**1) ¿Qué es Git? (idea rápida)**

Git es un **sistema de control de versiones distribuido** que guarda el historial de tu proyecto como *instantáneas* (snapshots) cada vez que haces un commit — es muy rápido y permite ramas baratas y trabajo local completo. [git-scm.com](https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F?utm_source=chatgpt.com)

Git es **distribuido**: cada desarrollador tiene una copia completa del historial en su máquina (no depende siempre de un servidor central). Esto hace que operaciones como commit, log o diff sean muy rápidas. [Atlassian](https://www.atlassian.com/git?utm_source=chatgpt.com)

**2) Conceptos clave (breve)**

* **Working directory (o working tree)**: tus archivos tal como están ahora.
* **Staging area / index**: lugar temporal donde pones cambios con git add antes de confirmar.
* **Commit**: snapshot guardado en el historial.
* **Branch (rama)**: puntero a commits; facilita trabajar en features aisladas.
* **Remote (origin)**: repositorio remoto (GitHub, GitLab, etc.).  
  (Estos conceptos están explicados en la documentación oficial y tutoriales). [git-scm.com+1](https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F?utm_source=chatgpt.com)

**3) Instalación rápida**

* macOS: brew install git (si usas Homebrew).
* Ubuntu/Debian: sudo apt update && sudo apt install git
* Windows: instala **Git for Windows** desde su instalador oficial.  
  (Después configura tu identidad global: git config --global user.name "Tu Nombre" y git config --global user.email "tucorreo@example.com").

**4) Primer repo — paso a paso (local → remoto)**

**Crear un repo nuevo local**

mkdir mi-proyecto

cd mi-proyecto

git init

git init crea el repositorio .git y ahora puedes empezar a versionar.

**Añadir archivos y primer commit**

# crea un archivo

echo "# Mi Proyecto" > README.md

# ver estado

git status

# añadir al staging

git add README.md # o `git add .` para todo

# confirmar (commit)

git commit -m "Initial commit"

**Conectar con un remoto (p.ej. GitHub) y subir**

git remote add origin https://github.com/mi-usuario/mi-proyecto.git

git branch -M main # (asegura que la rama principal se llame main)

git push -u origin main

**Clonar un repo existente**

git clone https://github.com/usuario/repo.git

**5)Flujo típico con ramas (feature → main)**

1. Crear rama para la tarea:

git checkout -b feature/mi-feature

# o en Git moderno: git switch -c feature/mi-feature

1. Trabajar, añadir y commitear:

git add archivo1 archivo2

git commit -m "Agregar X funcionalidad"

1. Traer cambios del remoto a main y fusionar:

git checkout main

git pull origin main

git merge feature/mi-feature

git push origin main

Un modelo extendido (Gitflow) organiza main, develop, feature, release y hotfix — es útil en equipos con releases formales. [Atlassian](https://www.atlassian.com/git/tutorials?utm_source=chatgpt.com)

**6) Merge vs Rebase (resumen y cuándo usar)**

* **Merge**: une una rama en otra creando un commit de merge; conserva historia tal como ocurrió (seguro y claro).
* **Rebase**: repliega (mueve) tus commits sobre otra base para crear una historia lineal; útil para limpiar historia, pero **no** hacer rebase de commits que ya compartiste con otros (puede reescribir historial). [Atlassian](https://www.atlassian.com/git/tutorials/merging-vs-rebasing?utm_source=chatgpt.com)

**7) Resolución de conflictos (pasos básicos)**

Si al hacer merge o rebase aparecen conflictos:

1. git status para ver archivos en conflicto.
2. Abre los archivos: verás marcas <<<<<<<, =======, >>>>>>>. Edita para solucionar.
3. git add archivo-resuelto
4. git commit (o git rebase --continue si estabas en un rebase).  
   Consejo: prueba git diff para ver diferencias y git mergetool si configuras una herramienta visual.

**8) Cheatsheet — comandos útiles (rápido)**

* git status — ver cambios.
* git add <file> — poner en staging.
* git commit -m "msg" — crear commit.
* git log --oneline --graph --all — historial compacto.
* git branch / git branch -d nombre — listar / borrar ramas.
* git checkout <branch> o git switch <branch> — cambiar rama.
* git pull — traer y fusionar del remoto.
* git push — enviar commits al remoto.  
  Para una guía de comandos y cheat sheets: recursos oficiales y guías prácticas. [GitHub](https://github.com/git-guides?utm_source=chatgpt.com)

**9)Buenas prácticas rápidas**

* Haz commits pequeños y con mensajes descriptivos.
* Trabaja en ramas por feature/bugfix.
* Antes de push, pull la rama remota para minimizar conflictos.
* No reescribas historial público (evita git rebase en branches ya compartidas).

**10) Recursos recomendados (para seguir aprendiendo)**

* *Pro Git* (Git Book) — explicación de snapshots e internals. [git-scm.com](https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F?utm_source=chatgpt.com)
* Atlassian Git tutorials — workflows, Gitflow y guías paso a paso. [Atlassian](https://www.atlassian.com/git/tutorials?utm_source=chatgpt.com)
* GitHub Guides / cheat sheets. [GitHub](https://github.com/git-guides?utm_source=chatgpt.com)