**TIPOS:**

* **Local**: Vive en nuestro computador. Si le ocurre una catástrofe vamos a perder nuestro trabajo, además de que Ctrl + Z tiene cierta limitación
* **Centralizado**: Depende de un súper server donde está almacenado el repositorio. Si se quema el súper server vamos a llorar y perdemos nuestro trabajo.
* **Distribuido**: Cada participante del repositorio tiene una copia local y no afecta el trabajo del resto. No perdemos nuestro trabajo si nuestro se computador se daña, simplemente pedimos una copia a cualquier servidor donde esté almacenada la copia

**GIT FUE CREADO POR LINUS TORVALDS.**

* **Velocidad**: Puedes trabajar fluidamente desde tu computador
* **Diseño sencillo**: El código es robusto con las herramientas necesarias, como viajar en el tiempo
* **Fuerte apoyo en el desarrollo no lineal**: No trabaja de manera lineal, la línea del tiempo tiene bifurcaciones de manera independiente al proyecto principal
* **Completamente distribuido**: Cada quien puede tener una copia del proyecto.
* **Capaz de manejar grandes proyectos**: Linux, Django, Laravel, etc. Usan git

Git almacena referencias a los archivos que no se han cambiado.  
Cualquier trabajo es local, puedes trabajar en cualquier parte incluso sin internet.

**ESTADOS EN GIT**

* **Working Directory**: Es donde trabajamos de manera local, pero para guardarlo hay que pasarlo al Staging Area
* **Staging Area**: Es el área de preparación, se almacenan justo antes de hacer commit
* **git repository**: El repositorio donde almacenaremos los cambios del proyecto

**Comandos para manejo de directorios:**

**cd** = Change directory  
**mkdir** = Make directoryl  
**ls** = list -> en windows deben usar **dir**  
**clear** = también se puede usar control + L  
**touch** = crea un archivo o modifica su fecha de modificación.  
**rm** = remove -rf(para borrar carpetas)

Tratar de no usar **rm -rf /** o se despiden de su pc 😞

**i** = para poder editar  
**:wq** = para guardar y salir  
**:q!** = para salir sin guardar

**Comandos para manejo de GIT:**

**git init** nos crea un repositorio de manera local y lo hará en la carpeta donde estamos posicionados o se le puede pasar *[nombre\_de\_la\_carpeta]*y creará la carpeta con ese nombre.

**ls -a** nos muestra los archivos o carpetas ocultas en la terminal

En WIndows: Explorador de Archivos > pestaña **Vista**> marcar **Elementos Ocultos**

**Untracked files**: Son archivos que están en nuestro Working Directory, lo que aparezca en rojo es lo que se ha modificado y hay que pasarlo al Staging.

**git add [archivo]**: Nos agrega al archivo al staging area(el limbo) \*\* -A\*\* nos agrega todos los archivos

**git rm --cached:** Nos devolvemos un paso  
**git rm -f [file]:** Elimina el archivo por completo.  
**git add -n [file]:** Que no haga nada

**Git commit -m [“mensaje”]**: Es bueno ser descriptivos con el mensaje para saber lo que se hizo en ese commit y para informar al resto de personas.

**–amend**: concatena nuevos cambios con cambios previos.

**git log:** nos muestra la historia de todos los commits que hemos realizados.

**git tag**: nos permite agregar etiquetas a nuestros cambios.  
**-a** para la anotación  
**-m** para el mensaje

***-l*** nos muestra la lista de etiquetas  
**-f** para renombrar  
**-d** para borrar

**git diff [SHA1]**: Nos muestra las cambios de ese commit.  
**rojo**: fueron cambios que se quitaron  
**verde**: se agregaron cosas

**git reset --soft [SHA1]**: Nos permite quitar los cambios de un commit específico. Deja los archivos en el staging area, listos para hacer un commit.

**git reset --mixed [SHA1]**: nos elimina los cambios, también del staging area.

**git reset --hard [SHA1]**: Nos elimina los cambios incluso del working directory, es el más peligroso de todos porque podemos perder parte de nuestro trabajo.

Si tenemos el hash de los más recientes commits, puedes salvarte de los regaños de tu jefe todavía, importante guardar los commits en un archivo. Solo por si acaso.

**git stash**: es otro de los limbos, como el staging area. Para agregar los cambios estos deben estar en el staging area.  
**git stash list**: nos muestra la lista de stash que tengamos.  
**git stash drop stash@{numero}**: nos permite borrar un stash.  
**git stash apply**: aplicamos el último cambio

Las ramas son muy importantes si quieres trabajar con un equipo y no quieres tocar la rama master para no crear conflictos,

**git branch [nombre]** se crea una nueva rama  
**-l**: listamos las ramas  
**-d/-D [nombre]**: borramos rama  
**-m [nombre] [nombre\_nuevo]**: para renombrar ramas

**git checkout [nombre/sha1]**: Nos permite mover entre ramas y entre commits, no vamos a borrar nada. Acá es donde podemos movernos en el tiempo.

git checkout -b [nombre\_rama]: Nos permite crear una nueva rama sin necesidad de usar *branch*

**git merge [rama]**: Nos permite mezclar los cambios realizados en dicha rama con la rama en la que estamos.

**fast-forward**: los mezcla automáticamente  
**recursive/auto-merging**: ambas ramas salieron al mismo tiempo y hay algo nuevo en la rama que la otra no recuerda, por eso hace la mezcla recursiva.  
**manual merge**: nos va a tocar decirle a git específicamente los cambios que queremos mezclar

**git rebase:** hace prácticamente lo mismo que *merge*, cambiamos la historia de nuestro proyecto sin crear bifurcaciones del proyecto. Es mejor usar **merge**  
Usar solo git rebase de manera local.

-i: de manera interactiva, nos abrira el editor que tengamos definido en la configuración de git.

**git cherry-pick [SHA1]** nos permite cambiar un commit a otra rama para salvarnos la vida.