📌 Comandos Básicos y Fundamentales

git config

Configura tu identidad y preferencias de Git.

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu@email.com"
```

Usa `--global` para que aplique a todos tus repos.
También puedes configurar el editor, colores, alias, etc.

---

git init

Crea un nuevo repositorio Git en el directorio actual.

```
git init
```

Úsalo para empezar a usar Git en un proyecto local.

---

git clone

Clona un repositorio remoto a tu máquina local.

```
git clone https://github.com/usuario/repositorio.git
```

Crea una copia completa del repositorio y su historial.

---

git add

Agrega archivos al área de preparación (staging area).

```
git add archivo.txt
git add .   # agrega todos los archivos modificados
```

Prepara los archivos para ser guardados con `git commit`.

---

git commit

Guarda los cambios del área de preparación al historial del proyecto.

```
git commit -m "Mensaje del commit"
```

Es como tomar una "foto" del estado del proyecto.

---

git status

Muestra el estado del repositorio: qué está modificado, sin seguir, listo para commit, etc.

```
git status
```

---

git log

Muestra el historial de commits.

```
git log
```

Ramas y Flujos

---

git branch

Muestra las ramas existentes o crea una nueva.

```
git branch          # muestra ramas
git branch nueva-rama  # crea rama
```

---

git checkout

Cambia de rama o revierte archivos.

```
git checkout main          # cambia a rama main
git checkout -b nueva      # crea y cambia a nueva rama
```

---

git switch (moderno)

Mejor alternativa a checkout para cambiar de rama.

```
git switch main
git switch -c nueva
```

---

git merge

Combina una rama con la actual.

```
git merge feature-1
```

Se usa para integrar cambios de otras ramas.

---

git rebase

Reaplica commits de una rama sobre otra (reorganiza la historia).

```
git rebase main
```

Útil para limpiar el historial antes de integrar ramas.

---

git cherry-pick

Aplica un commit específico de otra rama en la actual.

```
git cherry-pick abc1234
```

Ideal para traer un cambio puntual sin hacer merge.

---

git reset

Deshace commits o saca archivos del staging.

```
git reset --soft HEAD~1    # deshace el último commit, mantiene cambios
git reset archivo.txt       # quita archivo del staging
```

---

git stash

Guarda cambios sin hacer commit para trabajar en otra cosa.

```
git stash
git stash apply
```

Muy útil para guardar trabajo temporal sin hacer commit.

---

git worktree

Permite trabajar con varias ramas al mismo tiempo en directorios distintos.

```
git worktree add ../otro-main main
```

Ideal para evitar cambiar de rama constantemente.

🌐 Remotos y Sincronización

git remote

Muestra o gestiona conexiones a repositorios remotos.

```
git remote -v
git remote add origin https://github.com/user/repo.git
```

---

git fetch

Descarga cambios del repositorio remoto sin mezclarlos.

```
git fetch origin
```

---

git pull

Descarga y fusiona los cambios del remoto con tu rama actual.

```
git pull origin main
```

---

git push

Sube tus commits al repositorio remoto.

```
git push origin main
```

Tags y Versionado

git tag

Marca commits con nombres para indicar versiones.

```
git tag v1.0
git tag -a v1.0 -m "Versión estable"
git push origin --tags
```

Flujos y Herramientas Especializadas

git flow

Sistema de ramificación basado en convención: develop, feature/, *release/*, etc.

```
git flow init
git flow feature start nombre
git flow release finish 1.0
```

Automático pero rígido; puedes dejar de usarlo cuando desees.

---

git lfs

Git Large File Storage: permite manejar archivos grandes.

```
git lfs install
git lfs track "*.psd"
```

---

git diff

Muestra diferencias entre archivos o commits.

```

```

```
git diff
git diff main develop
```

---

git patch

Aplica o crea parches de código (.patch files).

```
git format-patch HEAD~1
git apply archivo.patch
```

---

git squash (no es un comando directo)

Combinar varios commits en uno usando rebase interactivo:

```
git rebase -i HEAD~3
# luego cambia "pick" por "squash" en los commits
```

❌ No Existe

git download

No existe como comando oficial. Generalmente se refiere a clonar o hacer fetch / pull.

🧠 Otros Comandos Útiles

- `git clean -fd`: borra archivos no rastreados.
- `git reflog`: historial de movimientos del HEAD, útil para recuperar cambios.

🏷️ Alias en Git

git alias

Permite crear atajos personalizados para comandos de Git, facilitando y agilizando su uso.

Crear un alias temporal (solo para la sesión actual):

```
git config alias.co checkout
git config alias.st status
```

```
git config alias.ci commit
git config alias.br branch
```

Ahora puedes usar, por ejemplo, `git co main` en vez de `git checkout main`.

Crear un alias global (para todos tus repositorios):

```
git config --global alias.lg "log --oneline --graph --decorate --all"
```

Esto te permite ejecutar `git lg` para ver un historial compacto y visual de los commits.

Ver todos tus alias configurados:

```
git config --get-regexp alias
```

Ejemplo de alias útiles:

- `git config --global alias.last "log -1 HEAD"`
- `git config --global alias.unstage "reset HEAD --"`
- `git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"`

> Los alias pueden ser combinaciones de comandos y opciones, pero no pueden reemplazar comandos que requieren argumentos complejos o subcomandos interactivos.

☑ Flujo de Trabajo Recomendado

```
# 1. Inicializas o clonas
git init / git clone

# 2. Trabajas con ramas
git checkout -b feature-x

# 3. Haces cambios
git add .
git commit -m "algo"

# 4. Combinas cambios
git checkout main
git merge feature-x

# 5. Subes cambios
git push origin main
```