

# GitHub: repositorios remotos, clonación y sincronización

Alba García Naharro, Aida  
González Fernández y Maxim Raúl  
Huerta Martínez



2025



# Índice

Presentación GitHub

## 01 Introducción a GitHub

Qué es, para qué sirve y por qué es tan útil

## 02 Repositorios y comandos básicos

Cómo se organiza el código y qué comandos usar

## 03 Colaboración y sincronización

Clonación, trabajo en equipo y resolución de conflictos

## 04 Buenas prácticas y conclusión

Consejos clave y cierre motivador



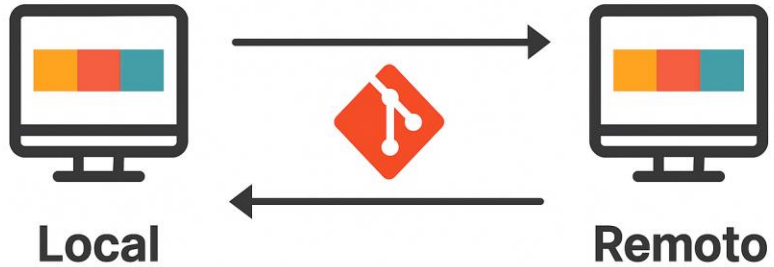
01

## ¿Qué es GitHub?

- Plataforma esencial para desarrolladores.
- Permite guardar, compartir y sincronizar código.
- Facilita el trabajo colaborativo con control de versiones.
- Cada cambio queda registrado automáticamente.

## ¿Qué es GitHub y para qué sirve?

- Repositorio remoto = espacio en la nube para código e historial.
- Almacena qué se modificó, quién y en qué momento.
- Repositorios públicos: cualquiera puede ver y colaborar.
- Repositorios privados: acceso restringido a miembros.
- Promueve el trabajo colaborativo y sincroniza los cambios.

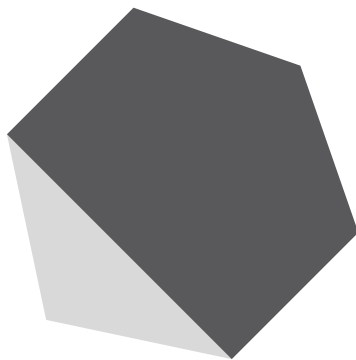


# ¿Qué es un repositorio?

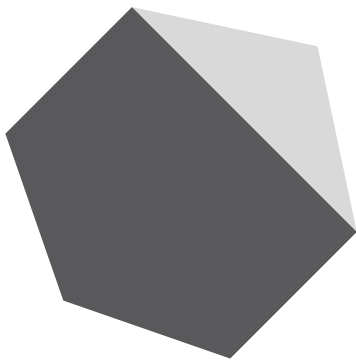
- Carpeta inteligente que guarda código, archivos e historial de cambios.



- Registra autor, fecha y motivo de cada modificación.



- Local (Git) vs Remoto (GitHub): trabajo en equipo desde la nube.



- Permite restaurar versiones sin pérdida de datos.



**GitHub facilita el trabajo en equipo manteniendo el código organizado y actualizado.**

# Comandos esenciales de Git

## Configuración

git config --global user.name  
git config --global user.email

git init (local)  
git clone <url> (remoto)  
git remote add origin <url>

## Clonación y actualización

git log  
git diff  
git reset soft/hard

## Inicialización y Conexión

git clone <url>  
git add .  
git commit -m "Texto  
descriptivo"  
git push origin  
main/master

## Historial y control

Categoría	Comando	Función
Configuración	<ul style="list-style-type: none"><li>git config --global user.name</li><li>git config --global user.email</li><li>git config --global color.ui</li><li>git config --list</li></ul>	Identifica al usuario
Inicialización	<ul style="list-style-type: none"><li>git init</li><li>git clone &lt;url&gt;</li></ul>	Crea un nuevo repositorio local
Conexión	<ul style="list-style-type: none"><li>git remote add origin &lt;url&gt;</li><li>git remote -v</li></ul>	Conecta el repo local con GitHub
Clonación	<ul style="list-style-type: none"><li>git clone &lt;url&gt;</li></ul>	Copia un repo remoto en tu equipo
Actualización	<ul style="list-style-type: none"><li>git add .</li><li>git commit -m "Texto que identifique qué se hizo"</li><li>git push origin main/master</li></ul>	Sube los cambios
Historial y control	<ul style="list-style-type: none"><li>git log</li><li>git status</li><li>git diff</li><li>git reset</li><li>git checkout --&lt;file&gt;</li><li>git merge &lt;nameBranch&gt;</li><li>git stash #</li><li>git stash pop #</li></ul>	Revisa, compara o deshace cambios

# Comandos esenciales de Git

```
C:\Users\albag\apuntes-1DAM>git status
On branch master
nothing to commit, working tree clean

C:\Users\albag\apuntes-1DAM>git commit -m "Ampliar apuntes de Programación y
vocabulario de inglés"
On branch master
nothing to commit, working tree clean

C:\Users\albag\apuntes-1DAM>
```

## Historial de commits

Aquí podeis observar ejemplos de commits

```
C:\Users\albag\apuntes-1DAM>git log
commit b42ffa5f87e4b608a50c0d09f68d2f58cc6eb898 (HEAD -> master)
Author: Alba García <albagn34@educastur.es>
Date: Tue Sep 23 13:51:52 2025 +0200
```

Añadir apuntes de Bases de datos con error

```
commit bc982bc0cb9bc78a87bda0cf3a8b5ecb84f2d53f
Author: Alba García <albagn34@educastur.es>
Date: Tue Sep 23 13:31:03 2025 +0200
```

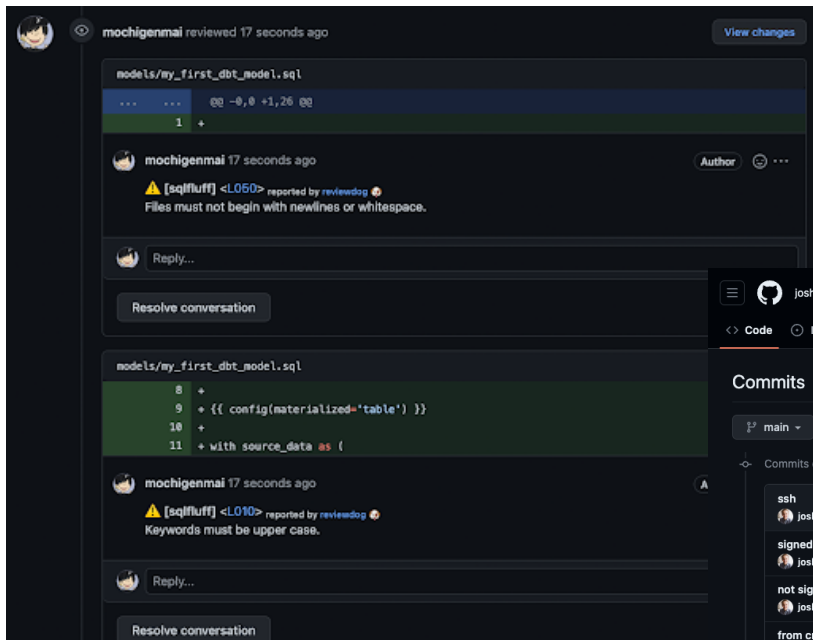
Añadir apuntes iniciales de programación e inglés

```
commit ddaf26bb1efc83243b70c5e2ff8f2c27670129f6
Author: Alba García <albagn34@educastur.es>
Date: Tue Sep 23 13:28:16 2025 +0200
```

Estructura inicial de carpetas y readme.md

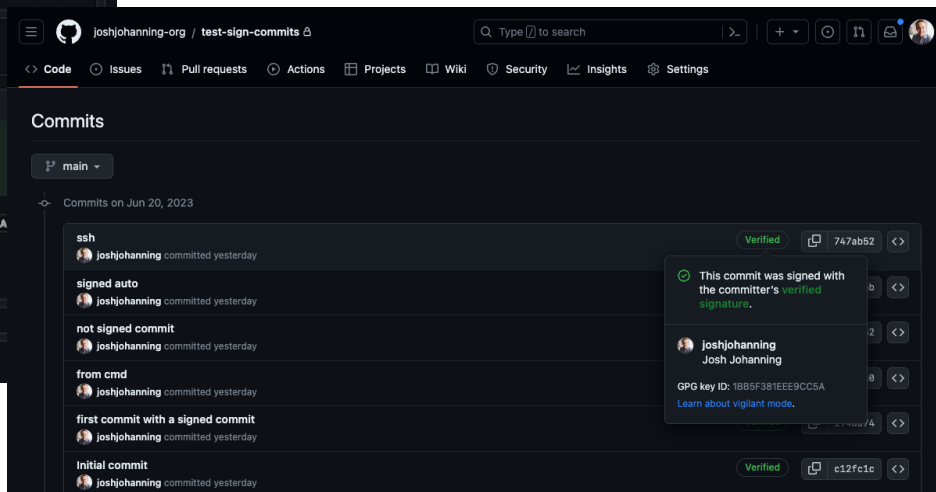
```
C:\Users\albag\apuntes-1DAM>
```

# Comandos esenciales de Git



## Historial de commits

Aquí podeis observar exemplos de commits





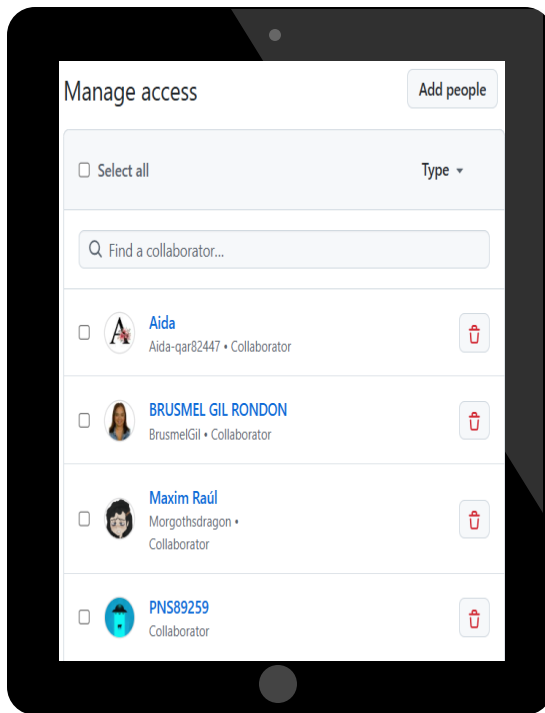
# Trabajo colaborativo

1.

Ir a Settings -->  
Collaborators.

3.

Clonar con ***git clone***



2.

Agregar usuario y  
aceptar invitación

4.

Sincronizar con ***git***  
***pull*** y ***git push***

Permite colaboración sin interferencias



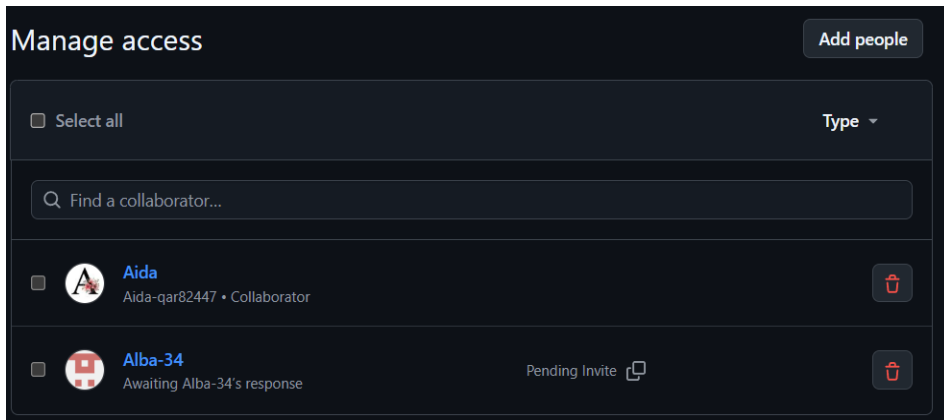
04

## Trabajo colaborativo, sincronización y clonación

# Clonación y sincronización

## ¿Como sincronizo mi repo con otro usuario?

Adentro de tu repo en GitHub, en ajustes y colaboradores, tienes que añadir el usuario y el otro debe de aceptar la invitación.



## ¿Y cómo lo clono?

Para clonar un repo de github se tiene que usar el siguiente comando:

Git clone (link del repo)

```
sidse@Vr1l MINGW64 ~  
$ git clone https://github.com/Morgothsdragon/repo-para-ejercicio  
Cloning into 'repo-para-ejercicio'...  
warning: You appear to have cloned an empty repository.  
sidse@Vr1l MINGW64 ~
```

Y así al hacer comits podrás hacerles push para que se suba de forma automática al repo



04

# Sincronización y resolución de problemas

# Resolución de conflictos

Cuando varios usuarios trabajan en el mismo repositorio, puede ocurrir que **dos personas editen el mismo archivo al mismo tiempo**.

Imagina que tú y otro compañero modifican la misma línea de código en momentos distintos.

Cuando intentas subir tus cambios con git push, GitHub detecta que el archivo también fue modificado por otra persona y te muestra un **conflicto**.

Estos conflictos son completamente normales en el trabajo colaborativo. GitHub no los resuelve automáticamente porque necesita que el equipo decida **qué versión del archivo conservar**.

**Siempre haz un git pull antes de hacer un git push.**

De esta forma, te aseguras de tener los cambios más recientes del repositorio antes de subir los tuyos, evitando conflictos innecesarios y manteniendo el proyecto limpio y sincronizado

Para solucionarlo, existen dos opciones:



## Resolverlo manualmente

editando el archivo y eligiendo qué fragmentos de código mantener.



## Usar herramientas visuales

O usar **herramientas visuales** que ofrece GitHub o editores como Visual Studio Code, que facilitan comparar las diferencias entre versiones.

Lo más importante para evitar estos problemas es mantener una **buena comunicación** con el equipo y seguir una regla básica:

# Consejos

01

Mensajes de commit claros.



02

Usa ramas (branch) para nuevas funciones.



03

Haz pull requests antes de fusionar cambios.



04

Sincroniza con frecuencia tu repositorio.





04

# Conclusión



## Conclusión

En resumen, GitHub no solo almacena código, sino que permite **trabajar en equipo de forma ordenada y segura**.

Gracias a los repositorios remotos, la clonación y la sincronización, los equipos pueden colaborar desde cualquier lugar y mantener su proyecto siempre actualizado.

Te invitamos a probarlo: crea tu primer repositorio y experimenta con tus compañeros.





# Gracias

Alba García Naharro, Aida  
González Fernández y Maxim Raúl  
Huerta Martínez

Os dejamos un cuestionario para que repaseis los  
conceptos: [Test de autoevaluación de GitHub](#)



2025