset a6sd4f6ad4sfsa4a4sffd4 - -soft (borra del git pero sigue manteniéndolo en el staggin para dar commit)

Git checkout 56asd4f654sddf5d4f – trae devuelta el archivo que quizá resetee (MUCHO MAS COMPLETO QUE EL RESET, PORQUE RESET VUELVE AL PASADO Y ES DEFINITIVO, CHECKOUT VUELVE AL PASADO PERO PODEMOS RECUPERARLO FACILMENTE Y CONTINUAR MODIFICANDOLO)

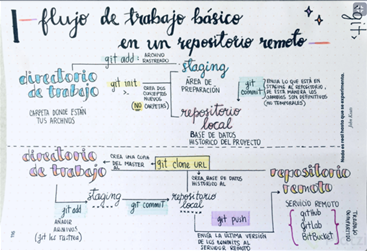
Git checkout master 1df64asd6f4s6ad4 – trae el archivo de otra rama

* 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 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.

Git clone url – se trae los archivos de otro repositorio a nuestra área de trabajo y a la vez a nuestro repositorio local

Git push – envio todos los archivos de mi servidor local al servidor remoto

Git pull – trae los archivos del servidor remoto tanto a mi área de trabajo o archivos como a mi repositorio local



Git commit -am “texto” - Hace la misma función que add al modificar un archivo YA CREADO

**ESC + I** sirve para poder escribir un texto en la pantalla que nos da si se nos olvido agregarlo a la hora de hacer un commit

Git Branch nombre – para crear una nueva rama

Git checkout nombre – me cambio a otra rama

**DESDE LA RAMA MASTER:** git merge cabecera (LUEGO EN LA PANTALLA LE AGREGAMOS EL TEXTO DEL COMMIT)

**AL HACER MERGE PUEDE GENERARSE CONFLICTO PERO SE SOLUCIONA DESDE VSC AL ACEPTAR CAMBIOS O NO**

**--------------- USO DE GITHUB ----------------**

**/////**

git checkout master git **branch** -m main

git config --global init.defaultBranch main

//// HACERLO PORQUE AHORA SE LLAMA MAIN SIEMPRE

**Git remote add origin <url>** - para hacer conexión con el repositorio online de Github

Fetch – traer cosas

Push – enviar cosas

Pull – traer mas completo todo

Git remote – verificar la conexión

Git remote -v – ver directorios existentes

**Git push origin main – estamos diciendo que la info que tenemos en main(nivel local) lo enviamos a origin (que acabamos de hacer la conexión con github)**

**Git pull origin main – me traigo todo al principio**

**Git pull origin main - - allow-unrelated-histories : me traigo todo nuevamente**

**Git push origin main – PARA ENVIAR NUEVAMENTE TODO**

**HACERLO SIEMPRE**

**Se puede hacer cambios desde GitHub y para que agarre a nivel local también lo hago como GIT PULL ORIGIN MAIN**

**-LLAVES-**

ssh-keygen -t rsa -b 4096 -C "mauriciomn@hotmail.es" 🡨------- PARA CREAR LLAVE PUBLIC/PRIV EN GITHUB-----🡪

eval $(ssh-agent -s) = Para verificar que el proceso se ejecuta

ssh-add ~/.ssh/id\_rsa = se agrega la llave

**La llaves se encuentran en la carpeta C/users/ale/.sh**

**RAMAS--------------**

git Branch -d = borrar una rama

git show-branch = historial de las ramas

git show-branch --all

gitk – ver las cosas a nivel grafico

git push origin cabecera – enviar nueva rama

**UNION DE OTROS USUARIOS A MI REPOSITORIO**

Git clone url(el del repositorio) – SE HACE DESDE EL PERFIL DEL NUEVO INTEGRANTE

Vim archivo – Jala el archivo para poder ser modificado

**RAMA FIX-TYPO**

git Branch fix-typo : rama para hacer correcciones

Pull request sirve para comparar diferencias entre ramas

.gitignore = para ignorar archivos (esconder del repositorio)(se crea en vsc)

\*.jpg (por ejemplo)

---**PARA SUBIR PAGINA Y QUE SEA UN DOMINIO MAS CORTO----**

Hay que nombrar el repositorio: nombre.github.io

------

Git rebase master = desde una nueva rama, y se ha escrito algo nuevo, une los cambios a la rama master (cambia la historia y desaparece la otra rama de la historia)

Git stash = he hecho cambios en un archivo, me quiero ir a otra rama, pero los cambios que he hecho aún no lo quiero hacer commit, entonces los guardo en un lugar temporal y me cambio de rama

Git stash list = archivos guardados momentáneos

Git stash pop = deshago el stash (saco los cambios a nivel de stagin de nuevo)

Git clean - - dry-run = hace una prueba de lo que va a borrar (principalmente las copias de archivos)

Git clean -f = borra los archivos

Git log –oneline = historial

Git Cherry-pick lkdjf41 = me traigo un commit viejo de otra rama a mi rama actual

Git commit –amend = corrijo el ultimo commit ( si esto sucede antes hay que hacer un add)

**PARA CASOS DE EMERGENCIA (TODO SE PIERDE)**

Git reflog = ver que ha pasado (lo mejor en historia)

Git reset <coloco el tag>

Git reset - -hard sladfld4563

**PARA BUSCAR ESPECIFICAMENTE**

Git grep color(palabra a buscar donde se usó) = busca en donde se usó esa palabra

Git grep -n color = específicamente en que línea

Git grep -c color = cuantas veces se ha usado la palabra

Git log -S “cabecera” = cuantas veces usé esa palabra en el historial

**A NIVEL GRUPAL**

* **git shortlog -sn** = muestra cuantos commit han hecho cada miembros del equipo.
* **git shortlog -sn --all** = muestra cuantos commit han hecho cada miembros del equipo hasta los que han sido eliminado
* **git shortlog -sn --all --no-merge** = muestra cuantos commit han hecho cada miembros quitando los eliminados sin los merges
* **git blame ARCHIVO** = muestra quien hizo cada cosa linea por linea
* **git COMANDO --help** = muestra como funciona el comando.
* **git blame ARCHIVO -Llinea\_inicial,linea\_final**= muestra quien hizo cada cosa linea por linea indicándole desde que linea ver ejemplo -L35,50
* \*\*git branch -r \*\*= se muestran todas las ramas remotas
* **git branch -a** = se muestran todas las ramas tanto locales como remotas