**GIT**

*Recordar siempre trabajar con la última versión del programa*

* **cd** / para entrar a carpetas.
* **ls** / ver archivos dentro de carpetas.
* **git init** / iniciamos un repositorio (master).
* **git init --bare** / repositorio puro solo contiene modificaciones de los archivos, no copias. Solo controla y se puede acceder remoto.
* **git status** / ver estado, en que rama esta, commits, archivos sin agregar (aparecen en rojo, una vez hecho add se ponen en verde), etc.
* **git add** / agregamos archivos al commit. "." agregamos todos los archivos. "nombre del archivo" agregamos ese archivo solamente.
* **git rm --cached <file>** / para dejar de monitoriar un archivo. rm es remove.
* **git commit -m "Descripcion"** / al hacer actualizaciones del archivo hacemos un commit.
* **git config --global o --local user.email "manuelgarcia24696@gmail.com"** / Para tener data de quien hizo los cambios.
* **git config --global o --local user.name "Manuel Garcia"** / config generales. Para tener data de quien hizo los cambios.
* **git log** / Vemos un hash (identificacion unica del commit) del commit, branch, autor, fecha y comentarios del commit.
* **clar** / limpiar pantalla.
* **git log** --oneline / Lo mismo que git log pero en una linea.
* **git log -p** / aca vemos toda la data y te muestra las modificaciones en los commit.
* **:q o :x** / salimos.
* **--pretty="format%h"** / estos son para el git log, para editar que es lo que queremos ver, podemos cambiar el % para ver diferentes cosas.
* -Si no quiero mostrar algo, como archivos de configuracion del programa lo que debo hacer es crear un archivo en el proyecto que se llame **".gitignore"**. Dentro de ese archivo agregamos los archivos o carpetas a ignorar. Este archivo si se le debe hacer add. Ej.: carpeta/

a.congif

* **cd..** / volvemos para atras en las carpetas. Si agregamos /.. backeamos dos carpetas.
* **mkdir** / creamos carpeta.
* **git remote add "nombre" rutaDeAccesoAlServidor** / agrego servidor.
* **git remote** / lista los servidores remotos que nuestro repositorio conoce.
* **git remote -b** / nos indica si el servidor es funcional (fetch y push).
* **git clone DireccionDelServidor NombreQueLeQuieroDar** / Clonar repositorio. Ej.: /c/user/etc.
* **git push Servidor/repositorio NombreBranch** / Pusheo la info al servidor, primero elijo al servidor y luego el branch (local - global). Si agrego al git push un -u hace que simpre elija ese origen.
* **git remote rename ServidorACambiarNombre NuevoNombre** / renombramos servidor.
* **git pull Servidor/repositorio NombreBranch** / Bajamos del servidor la data y nos dice que cambios hubo.
* **git NombreBranch NombreBranchNuevo** / creamos un branch.
* **git checkout nombreBranch** / Cambiamos de branch donde laburamos. Trabajamos en diferentes branches para no pisar el trabajo del otro. Generalmente se trabaja en dif areas.
* **git switch** / Nuevo comandon en v2.23 para reemplazar git checkout
* **git checkout -b nombreBranch** / creamos un branch y nos movemos ahi.
* **git merge NombreBranch** / lo que hago es unir dos branches, todo lo que tiene el que estoy manejando y los cambios que hice en el otro.
* **git rebase NombreBranch** / cambio la base del commit de un branch a otro.
* **git restore nombreDelArchivo** / Es como un CTRL + Z.
* **git restore –staged NombreArchivo** / Es volver un comando atras en git.
* **git revert CodigoDelCommit** / Hacemos CTRL + Z al commit, en realidad crea un commit nuevo del archivo antes de ser cambiado. El CodigoDelCommit lo vemos con git log, podemos usar los primeros 7 caracteres.
* **git stash** / Guardar algo temporal, algo sin commitearlo para luego terminarlo y hacer commit. WIP (Work In Progress).
* **git stash list** / Veo todo lo que esta en stash.
* **git stash pop** / Datos guardados en stash los vuelvo a mi carpeta de trabajo para continuar con eso. Toma la ultima modificacion del stash. Genera un merge.
* **git stash apply 0** / (0 es el numero del stash, puede haber varios). Tomo un stash para trabajar con el. Ojo que va a seguir guardado en stash, para quitarlo de ahi debo escribir “git stash drop”.
* **git checkout NumeroCommit**

+

**git checkout -b NuevoBranch**

/ Asi podemos volver a un commit viejo y se guardara en ese nuevo branch.

* **git diff CodigoCommit1..CodigoCommit2** / Ver todo lo que se modifico de un commit a otro.

Si cambie algo y aun no hice un commit y quiero ver que cambie puedo poner el codigo “git diff”.

TAG: marca un punto fijo que no puede ser modificado.

* **git tag -a nombreDelTag** / Puedo llamarlo “v.0.1.0”, y agregando -m puedo sumar un comentario!

Puedo pushear el servidor al tag asi queda una version segura a la que poder volver.

Se pueden observar los tag en GitHub - Releases