- **git init** = sirve para crear las estructuras necesarias para crear un repositorio de git

- **git status** = dice en que parte están los archivos

- **git add nombreArchivo o ruta** = pasamos el archivo de Working Directory a Stagging

- **git commit** = pasas los archivos de Stagging al repositorio local

- **git log --oneline** = nos salen los archivos y nos salen en que versión estan con todos sus datos al hacer el commit

- **gitignore** = contiene máscaras y todas las que esten alli no se metan en Stagging

- si al hacer un git status sale un archivo en Untracked, es que nunca se ha movido a ninguno de los 3 bloques: working area, stagging, commit

- **git diff** = te muestras las diferencias que tienen los archivos al hacer un cambio

- **git difftool** = sirve para ver el diff de forma más visual

- **git remote add origin urlDelRepositorioEnLaNube** = para conectarse a un repositorio remoto

- **git fetch origin** = recoje los objetos capturados en el add origin, es decir los muestra

- **git pull origin master** = siendo master tu rama actual, se descarga las cosas de origin

- **git push** = sube los archivos

- **git clone** = no habria que hacer por separado, init, fetch ni pull, ya que se hace a la vez

- **git branch -v -a** = da la información de tu rama

- **git checkout** (hash o HEAD (si es la última versión y si queremos 1 menos o menos del head usaremos virgulillaNumero))(-- nombreArchivoS o nombreRama) = para traer de commit al area de trabajo

- **git reset** (hash o HEAD (si es la última versión y si queremos 1 menos o menos del head usaremos virgulillaNumero))(-- nombreArchivoS o nombreRama) = para traer de Stagging al area de trabajo

- **git revert** hash del commit hecho con git commit -m = revierte un commit realizado

- **git branch** nombreRama ramaPartida

- **git checkout nombreRamaCreadaAnteriormente** = para moverse a la rama nueva

- **git branch -a(remoto) -v** (mensaje de commit del head de la rama, es decir el ultimo) = te lista las ramas creadas

- **git merge ramaDeDondeSeSaca ramaHaciaDondeVa** = para fusionar los cambios de la nueva rama sobre otra rama

- **git branch -d nombredeRamCreada** = para borrar la rama

- **git checkout -b nombreNuevaRama** = se usa para crear una nueva rama y posicionarte sobre ella con checkout

- con blame sabremos qué persona ha realizado algún cambio

- **git blame nombreArchivo** = nos dice que autor ha realizado una modificación a un archivo

- **git blame -L min,max nombreArchivo** = nos dice que autor ha realizado una modificación en el rango de lineas que se establece de un archivo

-- PREGUNTAR --

-con un git chechout puedo moverme entre ramas, puedo moverme a un puntero en el tiempo y generar todo el contenido o para crear una rama, TODO SEGUN ARGUMENTOS LE PASEMOS

-se podria hacer merge con las cosas que nosotros queremos con cherry-pick

-git cherry-pick remoto/principal hashid == se hace un merge del commit que nosotros queremos

-git cherry-pick --continue = se continua con el merge que hemos iniciado con el anterior comando

-git checkout --theirs . = haremos un merge con el archivo remoto

-git rebase nos permite hacer "commits comprimidos", es decir guardar una sucesión de muchos commits en uno

-git rebase --interactive(nos saldría una pantalla para decir de donde a donde queremos el rebase) --root(siendo el primer commit, se pueden poner rango con HEAD...HEADvirgulillaNumero) = nos haría un rebase de una serie de commit

-las etiquetas nos ayudan a poner un nombre de un commit en vez de un hash, puede ser ligera o anotada (más información)

-git tag -a v1.5 -m "nombreEtiqueta" = para poder ponerle una etiqueta a un commit

-git stash = crea una foto de un archivo, y te guarda el estado de este y te guarda un hashid, nos sirve para poder recuperar un archivo si se modifica

-git add --amend = para mover el puntero un punto hacia atras y arreglar un problema que hemos tenido en un parado