

# Iniciación a GitHub: Control de versiones y colaboración para principiantes

Esta presentación te guía paso a paso por los conceptos, herramientas y flujos de trabajo esenciales para empezar con Git y GitHub. Aprenderás a instalar y configurar Git, manejar repositorios locales y remotos, trabajar con ramas, colaborar mediante Pull Requests e Issues y adoptar buenas prácticas para proyectos personales o profesionales.





# ¿Por qué usar Git y GitHub?

Git guarda todo el historial de tus archivos, permitiendo volver a versiones anteriores y entender la evolución del proyecto. GitHub añade una capa social y de colaboración: aloja repositorios remotos, registra contribuciones, gestiona permisos y facilita revisiones de código. Incluso equipos ficticios como Hombre Lobo y Drácula pueden trabajar a la vez sin pisarse el trabajo gracias a ramas y merges.

## Control de versiones

Historial completo para deshacer cambios, comparar versiones y recuperar trabajo previo.

## Colaboración

Revisión de cambios, Pull Requests y discusión centralizada en cada contribución.

## Visibilidad

Portafolio público, gestión de proyectos y facilidad para encontrar errores con Issues.

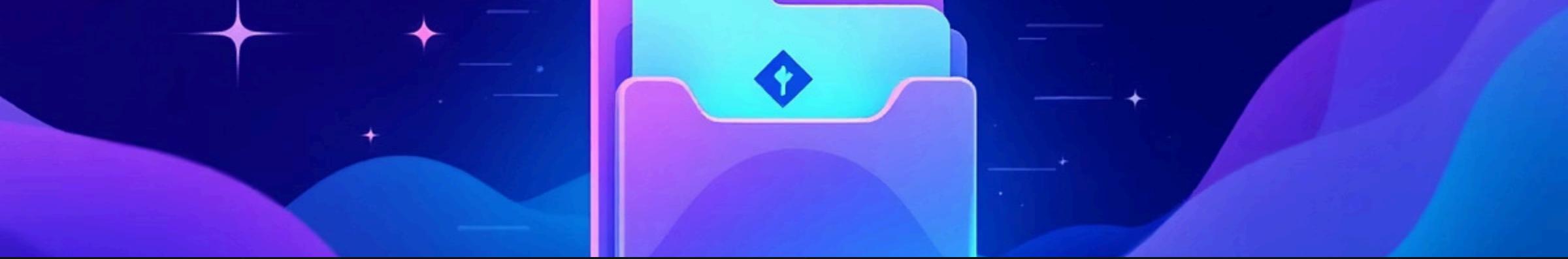


# Instalación y configuración básica de Git

Instala Git desde git-scm.com o mediante el gestor de paquetes de tu sistema (apt, homebrew, winget). Tras la instalación, configura tu identidad global para que los commits reflejen tu autoría:

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

Regístrate en GitHub (<https://github.com>), verifica tu correo y, si vas a usar claves SSH, genera y añade tu clave pública a tu cuenta para evitar introducir la contraseña constantemente.



# Conceptos clave: repositorios y estados de archivos

Un repositorio (repo) es la carpeta que contiene tu proyecto y el historial de Git (.git). Comprender los estados de archivos es fundamental para manejar el flujo de trabajo:



## Untracked

Archivo nuevo que Git aún no controla. Añádelo con git add para empezar a rastrearlo.



## Modified

Archivo modificado en el directorio de trabajo, pero no preparado para el commit.



## Staged

Archivo añadido al área de preparación (staging) y listo para ser commitado.



## Committed

Cambios guardados en el repositorio local con un mensaje que describe la intención.

# Primeros comandos básicos en Git

Estos comandos son la base del flujo de trabajo local. Practícalos con un repositorio de pruebas para interiorizarlos:

- **git init**  
Inicializa un nuevo repositorio Git en la carpeta actual.
- **git status**  
Muestra el estado actual: archivos modificados, staged o sin seguimiento.
- **git add & git commit**

git add <archivo> prepara cambios; git commit -m "mensaje" los guarda en el historial local.

Consejo: escribe mensajes de commit claros y breves que expliquen el "por qué" del cambio.



# Trabajando con repositorios remotos en GitHub

GitHub aloja repositorios remotos que permiten compartir y sincronizar trabajo entre equipos y máquinas. Flujo típico:



## **git clone <curl>**

Copia un repositorio remoto a tu equipo para trabajar localmente.



## **git push origin master**

Sube tus commits locales al repositorio remoto en GitHub (puede ser otra rama distinta a master).



## **git pull origin master**

Trae cambios remotos y los fusiona con tu copia local para mantenerte sincronizado.

Atención a las ramas y posibles conflictos al hacer push o pull; resolverlos pronto evita acumulación de problemas.



# Uso de ramas para trabajar en paralelo

Las ramas permiten desarrollar características aisladas sin afectar la rama principal. Practica este flujo para mantener tu proyecto ordenado y permitir revisiones controladas.



## Crear rama

`git branch <nombre>` — crea una línea de desarrollo separada para una nueva función o corrección.



## Cambiar de rama

`git checkout <rama>` — mueve tu área de trabajo a esa rama para empezar a trabajar.



## Fusionar

`git merge <rama>` — integra los cambios de una rama en otra; resuelve conflictos si aparecen.

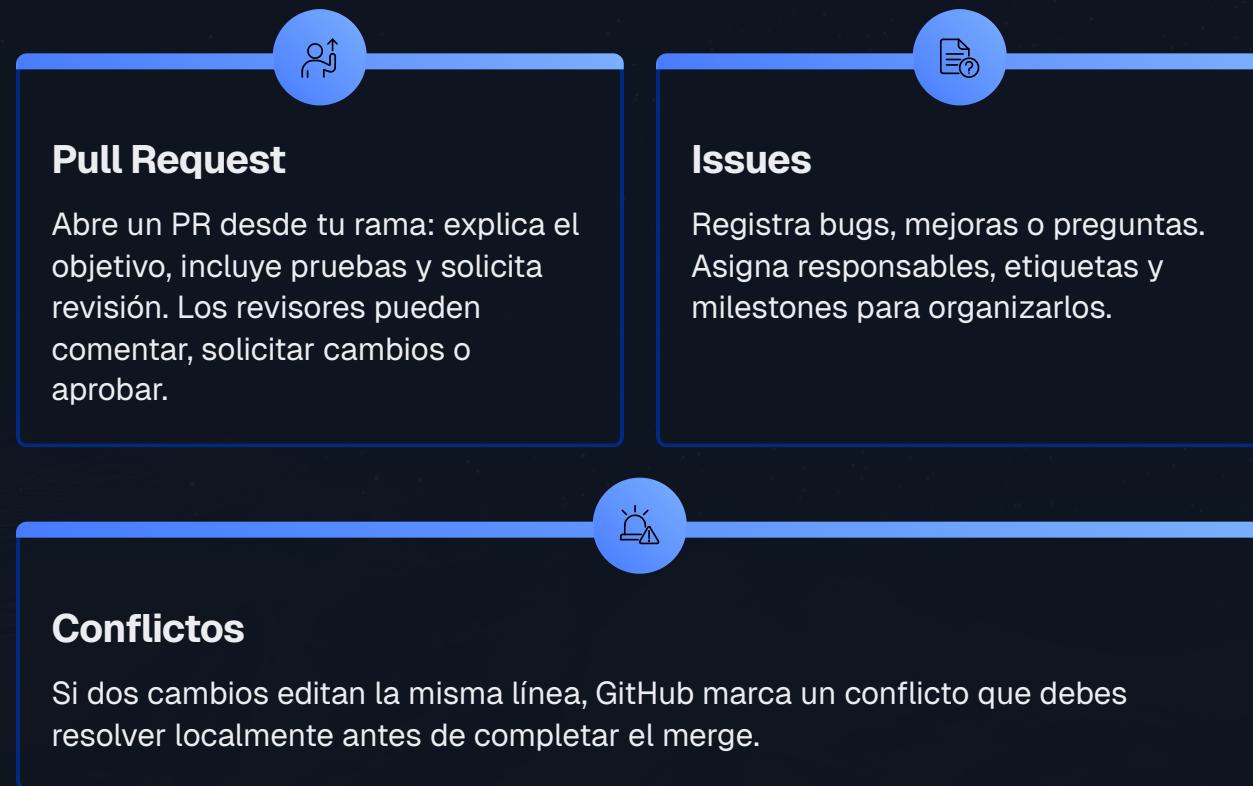


## Eliminar rama

`git branch -d <rama>` — borra ramas ya integradas para mantener el repo limpio.

# Colaboración en GitHub: Pull Requests e Issues

GitHub facilita la colaboración estructurada. Usa Issues para reportar tareas o bugs y Pull Requests (PR) para proponer cambios que serán revisados antes de integrarse en la rama principal.





# Buenas prácticas para principiantes

Adoptar hábitos sólidos desde el principio facilitará tu trabajo y la colaboración con otros. Estas prácticas mejoran la trazabilidad y reducen errores.



## Commits frecuentes

Haz commits pequeños y regulares con mensajes descriptivos que expliquen la intención del cambio.



## Ramas por tarea

Crea una rama para cada nueva funcionalidad o corrección; esto aísla el trabajo y facilita la revisión.



## Pruebas y revisión

Prueba localmente, añade tests mínimos y solicita revisiones antes de fusionar cambios a la rama principal.

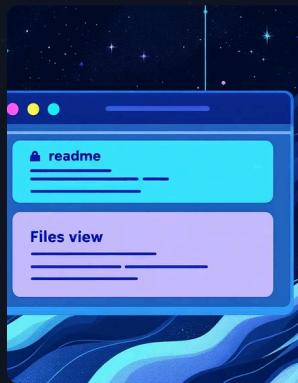
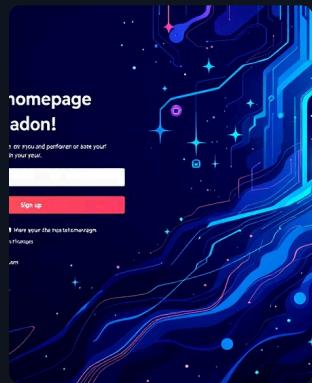


## Documentación

Mantén un README claro, añade instrucciones de instalación y uso, y documenta decisiones importantes en el repo.

# ¡Empieza hoy con GitHub!

Crea tu cuenta en <https://github.com> y practica con un repositorio sencillo: inicialízalo, añade archivos, realiza commits, crea ramas, sube cambios y abre un Pull Request. Explora Issues y la documentación oficial. Con paciencia y práctica, Git y GitHub se convertirán en herramientas esenciales que mejorarán tu productividad y tu capacidad para colaborar en proyectos reales.



## 1. Regístrate

Crea cuenta y configura tu perfil: foto, bio y repositorios públicos para mostrar tu trabajo.

## 2. Practica

Haz un proyecto pequeño: añade código, README y realiza commits; prueba push/pull con remoto.

## 3. Colabora

Contribuye a proyectos de otros mediante forks y Pull Requests; aprende leyendo revisiones y comentarios.

- Colores recomendados para énfasis: usa #6296ffff como color principal en títulos y CTAs; aplica los colores secundarios para etiquetas y cajas informativas.