COMANDOS GIT

```
git --version
git --help
git --help branch
```

CLONAR UN REPOSITORIO

```
git clone <URL git.git>
cd proyecto-git
```

PARA CREAR EL REPOSITORIO LOCAL

```
git init
```

CONFIGURAR LA CUENTA DEL GIT

```
git config --global user.email "tucorreo@tuservicio.com"
git config --global user.name "tu_usuario"

git config  --list --global
```

CONFIGURAR EL EDITOR PREDETERMINADO DE GIT

```
git config --global core.editor "path/to/pycharm --wait"
git config --global core.editor "nano"
```

IGNORAR LOS ARCHIVOS A SEGUIRLOS

Crea el archivo ".gitignore", https://www.toptal.com/developers/gitignore

PARA VER ESTADO DE LOS ARCHIVOS

```
Git status
```

AÑADIR ARCHIVOS DEL WORKING DIRECTORY AL STAGING ÁREA

```
git add <NombreArchivo> o git add .
git status
```

VER LAS DIFERENCIAS

```
git add <NombreArchivo>
git status
git diff <NombreArchivo>
```

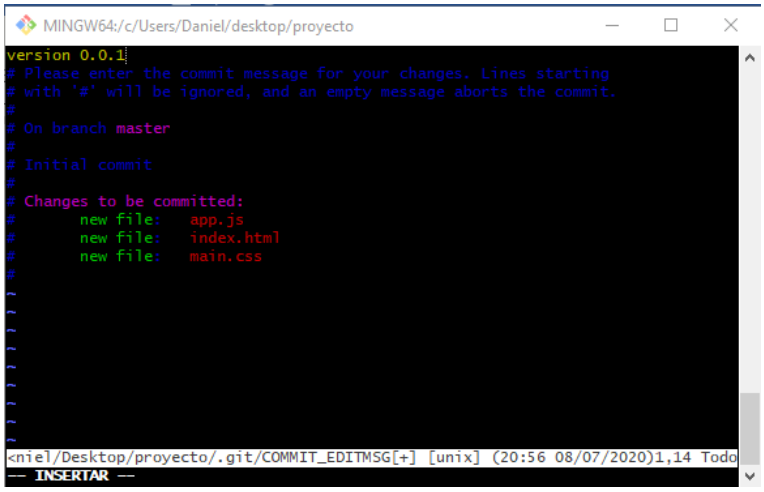
REGRESAR UN CAMBIO

```
git checkout -<NombreArchivo>
```

PRIMER COMMIT

```
git commit
git commit -m "mensaje"
```

Pulsar <i> para modificar, para salir de editor <esc> y para grabar y salir del editor <:wq><enter>



MODIFICAR EL MENSAJE

```
git commit --amend -m "mensaje"
```

CORRECCIONES EN LA CONFIRMACIÓN SIN CAMBIAR EL MENSAJE

```
git commit --amend --no-edit
```

VER LO COMMITS

```
git log
```

CONSULTAR LAS DIFERENCIAS ENTRE UN COMMIT Y OTRO

```
git diff <id_commit2>..<id_commit1>
```

DEVOLVERSE A UNA VERSIÓN ANTERIOR - COMMIT

```
git checkout <id_commit>.
```

La primera consiste en usar el id del commit al cual se desea regresar:

```
git reset --hard <id_commit>
```

La segunda, especificando el número de commits que se desea devolver:

```
git reset --hard HEAD~<numero_de_commit_atras>
```

VARIACIONES DEL COMANDO CHECKOUT

Deshacer cambios en su copia de trabajo local

```
git checkout HEAD <ruta_archivo>
```

SINCRONIZAR GIT CON GITHUB

```
git branch -M main
git remote add origin <URL git.git>
git push -u origin main
```

CREATE A NEW REPOSITORY

```
echo "# Prueba" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin <URL git.git>
git push -u origin main
```

PUSH AN EXISTING REPOSITORY

```
git remote add origin https://github.com/dgamarra/Prueba.git
git branch -M main
git push -u origin main
```

CAMBIAR EL URL DEL REPOSITORIO

```
git remote -v
git remote set-url origin https://github.com/dgamarra/HolaMundo.git
```

ACTUALICE LOS CAMBIOS EN EL REPOSITORIO REMOTO

```
git push -f origin main
```

RAMAS

PARA LISTAR LAS RAMAS

```
git branch

git branch -a
```

CREAR UNA RAMA

```
git branch <Nombre rama>
git branch
```

RENOMBRAR RAMA

```
git branch -m <nuevo_nombre>
```

RENOMBRAR UNA RAMA DIFERENTE (SIN CAMBIAR A ELLA)

```
git branch -m <nombre_viejo> <nuevo_nombre>
```

CAMBIAR DE RAMA

```
git checkout <Nombre rama>
git Branch
```

BORRA LA RAMA LOCAL (GIT)

```
git branch -d <Nombre rama>
```

BORRA LA RAMA REMOTA (GITHUB)

```
git push origin -delete <Nombre rama remota>
```

ACTUALIZAR EL REPOSITORIO REMOTO

```
git push origin --delete <nombre_viejo>    # Elimina la rama antigua en el remoto
git push origin <nuevo_nombre>             # Sube la rama con el nuevo nombre
```

ACTUALIZAR LAS REFERENCIAS REMOTAS EN OTROS CLONES

```
git fetch -prune
```

VINCULAR UNA RAMA LOCAL A UNA REMOTA:

```
git branch --set-upstream-to=origin/<nombre-de-la-rama>
```

ACTUALIZAR LA RAMA LOCAL CON CAMBIOS REMOTOS

```
git pull origin <nombre-de-la-rama>
```

SUBIR CAMBIOS LOCALES A LA RAMA REMOTA:

```
git push origin <nombre-de-la-rama>
```

COMPARAR LA RAMA LOCAL CON LA REMOTA

```
git diff remotes/origin/main main
```

CREAR Y EXTRAER ETIQUETAS

CREE UNA ETIQUETA ANOTADA PARA EL ÚLTIMO COMMIT

```
git tag -a <etiqueta> -m "mensaje"
```

LISTE LAS ETIQUETAS DEL REPOSITORIO.

```
git tag
```

CONSULTE LA ETIQUETA CREADA EN EL PRIMER PUNTO.

```
git show <etiqueta>
```

ELIMINE LA ETIQUETA V1.0.

```
git tag -d <etiqueta>
```

COMPARAR RAMAS

```
git diff --name-status < nombre-de-la-rama1>..< nombre-de-la-rama2>
```

MEZCLAR RAMAS

Mezcla la rama activa con la indicada

```
git merge <nombre-de-la-rama>
```

GIT REBASE

El comando git rebase permite "reorganizar" la historia de commits moviendo una rama a otro punto en el historial del repositorio. Es útil para mantener un historial de commits más limpio y lineal.
Existen dos tipos principales de rebase:

1. **Rebase en la misma rama**: Supongamos que trabajas en una rama llamada feature y quieres actualizarla con los últimos cambios de la rama main. Puedes hacer:

```
git checkout feature
git rebase main
```

Esto aplica los commits de feature encima de los commits más recientes en main, evitando un "merge commit". Es como si hubieras desarrollado tu trabajo sobre la versión actualizada de main.

2. **Rebase interactivo**: Usando -i (interactivo), puedes reordenar, editar, combinar o eliminar commits. Este es ideal para limpiar el historial antes de integrarlo en una rama principal:

```
git rebase -i HEAD~n
```

Donde n es el número de commits a revisar. Esto abre una interfaz donde puedes elegir qué hacer con cada commit.

GIT FETCH

El comando git fetch descarga los cambios y actualizaciones de una rama remota al repositorio local, pero no los fusiona ni los aplica automáticamente en tu rama actual.

```
git fetch origin
```

Este comando obtiene todas las actualizaciones del repositorio remoto (origin) y las coloca en tus referencias locales (e.g., origin/main), permitiéndote revisarlas antes de integrarlas a tu trabajo.

DIFERENCIA ENTRE GIT PULL Y GIT FETCH

- git fetch solo descarga los datos de la rama remota, sin aplicarlos a tu rama actual.
- git pull es una combinación de git fetch seguido de git merge, por lo que descarga y fusiona los cambios automáticamente.

Usar git fetch es útil para revisar el estado de la rama remota antes de tomar decisiones sobre cómo incorporar esos cambios.