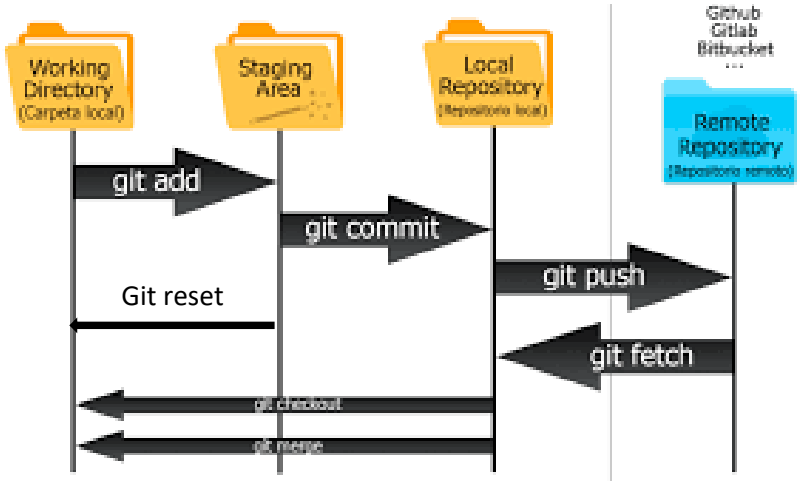
GIT COMANDOS



CONFIGURACIÓN

```
# Opciones obligatorias (nombre y correo)
git config --global user.name "Nombre y apellido"
git config --global user.email CORREO@ELECTRONICO
```

#Editor de preferencia:

- En Windows = Notepad ++

```
git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"
```

- En Linux = Visual Studio Code (si no se escoge este, git utiliza el editor vim por defecto, que es difícil)

```
git config --global core.editor "code --wait"
```

- Git init: se crea un repositorio (hay que situarse dentro de la carpeta deseada, o creándola mediante mkdir y ejecutar el comando)
- Git status: Identifica los archivos modificados/nuevos (rojo) y los archivos en área de preparación (verde)

```
xroot@mandalorian pr1 % git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   archivo1
        new file:   archivo2.txt
```

- Git diff / Git diff <archivo_o_ruta> : Compara tu directorio de trabajo con el área de preparación, enseñando los cambios realizados y que aún no están en la staging área.

  Working Directory ↔ Staging Area: What changes have I made but not yet staged? (`git diff`)

STAGING AREA (ÁREA DE PREPARACIÓN): Contiene los cambios que se añadirán a la nueva versión cuando ejecutemos un commit.

- Git add <archivo> / Git add . : Añade un archivo individual al stage area/ Añade todos los archivos nuevos o modificados al stage area.
- Git diff - - staged / Git diff - - Staged <archivo> : Compara el staging área con el último commit, enseñando lo que se incluirá en el próximo commit.

  Staging Area ↔ Repository: What changes have I staged that will be committed next? (`git diff --staged`)
- Git reset <archivo>: Quita el archivo del staging área (los cambios no se pierden)

  Al hacer un “git status”:
  - Cuando añadimos un fichero aquí aparece en color verde
  - Si modificamos el archivo de nuevo o modificamos otro, aparecerá en color rojo. Ya que no está en el área de preparación.
  - Si se ejecuta el “git commit” ahora solo se incluirá el verde.

COMMIT AREA: Guarda permanentemente todos los cambios realizados en un proyecto. Equivale a una nueva versión en el repositorio. Cada commit tiene un identificador único (hash).

- Git commit -m “MENSAJE”: Confirma los cambios (pasa del staged al commit)
- Git log / Git log - -graph : Muestra el histórico de commits del repositorio. Se puede navegar entre ellos.
- Git show <commit>: Nos permite mostrar los cambios que se introdujeron en un determinado commit. Con poner los 8-10 primeros caracteres del hash ya basta para identificar el commit.
- Guit checkout -- <archivo>: El archivo vuelve exactamente al estado del último commit. (Cualquier cambio no guardado en el repositorio se pierde.)

- Git tag NOMBRE_TAG: Crea un tag (alias) en el commit en que nos encontremos en este momento, facilitando su acceso sin necesidad de saber su hash.

LAS RAMAS

Git init: inicias la rama principal (ej: en una carpeta que se llama ramas)

Para activar la primera rama tenemos que hacer un commit

El asterisco cuando haces “git branch” significa que estás en la activa.

Si haces un commit a secas la rama se va a la master. Si la haces con el nombre de la rama se queda en esa rama.

---

git checkout rama --> para cambiar de rama

- Switched to branch x

El HEAD sirve para nombrar esa rama como principal/master, en la que vas a trabajar. Pero luego tienes que moverte a ella.