Recopilatorio de comandos de Git

1. Comandos básicos

- `git init`
  Inicializa un nuevo repositorio Git en el directorio actual.

- `git clone <URL>`
  Clona un repositorio remoto en tu máquina local.

- `git status`
  Muestra el estado actual del repositorio (archivos modificados, en stage, etc.).

- `git add <archivo>`
  Agrega un archivo específico al área de *staging* (preparado para el commit).

- `git add .`
  Agrega **todos** los archivos modificados al área de *staging*.

- `git commit -m "mensaje"`
  Guarda los cambios en el historial del repositorio con un mensaje descriptivo.

- `git log`
  Muestra el historial de commits del repositorio.

- `git log --oneline`
  Muestra el historial de commits de forma resumida (una línea por commit).

- `git diff`
  Muestra las diferencias entre los archivos en el directorio de trabajo y el área de *staging*.

2. Ramas (Branches)

- `git branch`
  Lista las ramas existentes en el repositorio.

- `git branch <nombre>`
  Crea una nueva rama con el nombre especificado.

- `git checkout <rama>`
  Cambia a una rama existente.

- `git checkout -b <nombre>`
  Crea una nueva rama y cambia a ella.

- `git switch <rama>`
  Otra forma de cambiar a una rama (alternativa moderna a `git checkout`).

- `git switch -c <nombre>`
  Crea una nueva rama y cambia a ella.

- `git merge <rama>`
  Fusiona la rama especificada con la rama actual.

- `git rebase <rama>`
  Reaplica los commits de la rama actual sobre la rama especificada.

- `git cherry-pick <hash>`
  Aplica un commit específico de otra rama en la rama actual.

3. Remotos (Remote Repositories)

- `git remote -v`
  Muestra los repositorios remotos asociados.

- `git remote add <nombre> <URL>`
  Agrega un nuevo repositorio remoto.

- `git pull`
  Descarga y fusiona los cambios del repositorio remoto a tu rama local.

- `git fetch`
  Descarga los cambios del repositorio remoto (sin fusionarlos automáticamente).

- `git push`
  Sube los cambios de tu rama local al repositorio remoto.

- `git push origin <rama>`
  Sube una rama específica al repositorio remoto.

4. Reescribir historial

- `git commit --amend`
  Modifica el último commit (útil para corregir el mensaje o añadir cambios).

- `git rebase -i HEAD~n`
  Reescribe los últimos **n** commits de forma interactiva (por ejemplo, para hacer *squash*).

- **Squash (combinar commits):**

  1. Ejecuta: `git rebase -i HEAD~n` (n es el número de últimos commits a revisar).
  2. Cambia `pick` por `squash` en los commits que quieras fusionar.
  3. Guarda y edita el nuevo mensaje del commit combinado.

5. Deshacer cambios

- `git restore <archivo>`
  Restaura un archivo modificado al último estado confirmado.

- `git restore --staged <archivo>`
  Quita un archivo del área de *staging* (sin perder los cambios locales).

- `git reset <archivo>`
  Lo mismo que `git restore --staged` (quita del *staging*).

- `git reset --hard`
  Revierte el estado del repositorio al último commit (¡pérdida de cambios no confirmados!).

- `git reset --soft HEAD~1`
  Deshace el último commit, pero mantiene los cambios en el área de *staging*.

- `git revert <hash>`
  Crea un nuevo commit que revierte los cambios del commit especificado.

6. Etiquetas (Tags)

- `git tag`
  Lista todas las etiquetas del repositorio.

- `git tag <nombre>`
  Crea una nueva etiqueta ligera.

- `git tag -a <nombre> -m "mensaje"`
  Crea una etiqueta anotada con un mensaje descriptivo.

- `git push origin <nombre>`
  Sube una etiqueta al repositorio remoto.

- `git push origin --tags`
  Sube **todas** las etiquetas locales al repositorio remoto.

7. Inspección y Depuración

- `git blame <archivo>`
  Muestra quién hizo cada línea de un archivo y en qué commit.

- `git show <hash>`
  Muestra los detalles de un commit específico.

- `git log -p`
  Muestra el historial de commits con los cambios realizados.

8. Stash (Guardar cambios temporalmente)

- `git stash`
  Guarda los cambios no confirmados temporalmente.

- `git stash pop`
  Recupera los últimos cambios guardados con `git stash`.

- `git stash list`
  Muestra la lista de *stashes* guardados.

- `git stash drop`
  Elimina el último *stash* guardado.

9. Alias (Atajos personalizados)

`git config --global alias.<alias> '<comando>'`
Crea un alias personalizado. Ejemplo:
git config --global alias.co 'checkout'

- Ahora puedes usar `git co` en lugar de `git checkout`.