

## Utilidades de Git

**Staging área** = Área de memoria donde se añaden los archivos de manera segura previo commit de estos.

**Trackear** = En palabras sencillas tener un archivo modificado añadido en el staging.

**Inicializar repositorio en una carpeta**

*git init*

**Indicar de manera global quien eres**

*git config --global user.email "paulgrimaldobravo@gmail.com"*

*git config --global user.name "Paul Grimaldo"*

**Listar configuraciones globales**

*git config -l*

**Clonar desde repositorio remoto**

*git clone url\_project*

**Añadir cambios al staging**

*git add .*

**Ver el estado actual de git (Archivos en staging, sin trackear)**

*git status*

**Confirmar cambios y enviarlos al BD Local**

*git commit -m "Mensaje de commit"*

**Añadir origen remoto**

*git remote add origin url*

**Ver mis orígenes remotos**

*git remote -v*

**Traer cambios de repositorio remoto sin modificar el trabajo actual**

*git fetch origin master (Para combinarse con el actual necesito git merge)*

**Traer cambios de repositorio remoto combinando con el trabajo actual**

*git pull origin master*

**Traer archivos de repositorio sin commit comunes**

*git pull origin master --allow --unrelated --histories*

**Enviar cambios a un repositorio remoto**



*git push -u origin master*

**Eliminar origen remoto (A donde apunta)**

*git remote remove origin*

**Actualizar URL de repositorio remoto**

*git remote set --url origin url*

**Añadir una nueva fuente de datos remota**

*git remote add nombreFuente urlFuente*

*(Para traer cambios de esta Fuente solo necesito ocuparla en los pull - git pull nombreFuente miRama)*

El nombre general para estas nuevas fuentes si se tratan de proyectos a los que le hicimos un fork es "upstream".

**Ver el último cambio realizado**

*git show*

**Ver el historial del repositorio**

*git log*

**Ver el historial de repositorio de manera específica por archivo**

*git log -stat*

**Ver el historial completo**

*git log --all*

**Ver el grafo del repositorio**

*git log --all --graph --decorate --online*

Si estoy trabajando en Unix puedo usar un alias para todo el comando

*alias gitGraph = "git log --all --graph --decorate --online"*

**Volver a versiones anteriores**

*git reset 9383929(hash) --hard (Hard vuelve completamente en el tiempo)*

*git reset 9383929(hash) --soft (Soft mantiene los archivos que estan en el staging)*

**Ver cambios entre archivo en modificación vs archive añadido a staging**

*git diff*

**Obtener una versión anterior de un archivo por hash y por rama**

*git checkout 8089(hash) archivo.txt*

*git checkout master archivo.txt*

**Sacar todos los archivos del staging**

*git reset HEAD*

**Quitar un archivo en específico del staging**

*git rm --cached (No elimina los cambios fuera del staging)*

*git rm --force (Elimina los cambios en staging, disco y git)*

**Commit de cambios en archivos previamente creados (No funciona con archivos nuevos)**

*git commit -am "Mensaje del commit"*

**Ver ramas actuales**

*git branch*

*git show - branch --all*

**Crear una nueva rama**

*git branch develop*

**Cambiar de una rama a la otra**

*git checkout develop*

**Merge de 2 ramas**

*git checkout master (Por recomendacion jalar el resto de ramas a master y no al reves)*

*git merge develop (En master se crea un nuevo commit y se fusiona con esta rama)*

**Eliminar una rama**

*git branch -D develop*

**Ver ramas remotas**

*git branch -r*

**Ver todas las ramas (Locales y remotas)**

*git branch -a*

**Crear un nuevo Tag**

*git tag -a v0.1 -m "Mensaje del tag" 131322(Hash)*

**Ver todos los Tags**

*git tag*

**Ver Tags con sus referencias a commits**

*git show - ref --tags*

**Enviar los Tags a un repositorio remoto**

*git push origin --tags*

**Eliminar Tag del repositorio**

*git tag -d nombreTag (Luego tengo que hacer push de tags nuevamente)*

**Eliminar Tag del repositorio remoto**

*git push origin : refs/tags/nombreTag*

## **Abrir contexto visual de GIT (Un Helper adicional)**

*gitk*

## **Reorganizar una rama**

*git rebase nombreRama (nombreRama es la rama a la cual pegaré mis cambios).*

La manera correcta de usarlo sería primero hacer el rebase en la rama donde estoy con cambios, y luego en la principal con respecto a mi rama.

## **Añadir archivos de trabajo a stash**

*git stash*

## **Ver el listado de stash**

*git stash list (WIP = Work In Progress)*

## **Recuperar archivos de trabajo quitándolos del stash**

*git stash pop*

## **Mover archivos de trabajo en stash a una nueva rama**

*git stash branch nombreRama*

## **Eliminar archivos de trabajo en stash**

*git stash drop*

## **Limpiar repositorio de archivos basura**

*git clean --dry --run (Para simulación del comando)*

*git clean -f (Ejecuta el comando como tal, simular antes con el --dry --run)*

## **Traer un commit antiguo (o no) de otra rama**

*git cherry -pick 1212123(Hash)*

## **Pegar cambios en staging al commit anterior**

Los archivos modificados ya fueron añadidos al staging. (*git add.*)

*git commit --amend*

## **Corregir errores (Humanos y generalmente no sabemos qué ocurre con el repositorio)**

*git reflog (Permite ver el historial de los Head y como se han ido muriendo a traves del tiempo)*

*git reset --soft HEAD{12}*

*git reset --hard HEAD{12}*

Soft me regresa a ese head manteniendo lo que tenga en staging, mientras que hard no lo toma en cuenta.

HEAD{12} es un apuntador copiado usando el comando *git reflog*, también es válido usar el id del commit.

## Búsquedas usando GIT

*git grep busqueda*

Con `-n -c` puedo obtener la línea donde coincide con la búsqueda, y la cantidad de veces que se repite respectivamente.

*git grep -n busqueda, git grep -c busqueda*

Para búsquedas dentro del log de git puedo ocupar

*git log -S busqueda*

## Comandos colaborativos

*git shortlog (Log por persona)*

*git shortlog -sn (Cantidad de commits por persona)*

*git shortlog -sn --all (Incluye commits eliminados)*

*git shortlog -sn --all --no --merges (Sin incluir commits que son merge)*

Si deseo agregar todo este comando como un alias dentro de Git puedo hacerlo modificando su configuración global.

*git config --global alias.stats = "shortlog -sn --all --no --merges"*

De esta manera creé un nuevo comando que puedo ocupar como

*git stats*

## Ver quien modificó que

*git blame -c archivo.txt*

*git blame archivo.txt -L35,60 (Quien modifiko esto entre estas lineas)*

## Introducción a claves públicas y privadas para git

Cuando tengo un texto plano, por ejemplo *"Este es un texto plano"* y necesito enviarlo a través de internet necesito cifrar este mensaje, por lo que usaré una contraseña, al cifrar este mensaje asumamos que el texto plano se convierte en algo como *"DasdjahsdkaNDJASND12jnskD@#"* a esto voy a llamarlo ciphertext o texto cifrado, y para que el usuario que recibió mi mensaje pueda obtener el valor original tiene que ocupar la misma contraseña para convertir *"DasdjahsdkaNDJASND12jnskD@#"* en *"Este es un texto plano"*, pero, para que el usuario final pueda usar esta contraseña debería conocerla o también recibirla, por lo que se usa una técnica llamada claves públicas y privadas o cifrado asimétrico de un solo camino.

Genero un par de llaves (Publicas y privadas), envié mi llave publica a la persona que me quiera enviar el mensaje, la persona cifra su mensaje usando mi llave publica y yo recibo el ciphertext (Texto cifrado con mi llave publica) y descifro con mi llave privada, así obtengo el mensaje original.

Cabe recalcar que, la llave pública y privada están vinculadas matemáticamente.

Para generar un nuevo par de llaves

```
ssh - keygen - t rsa - b 4096 - C paulgrimaldobravo@gmail.com
```

Dependiendo de la ubicación de donde creé este par de llaves necesito copiar el contenido de mi llave publica y configurarlo en algún servidor repositorios remotos (*GitHub, GitLab, BitBucket*).