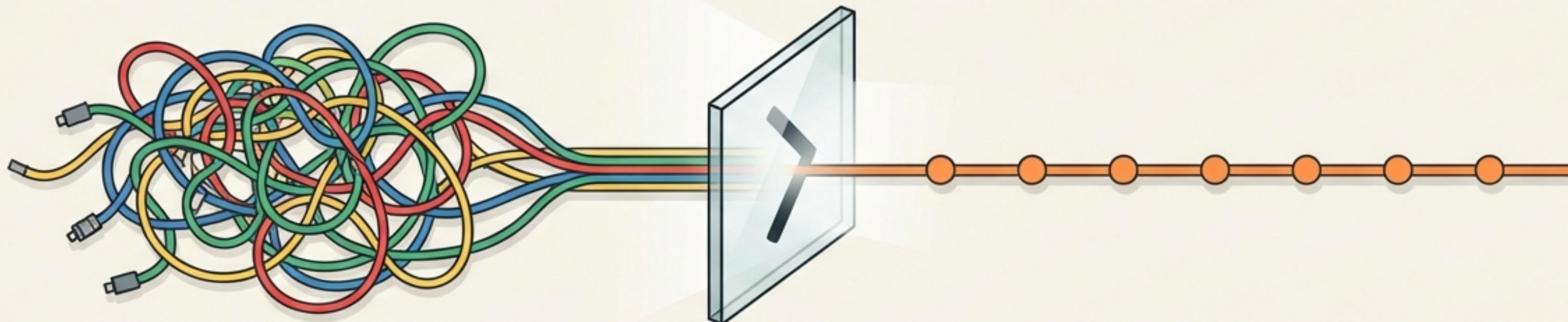


# Introducción a Git: Control de Versiones desde Cero

De `final\_final.zip` a una máquina del tiempo para tu código.



Caos de Archivos

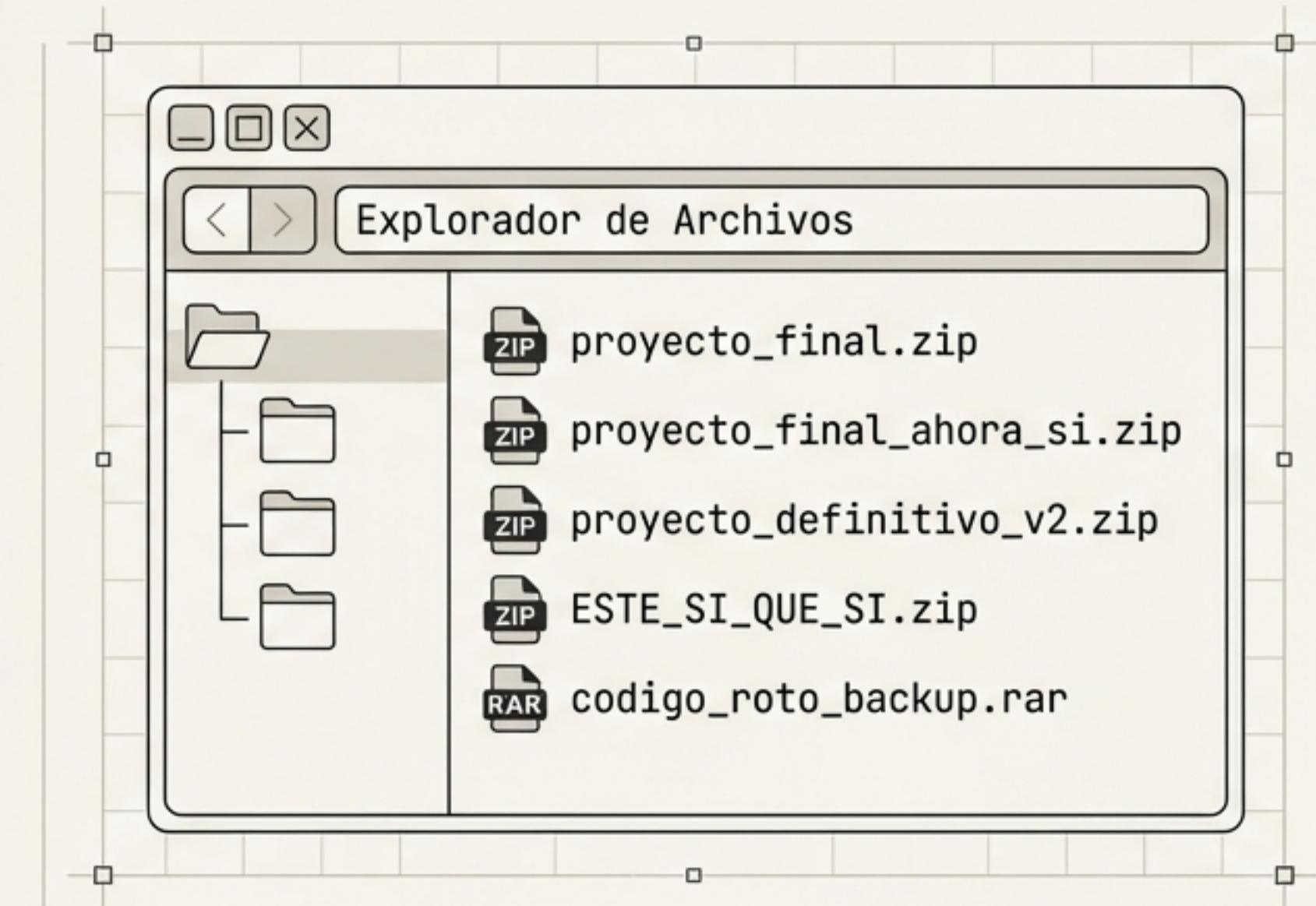
Historial de Commits Limpio

Guía esencial para principiantes

# El problema: El caos de las versiones manuales

Sin un sistema de control, programar se convierte en un juego de azar:

- 'Antes funcionaba... ¿qué he tocado?'
- 'He borrado algo importante sin querer y no puedo recuperarlo.'
- 'He mezclado cambios y ya no sé separar lo que hice.'

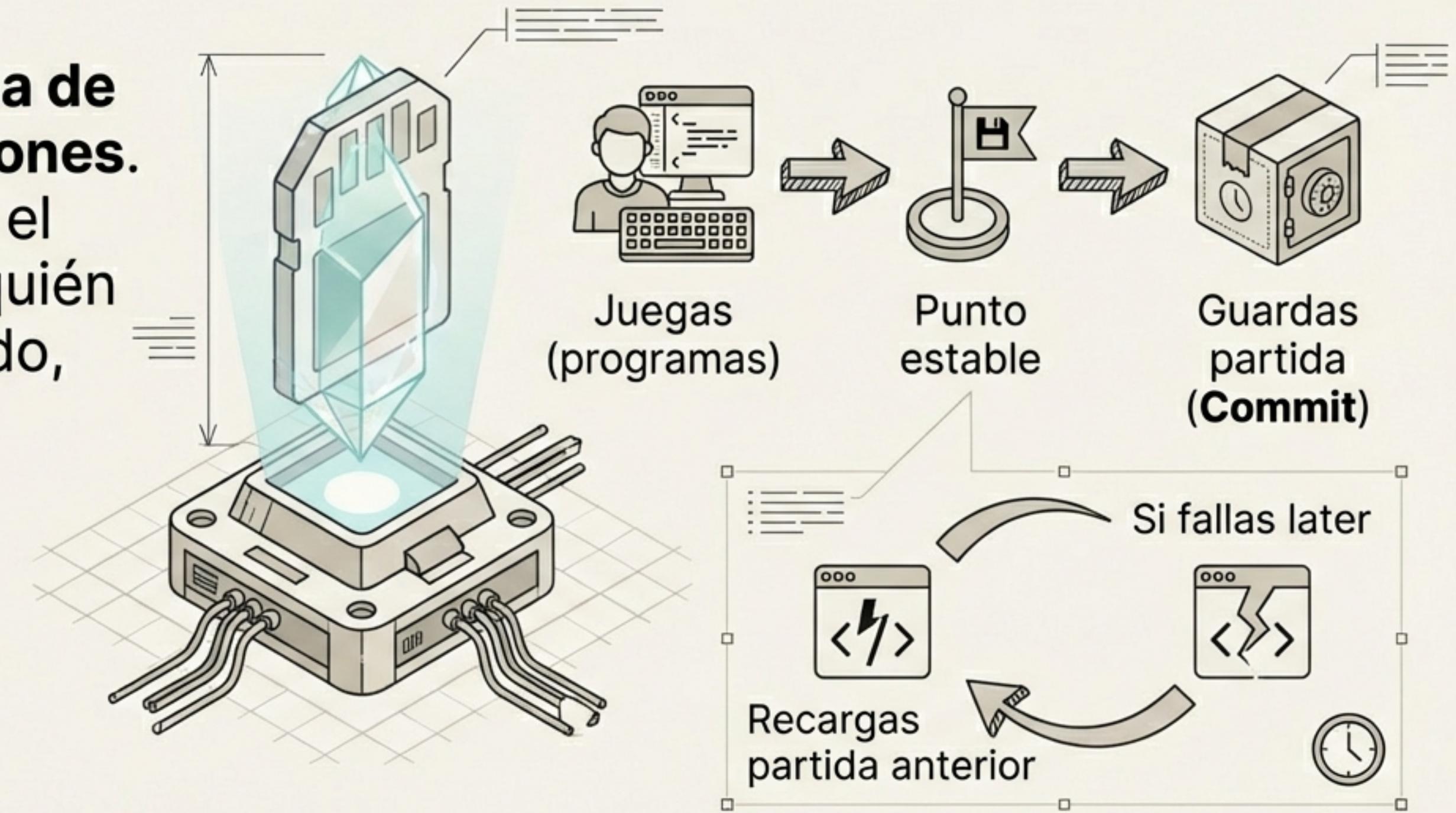


**Necesitamos un historial ordenado y recuperable.**

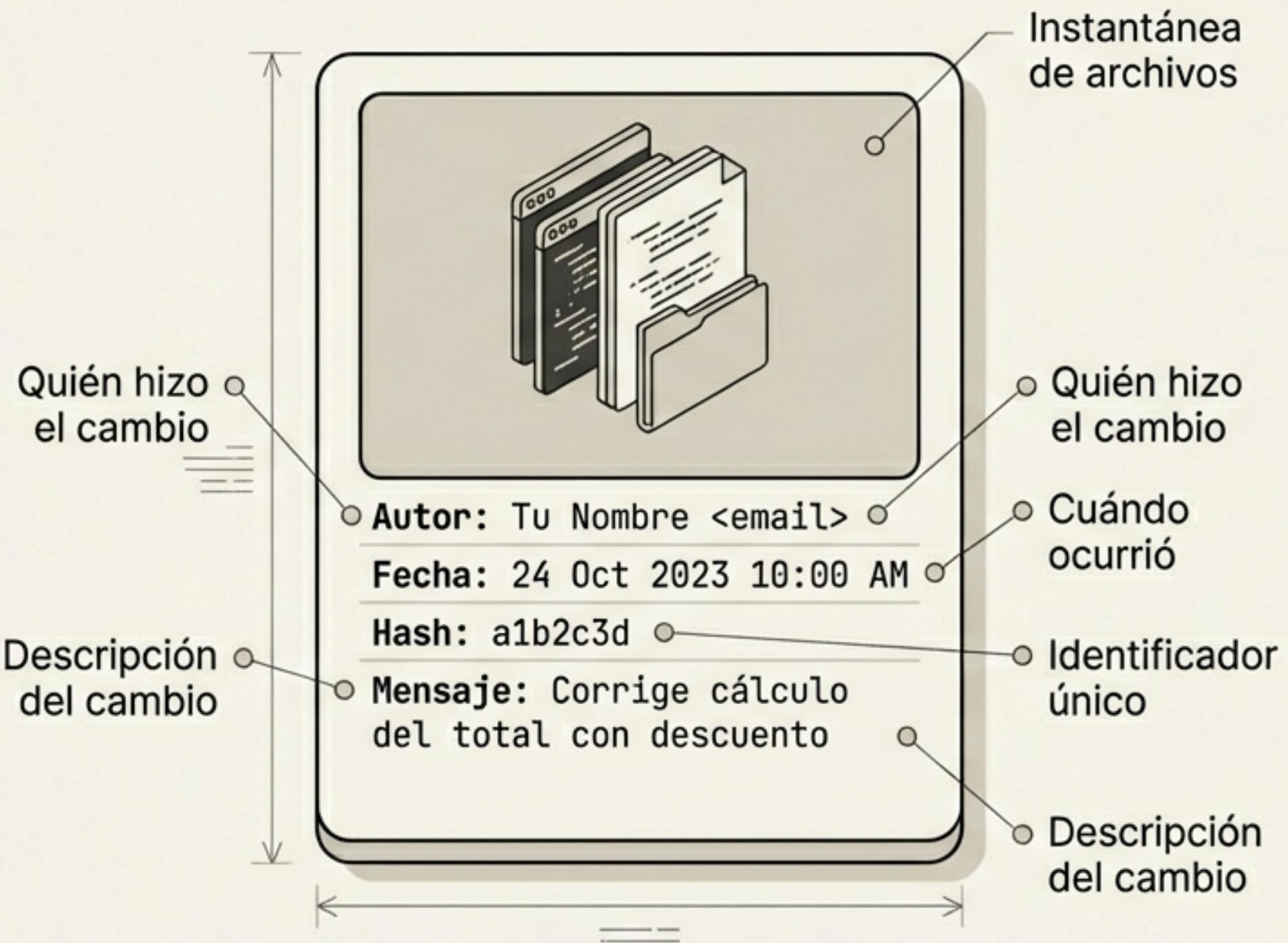
# Git es una máquina del tiempo para tu proyecto

Git es un **sistema de control de versiones**.

Sirve para llevar el historial, saber quién hizo qué y cuándo, y volver atrás si algo se rompe.



# El Commit: Una foto de tu proyecto en el tiempo



Un commit no es solo guardar cambios; es una "foto" completa del estado de los archivos en ese momento.

**Consejo Pro:** El mensaje debe decir QUÉ has hecho y PARA QUÉ.

- Bien: "Corrige cálculo del total con descuento"
- Mal: "Arreglos varios"

# El modelo mental: Escritorio, Bandeja y Álbum

Git funciona en pasos. No guarda todo a la vez.

**Escritorio**  
(Working Directory)



Trabajo/Edición

**Bandeja**  
(Staging Area)



Selección

**Álbum**  
(Historial)



Escritorio → Bandeja (preparar)

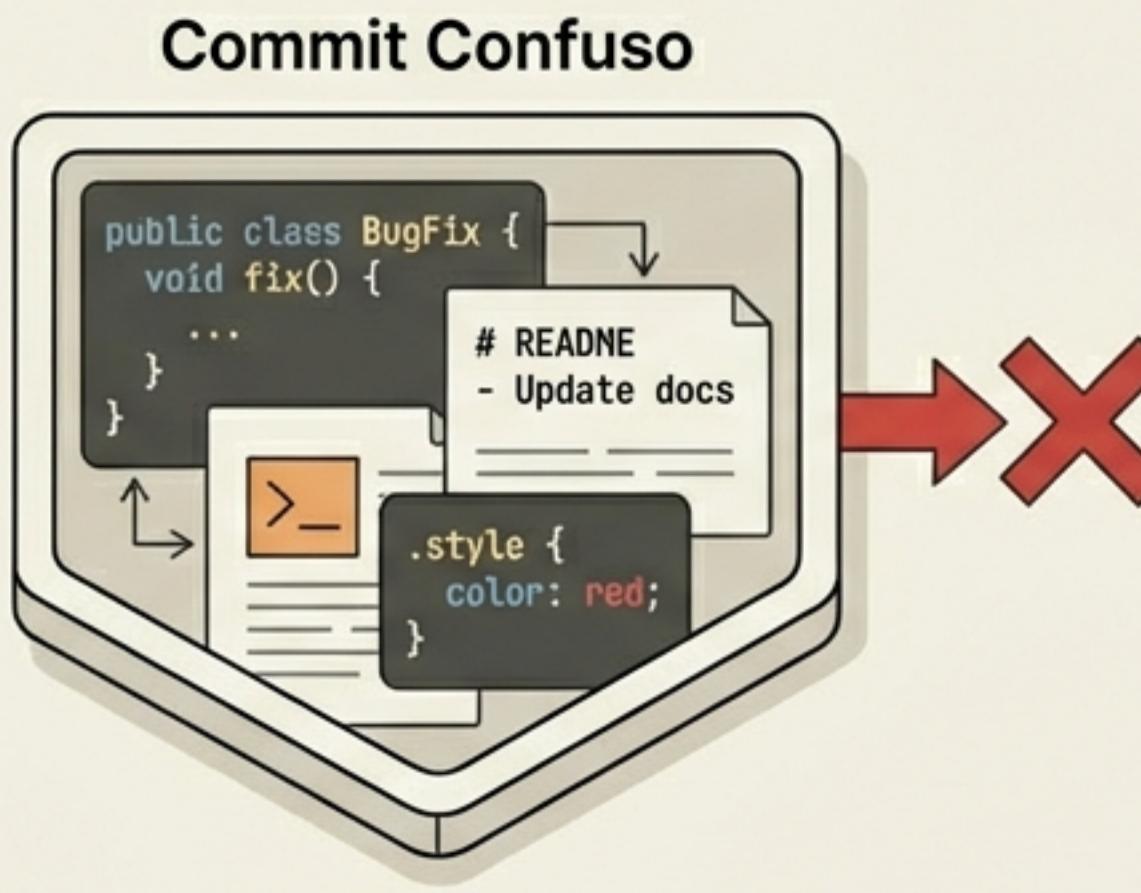
Bandeja → Álbum (guardar)

Versiones definitivas

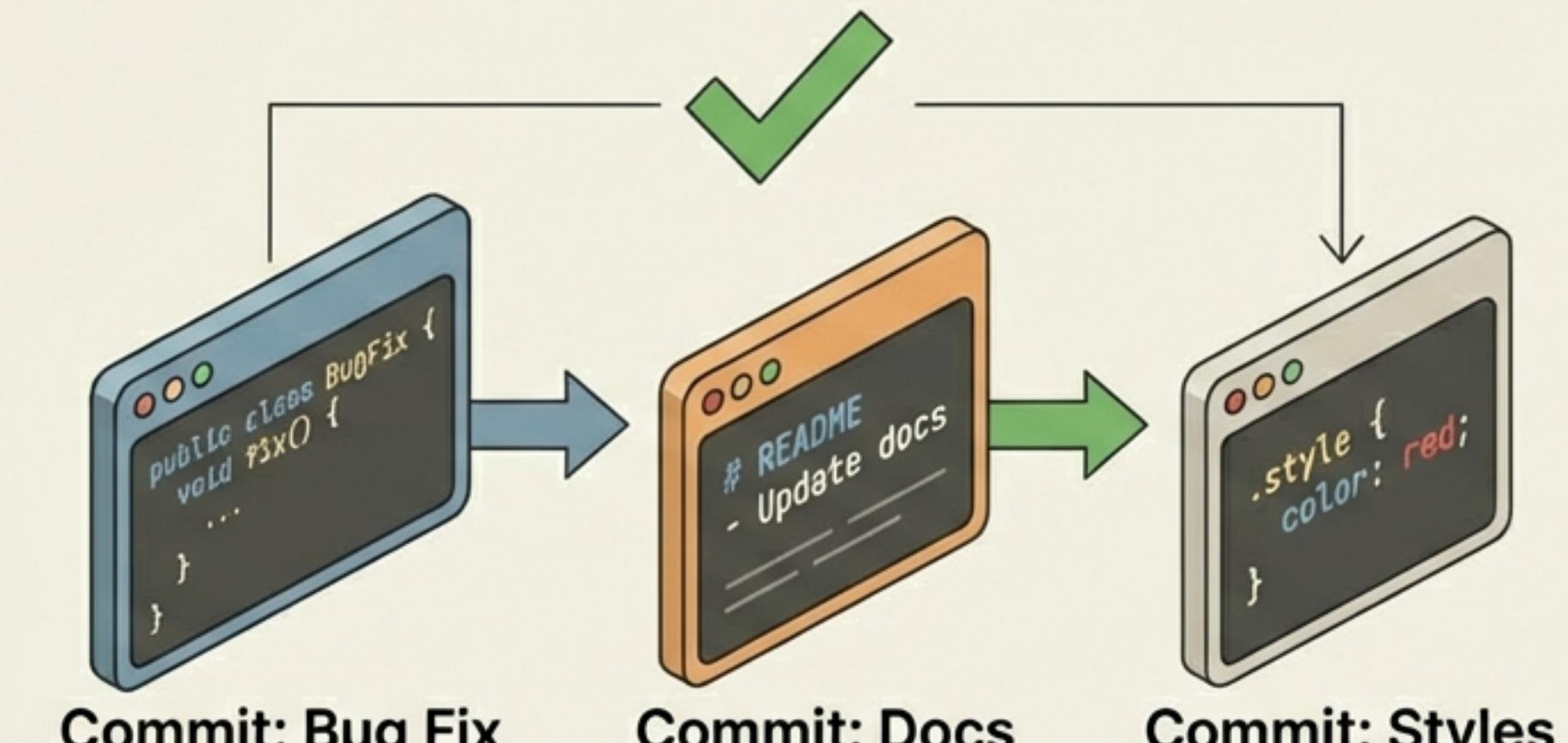
# ¿Por qué necesitamos la "Bandeja"?

**Staging = Elegir.** Permite separar cambios para tener un historial limpio.

Situación: Has tocado 3 cosas a la vez (Bug Java, Texto README, Estilo CSS).



Batiburrillo

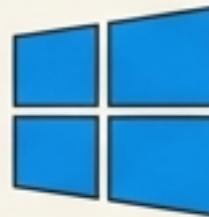


Historial claro y reversión fácil "por partes".

# Paso 1: Instalación y Verificación

```
$ git --version  
git version 2.39.1
```

- ✓ Si devuelve una versión, ya lo tienes.
- ✗ Si dice 'command not found', instálalo.



**Windows**

Instala **Git for Windows**.  
Usa **Git Bash** (evita  
PowerShell al principio).



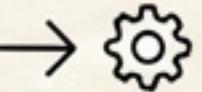
**macOS**

Si `git --version` falla,  
acepta instalar las  
**Command Line Tools**.



**Linux**

Ejecuta:  
`sudo apt install git`



# Paso 2: Configura tu identidad

Git necesita saber quién eres para firmar tus 'fotos' (commits). Esto se hace una sola vez.

```
# Configura tu nombre  
git config --global user.name "Tu Nombre"  
  
# Configura tu email  
git config --global user.email "tu@email.com"  
  
# Configuración extra recomendada  
git config --global init.defaultBranch main  
git config --global core.editor "code --wait"
```



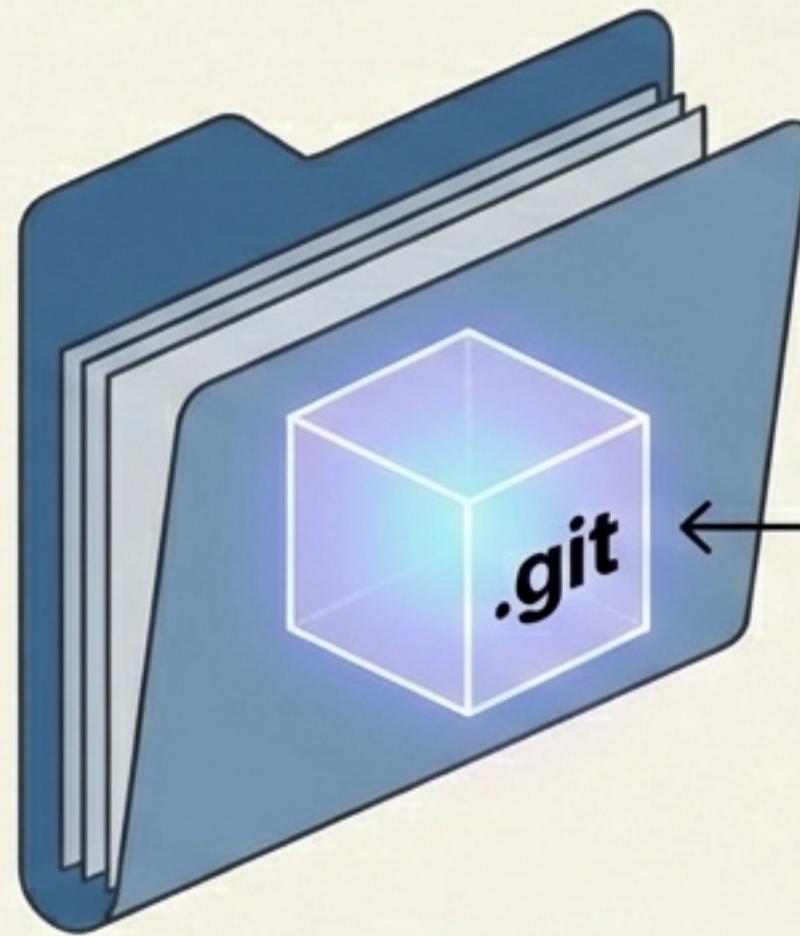
Si ves el error '**Please tell me who you are**', es que falta configurar esto.

# Creando un repositorio: `git init`

Convierte una carpeta normal en un proyecto con 'superpoderes' Git.

Navega a tu carpeta y ejecuta:

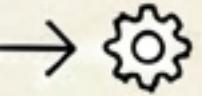
```
$ git init
```



El cerebro del proyecto  
(Carpeta Oculta).



**PELIGRO:** Nunca borres la carpeta '.git'  
o perderás todo el historial. Evita usar  
OneDrive/Google Drive para repositorios.



# El Navegador: `git status`

Regla de Oro: Siempre mira el status antes de hacer nada.

```
$ git status  
On branch main
```

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)  
new file: archivo\_listo.txt ←

EN BANDEJA  
(Verde)  
Staged

Untracked files:

(use "git add <file>..." to include in what will be committed)  
archivo\_nuevo.txt ←

EN ESCRITORIO  
(Rojo)  
Untracked

# Inspeccionando cambios: `git diff`

Verifica exactamente qué ha cambiado línea a línea antes de guardar.

```
diff view
1 function calcularTotal(precio) {
2 - return precio * 1.15;
3 + return precio * 1.21; // IVA actualizado
4 }
```

Rojo (-) = Lo que borraste.

Verde (+) = Lo que añadiste.

# Llenando la bandeja: `git add`

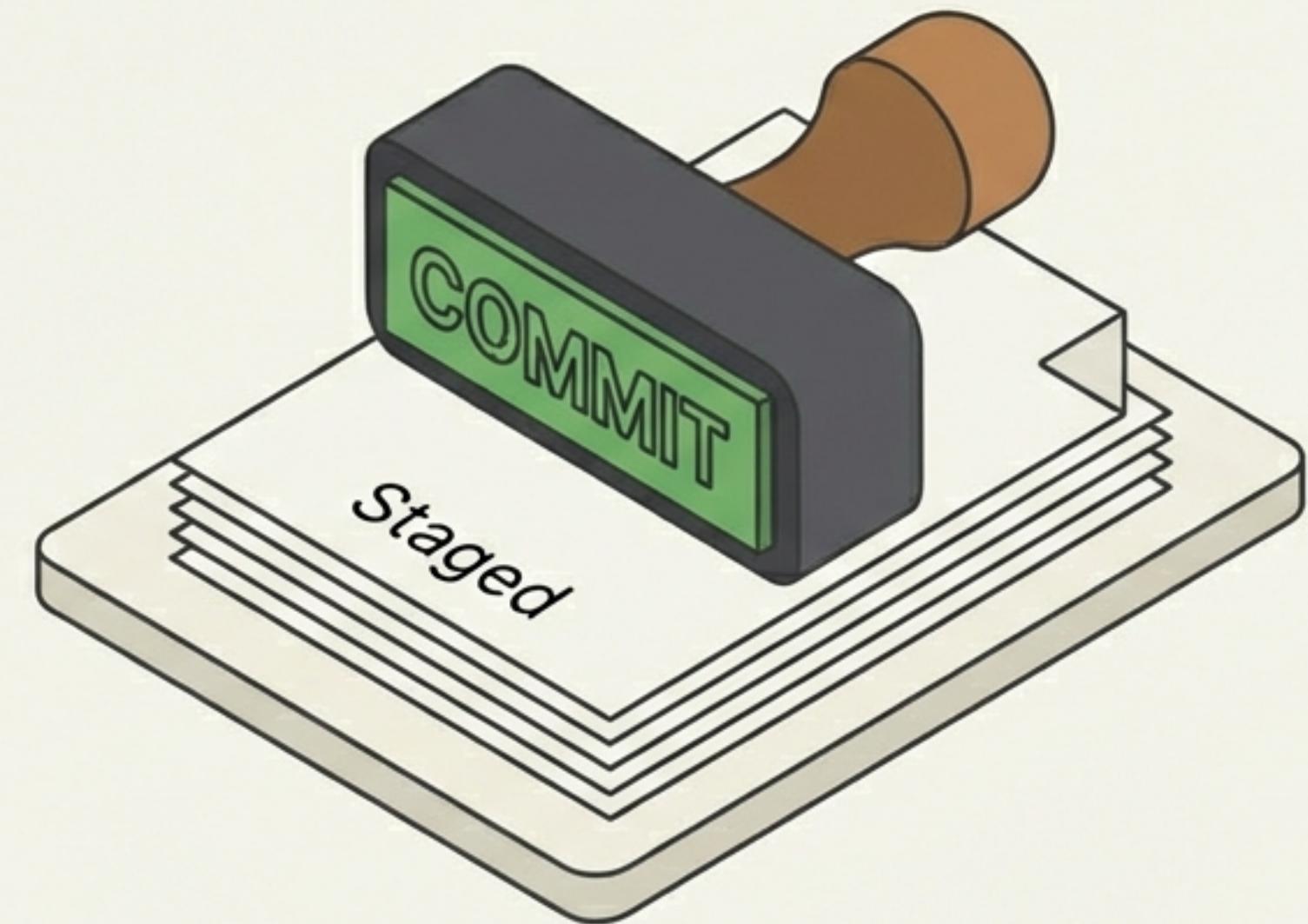
Mueve archivos del 'Escritorio' (Untracked/Modified) al 'Staging' (Bandeja).



⚠ Cuidado con `git add .` (añade todo). Al principio, añade archivo por archivo para tener control.

# Guardando la partida: `git commit`

Pega lo que hay en la 'Bandeja' en el 'Álbum' definitivo.



```
$ git commit -m "Mensaje descriptivo"
```

⚠ Key Insight: Este comando **solo** guarda lo que estaba en Staging. Los archivos Untracked se ignoran.

# Revisar y Restaurar: `git log` y `git restore`

## Ver Historial

```
$ git log --oneline
```



a1b2c3d - Fix bug

9f8e7d6 - Add README

## Arreglar errores

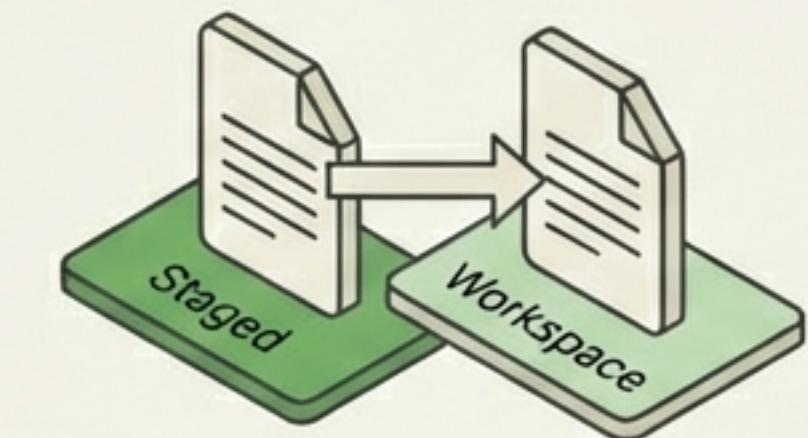
Danger Zone

```
$ git restore archivo
```



Safe Zone

```
$ git restore --staged archivo
```



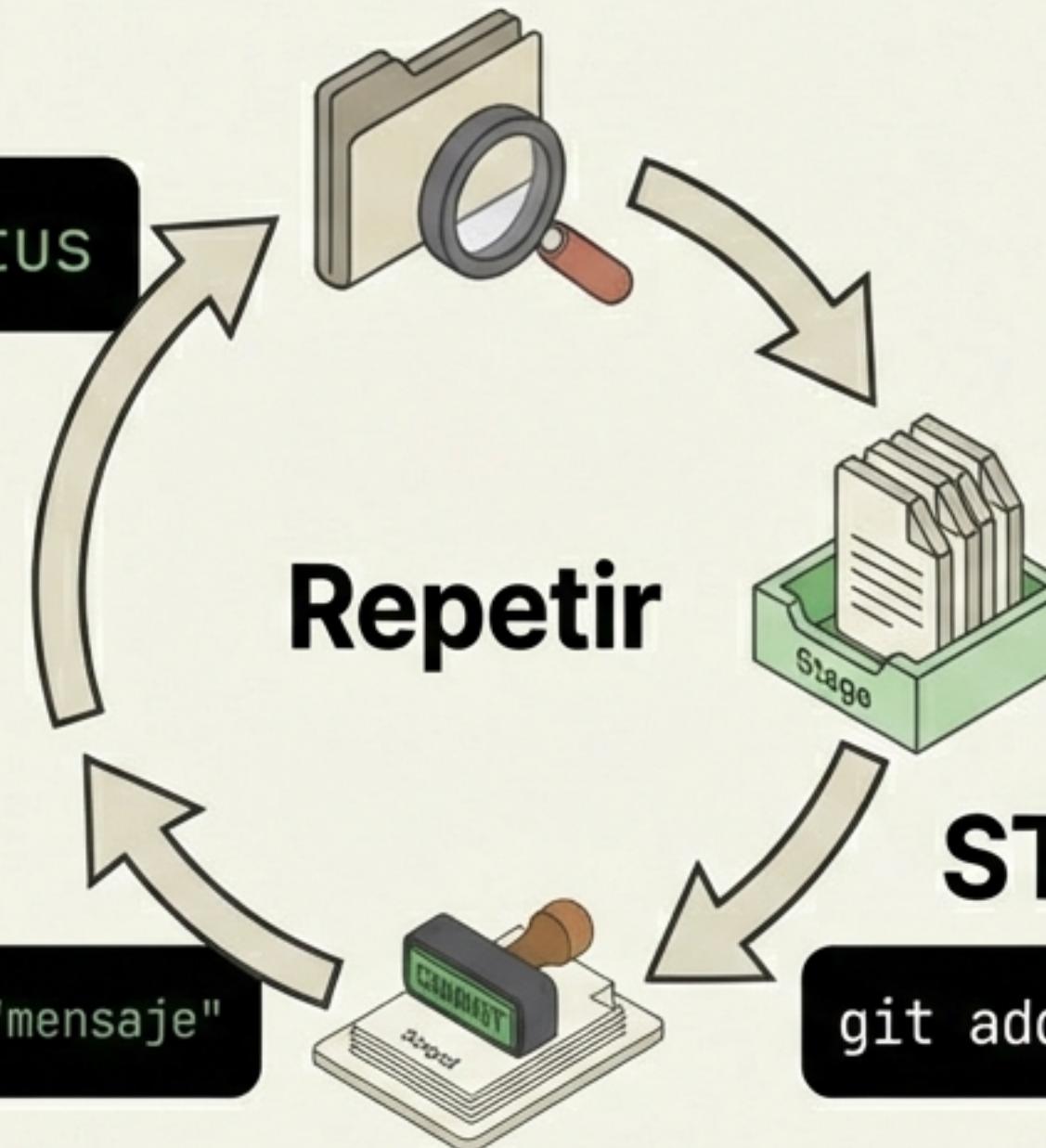
Descarta tus cambios y vuelve al último commit. **Irreversible**.

Saca el archivo de la bandeja, pero **mantiene** tus cambios en el escritorio.

# Tu flujo de trabajo diario (Cheat Sheet)

## CHECK

```
$ git status
```



## Comandos Útiles

- `git init`: Empezar repo
- `git diff`: Ver cambios
- `git log`: Ver historia
- `git restore`: Deshacer

⚠ Recuerda: Staging es elegir. Trabaja con **orden**.