

COMANDOS GIT

Redacción paso a paso de como usar git/github.

Configuramos la “sesión” con:

'git config --global user.name "nombre"'

'git config --global user.email "mail"'

```
[alumno@vm1:~]$ git --version
git version 2.30.2
[alumno@vm1:~]$ git config --global user.name "PedroDavid"
git: 'config' no es un comando de git. Mira 'git --help'.

El comando más similar es
    config
[alumno@vm1:~]$ git config --global user.name "PedroDavid"
[alumno@vm1:~]$ git config --global user.email "4103404@alu.murciaeduca.es"
[alumno@vm1:~]$
```

Check de que esta bien **'git config --list'**.

```
[alumno@vm1:~]$ git config --global user.name "PedroDavid"
[alumno@vm1:~]$ git config --list
user.name=PedroDavid
user.email=4103404@alu.murciaeduca.es
```

Creamos repositorio y entramos en él.

```
[alumno@vm1:~]$ mkdir hello_world
[alumno@vm1:~]$ cd hello_world/
[alumno@vm1:~/hello_world]$
```

Iniciamos el repositorio con **'git init'**.

```
[alumno@vm1:~/hello_world]$ git init
ayuda: Using 'master' as the name for the initial branch. This default branch name
ayuda: is subject to change. To configure the initial branch name to use in all
ayuda: of your new repositories, which will suppress this warning, call:
ayuda:
ayuda:   git config --global init.defaultBranch <name>
ayuda:
ayuda: Names commonly chosen instead of 'master' are 'main', 'trunk' and
ayuda: 'development'. The just-created branch can be renamed via this command:
ayuda:
ayuda:   git branch -m <name>
Iniciado repositorio Git vacío en /home/alumno/hello_world/.git/
```

Comprobamos el estado del repositorio con '**git status**'.

```
[alumno@vm1:~/hello_world]$ git status
En la rama master

No hay commits todavía

no hay nada para confirmar (crea/copia archivos y usa "git add" para hacerles seguimiento)
```

Creamos un archivo.

```
[alumno@vm1:~/hello_world]$ nano readme.txt

[alumno@vm1:~/hello_world]$ git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    readme.txt

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
```

Añadimos el archivo con '**git add nombreArchivo**' para que así lo tengamos en cuenta a la hora de hacer el commit.

```
[alumno@vm1:~/hello_world]$ git add readme.txt
[alumno@vm1:~/hello_world]$ git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
  (usa "git rm --cached <archivo>..." para sacar del área de stage)
    nuevo archivo:  readme.txt
```

Hacemos un registro del cambio con '**git commit -m "mensaje explicando cambios"**'.

```
[alumno@vm1:~/hello_world]$ git commit -m "Este es mi segundo commit"
[master (commit-raíz) 7feee77] Este es mi segundo commit
1 file changed, 2 insertions(+)
create mode 100644 readme.txt
```

Si añadimos una nueva línea al archivo readme podemos ver las diferencias en la versión con '**git diff**'. Vemos que git a tenido en cuenta que hemos insertado esa línea

```
create mode 100644 README.txt
[alumno@vm1:~/hello-world]$ nano README.txt
Tiene correo nuevo en /var/mail/alumno
[alumno@vm1:~/hello-world]$ git diff
diff --git a/README.txt b/README.txt
index c7726b4..4b6711c 100644
--- a/README.txt
+++ b/README.txt
@@ -1,2 +1,2 @@
 Este es el archivo README.txt del proyecto.
-
+Eso es todo.
[alumno@vm1:~/hello-world]$
```

De nuevo con **status** vemos que tenemos que guardar.

```
[alumno@vm1:~/hello-world]$ git status
En la rama master
Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
        modificado:    README.txt

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
[alumno@vm1:~/hello-world]$
```

Viendo la imagen anterior vemos que tenemos que volver a hacer un **add** para meter el readme en el siguiente commit.

```
[alumno@vm1:~/hello-world]$ git add README.txt
[alumno@vm1:~/hello-world]$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
        modificado:    README.txt

[alumno@vm1:~/hello-world]$
```

Realizamos el segundo **commit** para guardar los cambios.

```
[alumno@vm1:~/hello-world]$ git commit -m "Una nueva línea"
[master 0113f76] Una nueva línea
 1 file changed, 1 insertion(+), 1 deletion(-)
[alumno@vm1:~/hello-world]$
```

Con **git log** podemos ver el historial de versiones y con **git checkout** y el **identificador del commit** podemos volver a esa versión, pero ojo, es peligroso y para nada recomendable cuando se trabaja en grupo, ya que otra persona podría estar trabajando con la nueva versión que tu quieres “desechar”, digamos que lo que he compartido no debo deshacerlo.

```
[alumno@vm1:~/hello-world]$ git log
commit 0113f76ab20aa2fa9b574108e7f9e388e6012ad6 (HEAD -> master)
Author: Jesus <4186790@alu.murciaeduca.es>
Date: Mon Oct 23 08:18:44 2023 +0200

    Una nueva línea

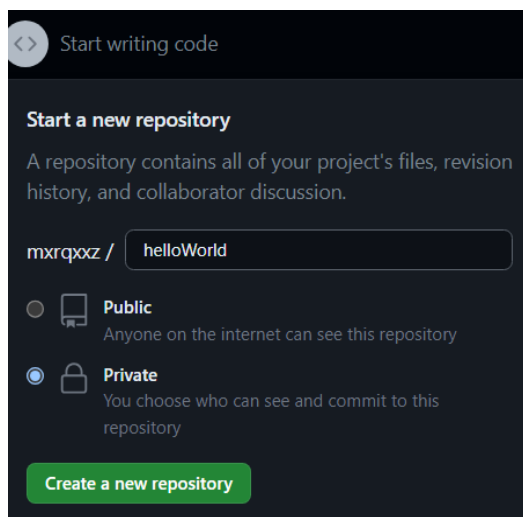
commit a366c0625049fab5bf9285a0cd2c0548d24f4f41
Author: Jesus <4186790@alu.murciaeduca.es>
Date: Thu Oct 19 09:51:57 2023 +0200

    primer commit
[alumno@vm1:~/hello-world]$
```

Configuramos nuestro nombre de GitHub con user.username

```
[alumno@vm1:~/hello-world]$ git config --global user.username mxrqxxz
[alumno@vm1:~/hello-world]$
```

Creamos en github un repositorio, se recomienda que tenga el mismo nombre que la carpeta en la que hemos hecho el git init.



The screenshot shows the GitHub 'Start a new repository' interface. At the top, there's a 'Start writing code' button. Below it, the title 'Start a new repository' is followed by a description: 'A repository contains all of your project's files, revision history, and collaborator discussion.' The repository name is set to 'mxrqxxz / helloWorld'. There are two radio button options: 'Public' (with a globe icon) and 'Private' (with a lock icon). The 'Private' option is selected. At the bottom, there is a green button labeled 'Create a new repository'.

Quick setup — if you've done this kind of thing before

 Set up in Desktop or  HTTPS  SSH <https://github.com/mxrqxxz/helloWorld.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository in

Añadimos el repositorio de GitHub a nuestro equipo y le ponemos el nombre origin (por convenio). **git remote add**
Para borrar un repositorio de GitHub en nuestro equipo lo podemos hacer con **git remote remove**.

```
[alumno@vm1:~/hello-world]$ git remote add origin https://github.com/mxrqxxz/helloWorld.git
[alumno@vm1:~/hello-world]$ git remote -v
origin  https://github.com/mxrqxxz/helloWorld.git (fetch)
origin  https://github.com/mxrqxxz/helloWorld.git (push)
[alumno@vm1:~/hello-world]$
```

Tenemos que añadir una clave ssh para poder hacer push, para ello seguimos este enlace:

<https://docs.github.com/es/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent?platform=linux>

```
[alumno@vm1:~/hello-world]$ ssh-keygen -t ed25519 -C "4186790@alu.murciaeduca.es"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/alumno/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alumno/.ssh/id_ed25519
Your public key has been saved in /home/alumno/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:W/KyPB+Pryf+0a5kg3PbsxXcQvXLxtcc+iPjNIuUe3I 4186790@alu.murciaeduca.es
The key's randomart image is:
+--[ED25519 256]--+
|          .|
|         ..|
|        ...|
|       .|=|
|      S .  .B=|
|     =  ...0.0|
|    o =0*=.0.|
|   ..ooX*E*..|
|  o+=OX++o |
+-----[SHA256]-----+
[alumno@vm1:~/hello-world]$ eval "$(ssh-agent -s)"
Agent pid 13050
[alumno@vm1:~/hello-world]$ ssh-add ~/.ssh/id_ed25519
Enter passphrase for /home/alumno/.ssh/id_ed25519:
Identity added: /home/alumno/.ssh/id_ed25519 (4186790@alu.murciaeduca.es)
[alumno@vm1:~/hello-world]$
```

Copiamos la clave, la necesitamos para añadirla a la cuenta de GitHub.

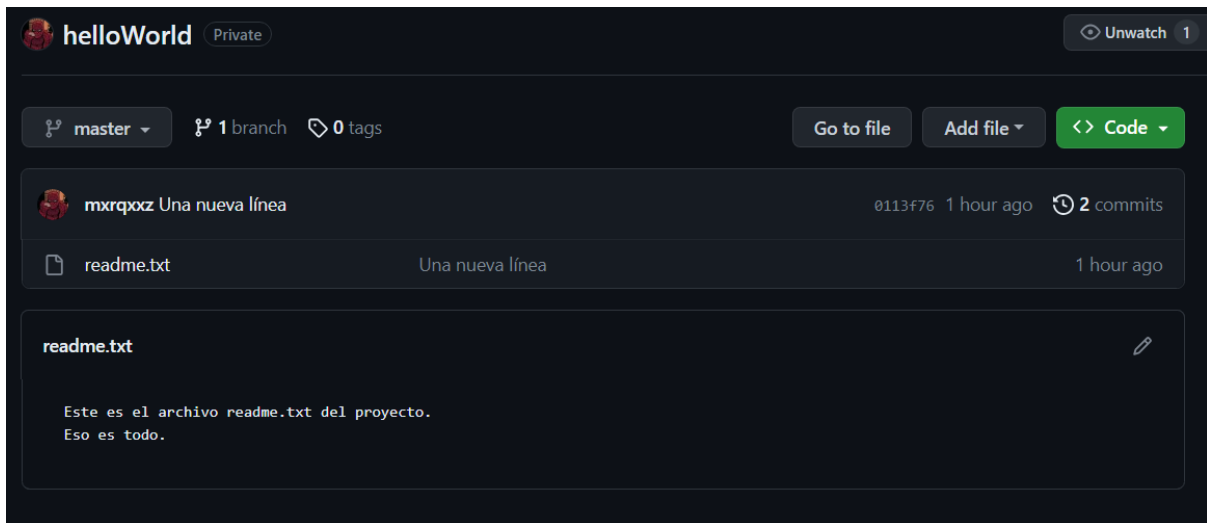
```
[alumno@vm1:~/hello-world]$ cat ~/.ssh/id_ed25519.pub  
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAgtZz9XcXlr6Ibsp7rXk3+vi5CP/Lnqpz1mOVwccAub 4186790@alu.murciaeduca.es
```

The screenshot shows the GitHub 'Add new SSH Key' interface. On the left is a sidebar with navigation links: Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys (highlighted), Organizations, Enterprises, and Moderation. The main content area is titled 'Add new SSH Key' and contains a form with the following fields: 'Title' (with the value 'clave'), 'Key type' (set to 'Authentication Key'), and 'Key' (containing the SSH public key from the terminal output above). A green 'Add SSH key' button is at the bottom right.

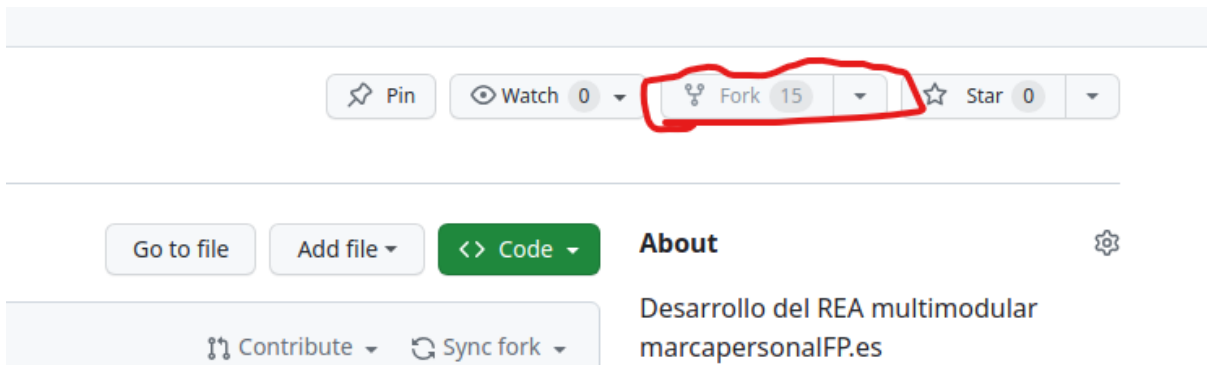
Añadimos el remote de ssh y hacemos **git push origin master**

```
[alumno@vm1:~/hello-world]$ git remote set-url origin git@github.com:mxrquxz/helloWorld.git  
[alumno@vm1:~/hello-world]$ git push origin master  
Enumerando objetos: 6, listo.  
Contando objetos: 100% (6/6), listo.  
Compresión delta usando hasta 2 hilos  
Comprimiendo objetos: 100% (3/3), listo.  
Escribiendo objetos: 100% (6/6), 530 bytes | 530.00 KiB/s, listo.  
Total 6 (delta 0), reusado 0 (delta 0), pack-reusado 0  
To github.com:mxrquxz/helloWorld.git  
* [new branch]      master -> master  
[alumno@vm1:~/hello-world]$
```

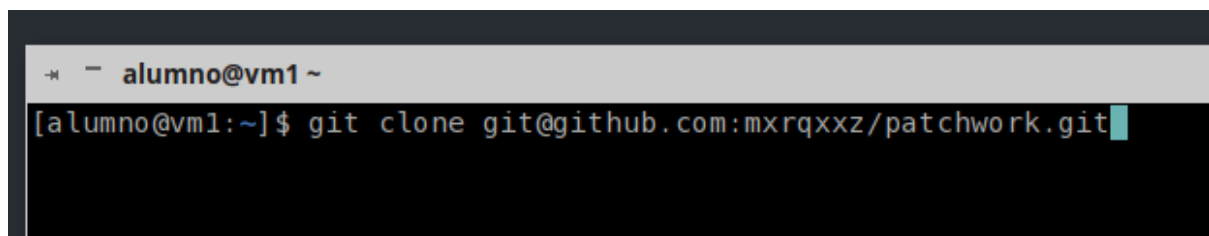
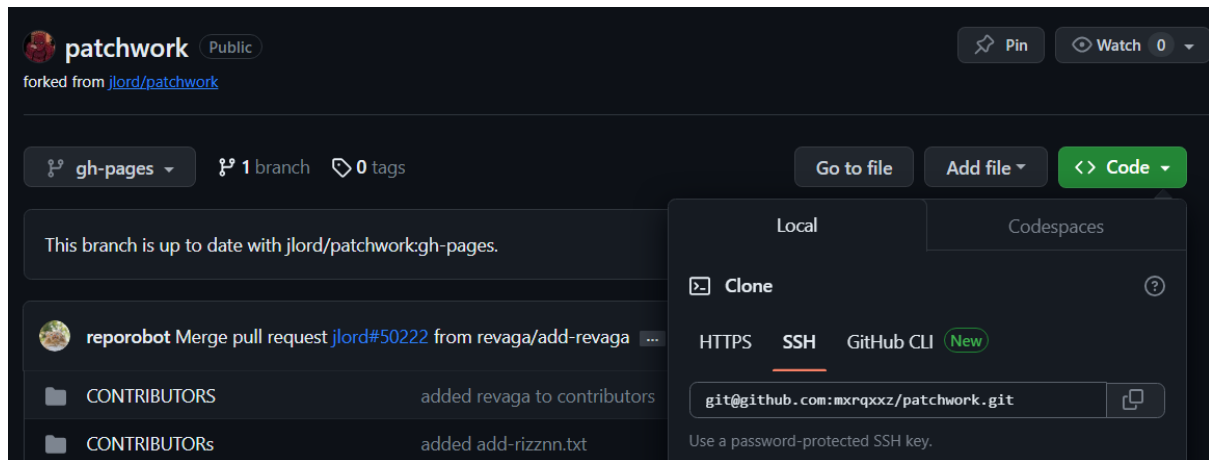
Ya tenemos los cambios listos.



Fork, es hacer una copia de un proyecto de GitHub en tu perfil, se hace desde el otro proyecto con botón **FORK**.

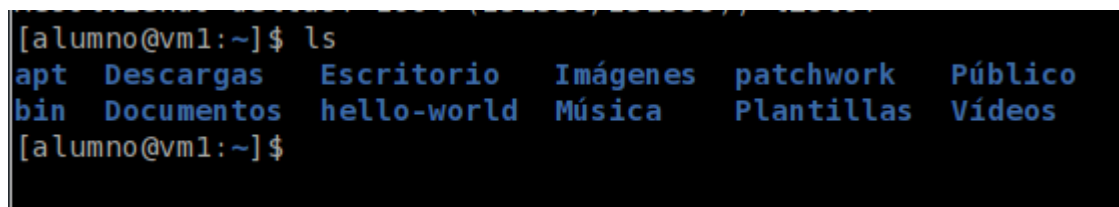


Hacemos **git clone** con la clave ssh para copiar el repositorio al equipo

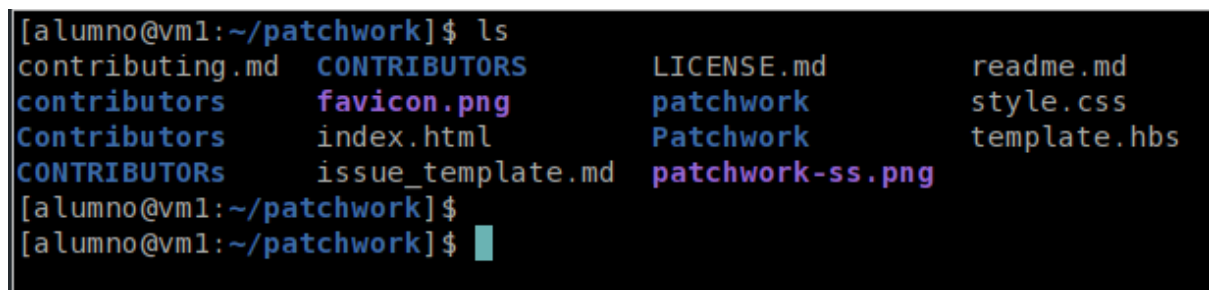


AHORA HACEMOS UN EJEMPLO

Ya tenemos la carpeta del proyecto en el equipo (patchwork)



Si nos metemos ya tenemos todos los archivos del repositorio.



Nos interesa estar conectados también al repositorio original, ya que también habrá otra gente que lo actualice, para ello hacemos otro remote add pero tendrá el nombre de **upstream**


```
[alumno@vm1:~/patchwork]$ git remote add upstream https://github.com/jlord/patchwork.git
[alumno@vm1:~/patchwork]$
```

Vamos a crear una rama del repositorio con **git branch**, con **-a** listamos todas las ramas, nos cambiamos a nuestra rama con checkout.

```
[alumno@vm1:~/patchwork]$ git branch add-mxrqxxz
[alumno@vm1:~/patchwork]$ git branch -a
  add-mxrqxxz
* gh-pages
  remotes/origin/HEAD -> origin/gh-pages
  remotes/origin/gh-pages
[alumno@vm1:~/patchwork]$ git checkout add-mxrqxxz
Cambiado a rama 'add-mxrqxxz'
[alumno@vm1:~/patchwork]$
```

Nos metemos a la carpeta contributors y creamos un nuevo archivo. nano add-xxxx.txt

```
[alumno@vm1:~/patchwork]$ cd contributors/
[alumno@vm1:~/patchwork/contributors]$ nano add-mxrqxxz
[alumno@vm1:~/patchwork/contributors]$ git status
En la rama add-mxrqxxz
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
  add-mxrqxxz

no hay nada agregado al commit pero hay archivos sin seguimiento presentes
(usa "git add" para hacerles seguimiento)
[alumno@vm1:~/patchwork/contributors]$
```

Hacemos un git add <nombreArchivo>

```
GNU nano 5.4 add-mxrqxxz
Hola, prueba de branch.
```

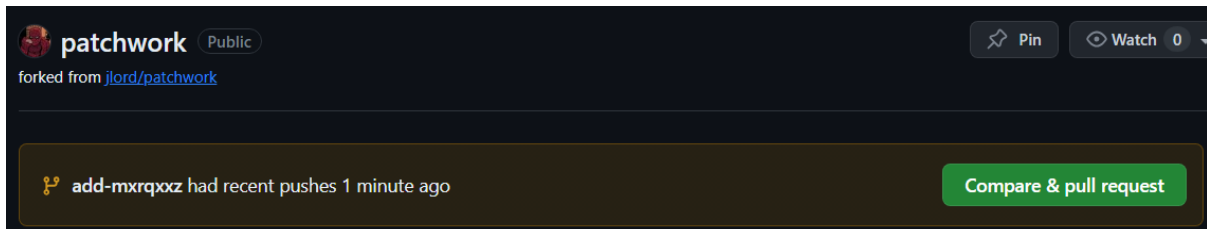
Hacemos el commit

```
[alumno@vm1:~/patchwork/contributors]$ git commit -m "Soy un nuevo colaborador"
[add-mxrqxxz f52bbdfale] Soy un nuevo colaborador
1 file changed, 1 insertion(+)
create mode 100644 contributors/add-mxrqxxz
```

Hacemos un push de nuestra rama

```
[alumno@vm1:~/patchwork/contributors]$ git push origin add-mxrqxxz
Enumerando objetos: 6, listo.
Contando objetos: 100% (6/6), listo.
Compresión delta usando hasta 2 hilos
Comprimiendo objetos: 100% (3/3), listo.
Escribiendo objetos: 100% (4/4), 383 bytes | 383.00 KiB/s, listo.
Total 4 (delta 1), reusado 0 (delta 0), pack-reusado 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'add-mxrqxxz' on GitHub by visiting:
remote:   https://github.com/mxrqxxz/patchwork/pull/new/add-mxrqxxz
remote:
To github.com:mxrqxxz/patchwork.git
 * [new branch]      add-mxrqxxz -> add-mxrqxxz
[alumno@vm1:~/patchwork/contributors]$
```

Ya sale en GitHub.



Para mantener actualizado en local nuestro repositorio de alguna actualización que haya podido hacer un colaborador necesitamos usar el comando `git pull origin(repositorio) add-mxrqxxz(rama)`

```
[alumno@vm1:~/patchwork]$ git pull origin add-mxrqxxz
ayuda: Hacer un pull sin especificar cómo reconciliar las ramas es poco
ayuda: recomendable. Puedes eliminar este mensaje usando uno de los
ayuda: siguientes comandos antes de tu siguiente pull:
ayuda:
ayuda:   git config pull.rebase false  # hacer merge (estrategia por defecto)
ayuda:   git config pull.rebase true   # aplicar rebase
ayuda:   git config pull.ff only        # aplicar solo fast-forward
ayuda:
ayuda: Puedes reemplazar "git config" con "git config --global" para aplicar
ayuda: la preferencia en todos los repositorios. Puedes también pasar --rebase,
ayuda: --no-rebase, o --ff-only en el comando para sobrescribir la configuraci
ayuda: ón
ayuda: por defecto en cada invocación.
ayuda:
Desde github.com:mxrqxxz/patchwork
* branch              add-mxrqxxz -> FETCH_HEAD
Ya está actualizado.
[alumno@vm1:~/patchwork]$
```

