

# Git Básico

---

**Arturo Silvelo**

Try New Roads

# Trabajando Local

# Git de forma local

---

## Iniciar un nuevo proyecto

---

Para crear un nuevo proyecto en Git, usa el comando:

```
git init <nombre del proyecto>
```

Si ya tienes un directorio creado y deseas convertirlo en un repositorio de Git, navega a él con:

```
cd <directorio>  
git init
```

```
silvelo@silvelo-Inspiron-7559 ~/repos > git init course-example
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/silvelo/repos/course-example/.git/
```

En ambos casos, Git crea una rama principal por defecto y el directorio `.git` se genera para almacenar toda la información del proyecto.

Para comprobar si tu proyecto tiene un repositorio inicializado, puedes usar el comando:

```
git status
```

```
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > git status  
On branch master  
  
No commits yet  
  
nothing to commit (create/copy files and use "git add" to track)
```

## Directorio de trabajo

---

El **directorio de trabajo** es la carpeta donde tienes todos los archivos y en la que has iniciado tu repositorio.

Creamos un nuevo archivo con el comando:

```
touch index.html
```

Luego, revisamos el estado del repositorio con:

```
git status
```



Esto mostrará que el archivo `index.html` ha sido añadido y está en estado **modificado**.

Para obtener una vista más simplificada, puedes usar:

```
git status -s
```

```
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > touch index.html
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    index.html

nothing added to commit but untracked files present (use "git add" to track)
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > git status -s
?? index.html
```

# Staging

---

## Añadir Staging

---

El área de `staging` es una zona temporal donde preparamos los archivos modificados antes de confirmarlos con un `commit`.

```
git add index.html          # Prepara el archivo index.html
git add archivo1.js archivo2.js # Prepara varios archivos
git add *.js                # Prepara todos los archivos .js
git add -A                  # Prepara todos los cambios (incluyendo eliminaciones)
git add .                   # Prepara todos los cambios en el directorio actual
git add resources/          # Prepara todos los archivos en el directorio resources
```

```
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > echo "HOLA MUNDO.txt" >> texto.txt
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > cat texto.txt
HOLA MUNDO.txt
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > git add texto.txt
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > git status -s
A   texto.txt
??  index.html
```

## Sacar archivos de Staging

Podemos eliminar los ficheros del área de `staging` y devolverlos al estado de modificados con el comando `git reset`.

```
git reset index.html          # Elimina index.html del área de staging
git reset archivo1.js archivo2.js  # Elimina varios archivos del área de staging
git reset *.js                # Elimina todos los archivos .js del área de staging
git reset -A                  # Elimina todos los archivos del área de staging
git reset .                    # Elimina todos los cambios del directorio actual del área de staging
git reset resources/           # Elimina todos los archivos del directorio resources del área de staging
```

```
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > echo "HOLA MUNDO 2.txt" >> texto2.txt
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git status -s
?? index.html
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git add texto2.txt
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git status -s
A texto2.txt
?? index.html
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git reset texto2.txt
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git status -s
?? index.html
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > 
```

# Commit

---



## ¿Qué es un commit?

---

Los commits sirven para registrar los cambios que se han producido en el repositorio. Cada commit muestra el estado de todos los archivos del repositorio, el autor, la fecha y otra información útil.

## ¿Cómo hacer un commit?

---

Para guardar los ficheros del área de `staging`, se utiliza el comando:

```
git commit    # Este comando creará una referencia al commit
```

Este comando abrirá el editor para que puedas poner un mensaje de commit. Si quieres añadir el mensaje directamente en el comando, puedes usar la opción `-m`:

```
git commit -m 'new feature'    # Realiza un commit con el mensaje 'new feature'
```

```
silvelo@silvelo-Inspiron-7559 ~/..📁./course-example 🍷 master > git status -s
A texto.txt
?? index.html
silvelo@silvelo-Inspiron-7559 ~/..📁./course-example 🍷 master > git commit -m 'Nuevo fichero'[master (root-commit) 64614f7] Nuevo fichero
1 file changed, 1 insertion(+)
create mode 100644 texto.txt
silvelo@silvelo-Inspiron-7559 ~/..📁./course-example 🍷 master > git status -s
?? index.html
silvelo@silvelo-Inspiron-7559 ~/..📁./course-example 🍷 master > git log
commit 64614f7a855aec9571d03259eea4daf6e38e8940 (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:10:43 2025 +0200

    Nuevo fichero
```

## Commit sin staging

---

También es posible evitar añadir directamente los archivos modificados al área de `staging`. Para realizar esta operación se utiliza el comando:

```
git commit -a    # Realiza un commit de todos los archivos modificados sin necesidad de añadirlos a staging
```

Este comando realizará un commit directamente de los archivos modificados. Además, se puede añadir la opción `-m` para incluir el mensaje de commit directamente:

```
git commit -am 'new feature'    # Realiza un commit con el mensaje 'new feature' sin pasar por staging
```

**Nota:** Esto sólo funciona para archivos modificados. Los archivos nuevos o eliminados necesitan ser añadidos a staging primero.

```
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > echo "NEW LINE" >> texto.txt
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > git status -s
M texto.txt
?? index.html
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > git commit -am 'Commit sin ad
d'
[master 17f1696] Commit sin add
1 file changed, 2 insertions(+)
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > 
```

# HEAD

---

Cada `commit` se graba con un hash único que puede ser complicado de utilizar como referencia rápida. Para esto existe `HEAD`, que normalmente apunta al último `commit` de la rama activa.

```
# Mostrar la rama a la que apunta HEAD
git symbolic-ref HEAD

# Mostrar el hash del commit al que apunta HEAD
git rev-parse HEAD
```

```
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > git log
commit 17f16962f9c14c619c4da3eeefc1df35b4a884a8 (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date:   Mon Sep 22 00:20:30 2025 +0200

    Commit sin add

commit 64614f7a855aec9571d03259eea4daf6e38e8940
Author: silvelo <arturo.silvelo@gmail.com>
Date:   Mon Sep 22 00:10:43 2025 +0200

    Nuevo fichero
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > 
```

## Deshacer Cambios (Commits)

---

Si necesitamos deshacer el último `commit` porque nos hemos equivocado o faltan archivos, podemos hacerlo de dos maneras:

```
# Mantener los cambios
git reset --soft HEAD~1

# No mantener los cambios
git reset --hard HEAD~1
```

El `HEAD~1` indica que queremos movernos a la versión inmediatamente anterior a la actual.

**Nota:** Esto solo funcionará si los cambios no se han subido al repositorio remoto.



```
silvelo@silvelo-Inspiron-7559 ~/.course-example master > git status -s
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/.course-example master > git log
commit b50d1f1425fa4ffd135c65a7c32ff1ad9ac7dd26 (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:37:49 2025 +0200

    Commit sin add

commit 64614f7a855aec9571d03259eea4daf6e38e8940
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:10:43 2025 +0200

    Nuevo fichero
silvelo@silvelo-Inspiron-7559 ~/.course-example master > git reset --soft HEAD~1
silvelo@silvelo-Inspiron-7559 ~/.course-example master > git log
commit 64614f7a855aec9571d03259eea4daf6e38e8940 (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:10:43 2025 +0200

    Nuevo fichero
silvelo@silvelo-Inspiron-7559 ~/.course-example master > git status -s
M texto.txt
?? texto2.txt
```

```
silvelo@silvelo-Inspiron-7559 ~/.course-example & master > git status -s
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/.course-example & master > git log
commit 86c24f569c8da7c78fb5ad4b660ef17e8063fb0f (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:38:45 2025 +0200

    Commit sin add

commit 64614f7a855aec9571d03259eea4daf6e38e8940
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:10:43 2025 +0200

    Nuevo fichero
silvelo@silvelo-Inspiron-7559 ~/.course-example & master > git reset --hard HEAD~1
HEAD is now at 64614f7 Nuevo fichero
silvelo@silvelo-Inspiron-7559 ~/.course-example & master > git log
commit 64614f7a855aec9571d03259eea4daf6e38e8940 (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:10:43 2025 +0200

    Nuevo fichero
silvelo@silvelo-Inspiron-7559 ~/.course-example & master > git status -s
?? texto2.txt
```

## Arreglar Commit

---

Si lo único que necesitamos es corregir el último `commit`:

```
# Editar el mensaje
git commit --amend -m 'Nuevo mensaje'

# Añadir archivos y modificar el commit
git add archivo3.js
git commit --amend -m 'Nuevo mensaje'
```

El comando `amend` no crea un nuevo `commit`, sino que actualiza el anterior.

```
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git status -s
M texto.txt
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git commit -am "Mensaje erroneo"
[master c590050] Mensaje erroneo
1 file changed, 1 insertion(+)
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git log
commit c590050ed67fb654eaa97c683cc9a1c046be385a (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:48:03 2025 +0200

    Mensaje erroneo

commit 64614f7a855aec9571d03259eea4daf6e38e8940
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:10:43 2025 +0200

    Nuevo fichero

silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git commit --amend -m "Mensaje corregido"
[master 490bfb7] Mensaje corregido
Date: Mon Sep 22 00:48:03 2025 +0200
1 file changed, 1 insertion(+)
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git log
commit 490bfb797ee78925c70ae27ffa34ab3f623bfbb0 (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:48:03 2025 +0200

    Mensaje corregido

commit 64614f7a855aec9571d03259eea4daf6e38e8940
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 00:10:43 2025 +0200

    Nuevo fichero
```

# Deshacer cambios

---

## Deshacer un archivo modificado (Usando `git restore`)

---

Si modificamos un archivo y queremos volver al estado inicial, podemos usar el comando:

```
git restore index.html  
git restore .  
git restore '*.js'
```

**Nota:** Este comando hará que los cambios se pierdan. Si el archivo no está guardado en un commit previo, Git nos dará un error.

```
silvelo@silvelo-Inspiron-7559 ~/..../course-example % git status -s
?? index.html
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/..../course-example % cat texto.txt
HOLA MUNDO.txt
NEW LINE
NEW LINE
silvelo@silvelo-Inspiron-7559 ~/..../course-example % echo "HOLA MUNDO" >> texto.txt
silvelo@silvelo-Inspiron-7559 ~/..../course-example % cat texto.txt
HOLA MUNDO.txt
NEW LINE
NEW LINE
HOLA MUNDO
silvelo@silvelo-Inspiron-7559 ~/..../course-example % git restore texto.txt
silvelo@silvelo-Inspiron-7559 ~/..../course-example % cat texto.txt
HOLA MUNDO.txt
NEW LINE
NEW LINE
silvelo@silvelo-Inspiron-7559 ~/..../course-example %
```

```
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > git status -s
?? index.html
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > git restore texto2.txt
error: pathspec 'texto2.txt' did not match any file(s) known to git
silvelo@silvelo-Inspiron-7559 ~/../course-example & master > 
```



## Deshacer un archivo modificado (Usando `git checkout`)

---

El comando `git restore` es relativamente nuevo, y puede que no esté disponible en versiones antiguas de Git. En ese caso, podemos usar como alternativa:

```
git checkout -- index.html  
git checkout -- '*.md'  
git checkout .
```

Estos comandos tienen la misma función que `git restore`, restaurando el archivo o conjunto de archivos al estado anterior.

## Eliminar archivos no rastreados (Usando `git clean`)

---

Si queremos eliminar archivos no rastreados del directorio de trabajo, podemos usar el comando `git clean`.

```
touch index2.html  
git clean
```

## Opciones principales de `git clean`:

- `-n`: Muestra qué se eliminaría sin ejecutar la acción.
- `-f`: Fuerza la eliminación de los archivos.
- `-d`: Permite eliminar directorios no rastreados.
- `-i`: Activa el modo interactivo para confirmar cada acción.

```
git clean -n    # Muestra qué se eliminaría  
git clean -f    # Fuerza la limpieza
```

```
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > git clean -n index.html
Would remove index.html
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > git clean -f index.html
Removing index.html
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > git status -s
?? texto2.txt
silvelo@silvelo-Inspiron-7559 ~/../course-example % master > 
```