GitHub

<COMANDOS GIT>

1.	**Git init**	crea el repositorio git
2.	**Git status**	revisa el estado del repositorio y define 2 tipos de archivo **Untracked** & **Tracked**
3.	**Git add**	agrega archivos al staging área
4.	**Git add .**	agrega todos los archivos al staging área
5.	**Git commit -m "mensaje"**	agrega archivos al repositorio local
6.	**Git commit -am**	fusiona las funciones de commit y add, pero solo para los archivos editados
7.	**Git commit --amend**	repara los cambios del ultimo commit en caso de error y lo sobrescribe
8.	**Git rm --cached**	borra un archivo del staging área
9.	**Git rm -f**	borra un archivo definitivamente
10.	**Git log**	historial de commits
11.	**Git log --oneline**	historial de commits resumido
12.	**Git log --stat**	historial de commits mostrando la cantidad de líneas agregadas y quitadas
13.	**Git log --graph**	historial de commits con árbol de ramas
14.	**Git log > ruta_nombre.txt**	guarda en un archivo .txt el historial de commits
15.	**Git log -S palabra**	busca una palabra en el historial de commits
16.	**Git show**	visualiza los cambios de un archivo
17.	**Git show-branch--all**	muestra el historial de las ramas existentes
18.	**Git diff sha-1 sha-1**	compara commits con sha-1 o tag
19.	**Git checkout**	viaja entre commits y entre ramas, colocando el sha-1, nombre_archivo o rama
20.	**Git reset --soft**	reinicia hasta el commit indicado dejándote los archivos en el staging area
21.	**Git reset --mixed**	reinicia hasta el commit indiciado dejándote los archivos en el working directory
22.	**Git reset --hard**	reinicia hasta el commit indicado borrando todo lo que había después
23.	**Git tag**	agrega etiquetas a los commits: git tag -a Version -m "mensaje" sha-1
24.	**Git tag -l**	historial de tags
25.	**Git tag -d**	borra etiquetas con poner su nombre
26.	**Git fetch**	descarga las actualizaciones del repositorio remoto al local (poco usado)
27.	**Git pull origin master**	descarga todo el repositorio remoto al repositorio local
28.	**Git push origin master**	sube todo el repositorio local al repositorio remoto
29.	**Git push origin --tags**	envía los tags al repositorio remoto
30.	**Git branch**	crea ramas, puedes visualizar las ramas o crearlas con asignar un nombre
31.	**Git branch -D**	borra una rama con poner su nombre
32.	**Git branch -a**	muestra las ramas remotas y locales
33.	**Git checkout**	moverse entre ramas o en el tiempo (asignando sha-1, tags, nombre_archivo)
34.	**Git merge**	fusiona ramas, se hace el merge en la rama principal invocando la otra rama
35.	**Git clone url_repo_remoto**	clona un repositorio remoto en github a tu repositorio local

@dacara

git

GitHub

36. **Git remote add origin url** establece la conexión entre tu repo local y remoto, colocando la url de tu Github
37. **Git remote -v** muestra url del repo remoto
38. **Gitk** despliega el historial de commits en un software de forma más visual
39. **Git rebase** reescribe la historia del repo fusionando ramas silenciosamente (mala práctica)
40. **Git stash** guarda los cambios en memoria cuando no quieres hacer un commit aun
41. **Git stash save "mensaje"** guarda un stash con mensaje
42. **Git stash list** lista de todos los stash
43. **Git stash pop** trae de vuelta los cambios que teníamos guardados en el último stash
44. **Git stash apply stash@{n}** trae el stash que necesites con indicar su número head dentro de las llaves
45. **Git stash drop** borra el ultimo stash
46. **Git stash clear** borra todos los stash
47. **Git clean --dry-run** vista previa de los archivos que se borraran que están en el working directory
48. **Git clean -f** borra los archivos que no desees
49. **Git cherry-pick sha-1** trae un commit viejo al head (mala práctica)
50. **Git reflog** historial completo, muestra hasta los commits borrados y sobrescritos
51. **Git grep** buscador de caracteres (el código se busca dentro de " ")
52. **Git grep –n** indica en que línea se encuentra la búsqueda
53. **Git grep -c** indica el número de repeticiones y lugar donde se encuentro la búsqueda
54. **Git shortlog** historial de commits por colaborador
55. **Git shortlog -sn** número de commits que hizo cada colaborador
56. **Git shortlog -sn --all** número de commits que hizo cada colaborador incluyendo los eliminados
57. **Git blame archivo.txt** muestra todas las líneas del archivo, indicando la culpa de quien hizo cada línea
58. **Git blame archivo.txt -L#,#** muestra la culpa, pero indicando desde que línea, hasta que línea queremos
59. **Git comando --help** muestra el manual del comando indicado

COMANDOS DE CONFIGURACION DE GIT

1. **Git config --global alias.name** coloca un alias a un comando muy largo para invocarlo más rápido
2. **Git config --list** opciones de configuración
3. **Git config --global user.name** ingresa tu nombre
4. **Git config --global user.email** ingresa tu correo
5. **Git pull origin master --allow-unrelated-histories** se usa la primera vez para fusionar los repositorios
6. **Git remote add upstream url** trae la actualización de un repo open source creando otro origen

@dacara

git