



# Cómo sobrevivir a un proyecto en grupo usando Github



Markel Ferro Postigo



¿Objetivo de la charla?

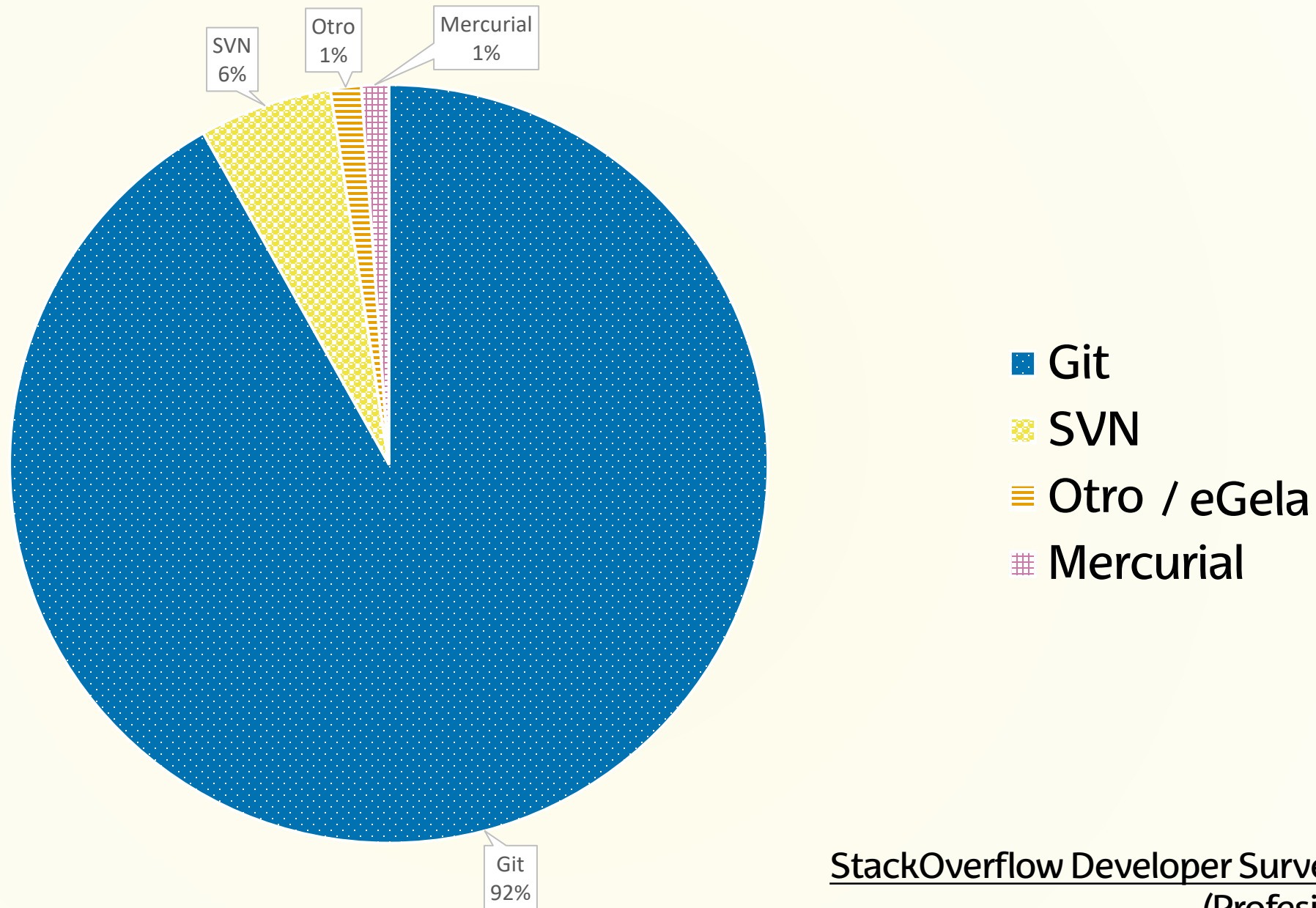
# Sobrevivir a un proyecto en grupo usando Github

(Es decir, explicar los aspectos necesarios y dar pequeños trucos)

¿Y por qué es necesario aprender Git?



# Uso de sistemas de control de versiones



StackOverflow Developer Survey 2022  
(Profesionales)



# Algunas aclaraciones

- No soy un experto en Git (in the slightest)
- La charla tiene pinta de que va a ser larga, esperemos que sea amena.
- Hay cosas que me voy a saltar
- Si no hacéis preguntas os las haré yo a vosotros/as
- Tenéis el PPT disponible en: <https://labur.eus/gitmk>



# Conceptos iniciales



1. Commits
2. Stages
3. Branches



# Conceptos iniciales



myfile.html

```
<html>
<body>
  <h1>Izenburua</h1>
</body>
</html>
```

Cuando creamos un archivo, se crea un nuevo estado en el historial del repositorio.

1. Commits
2. Stages
3. Branches

# Conceptos iniciales



myfile.html

```
<html>
<body>
  <h1>Izenburua</h1>
  <h2>Nire subtitulua</h2>
</body>
</html>
```

Lo mismo cuando modificamos un archivo, aunque sea añadir o borrar una línea. Y siempre podemos volver a un estado anterior.

1. Commits
2. Stages
3. Branches

# Conceptos iniciales



myfile.html

```
<html>
<body>
  <h1>Izenburua</h1>
  <h2>Nire subtitulua</h2>
  <p>Eta textu pixka bat.</p>
</body>
</html>
```

Lo mismo cuando modificamos un archivo, aunque sea añadir o borrar una línea. Y siempre podemos volver a un estado anterior.

1. Commits
2. Stages
3. Branches



# “Pero yo tengo un buen sistema”

El sistema:



Trabajo  
final.docx



Trabajo  
FINALFINAL.docx



Trabajo  
FINALREVISADO  
PARA  
ENVIAR.docx



Trabajo  
FINALREVISADO.  
docx



Trabajo v2.docx











Trabajo.docx



# Puede que no sea un sistema perfecto

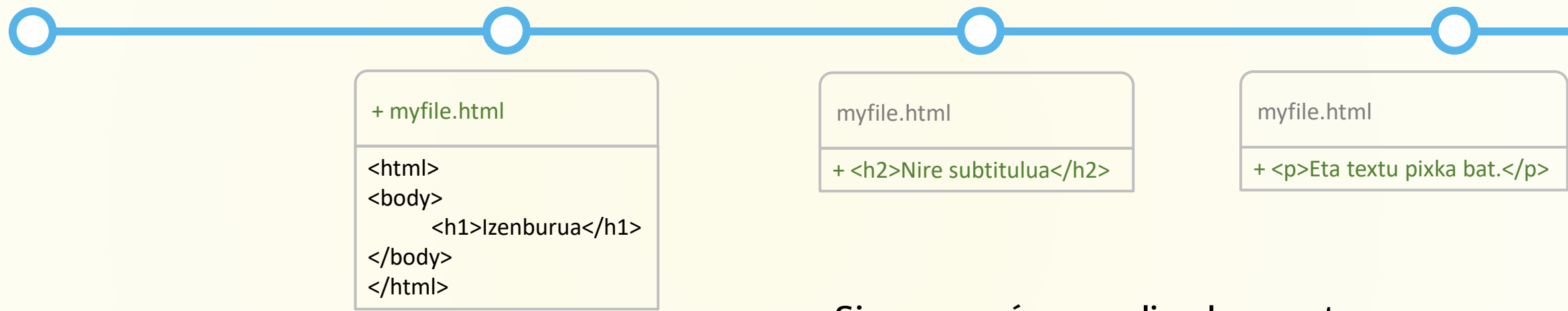
## Complejidad

 /db
 /doc
 /javadoc
 /resources
 /java
 /src
 pom.xml
 Readme.md

## Tamaño

Nombre	Tamaño (MB)
Trabajo final.docx	0,6
Trabajo FINALFINAL.docx	1
Trabajo FINALREVISADO PARA ENVIAR.docx	1,3
Trabajo FINALREVISADO.docx	1,1
Trabajo v2.docx	0,4
Trabajo.docx	0,2

Total: 4.6MB



¡Siempre más complicado y costoso  
que guardar solo la última versión!  
Pero...

# Solo se guardan las diferencias

1. Commits
2. Stages
3. Branches



# Permite trabajar en equipo

Lur



Trabaja en  
*granja.html*

Eder







Trabaja en  
*contacto.html*

¡Al mismo tiempo!\*

\*Sigue requiriendo coordinarse

# Solo subir algunos archivos. ¿Qué añadir en el commit?




 /src	
 pom.xml	C
 Readme.md	C

Hacemos  
stage a este  
archivo




1. Commits
2. Stages
3. Branches

# Solo subir algunos archivos. ¿Qué añadir en el commit?

Al hacer commit



A horizontal blue line with two white circles at each end, representing a timeline or process flow.

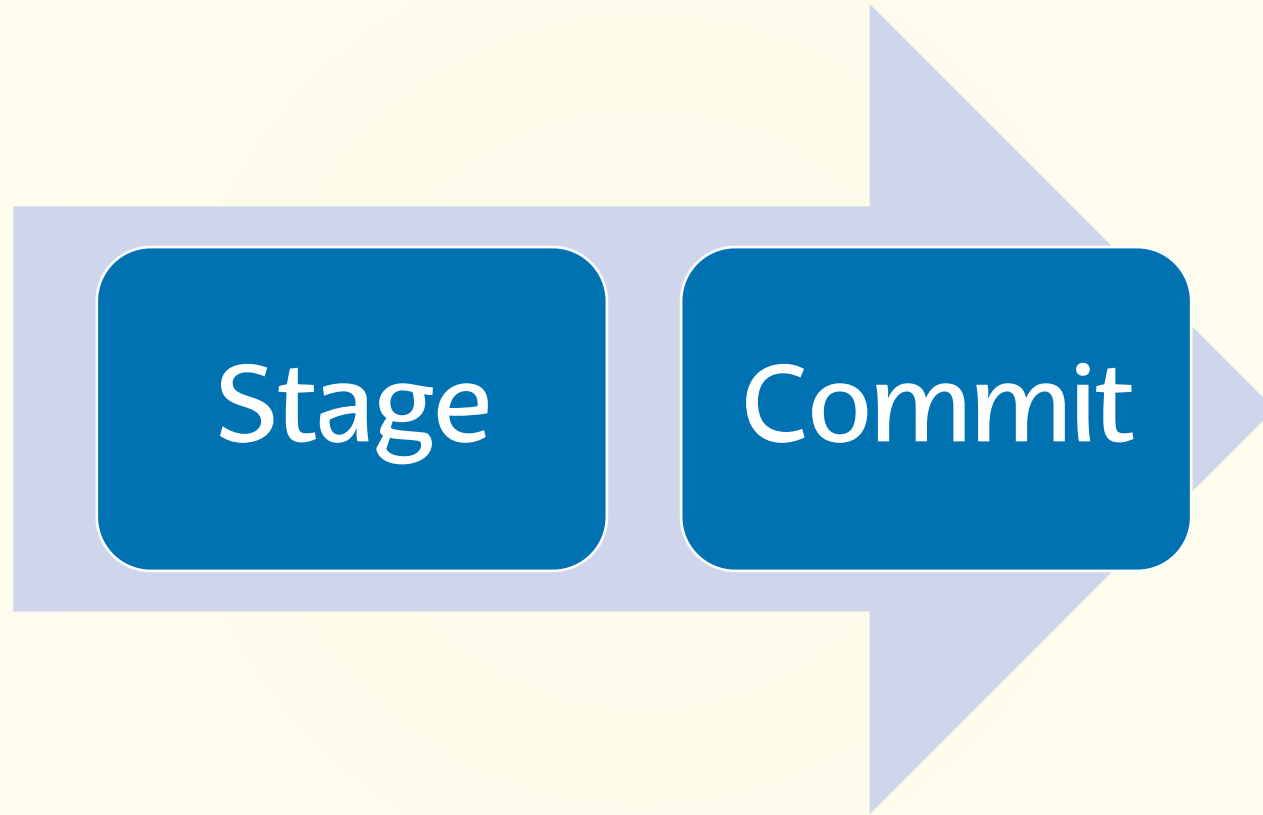
 /src	
 pom.xml	
 Readme.md	C

Actualizado

Pendiente de  
subir

1. Commits
2. Stages
3. Branches

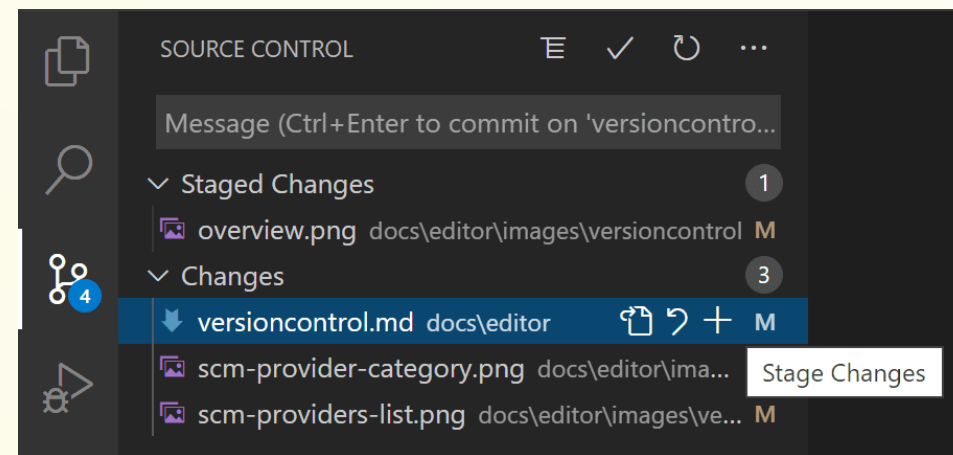
# El proceso



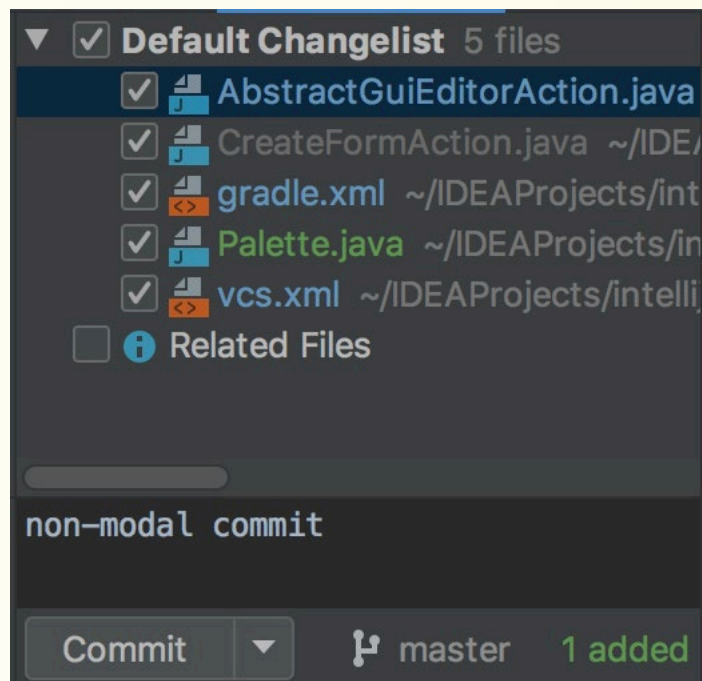
# Cómo hacer esto

```
git add . # Stage todos los ficheros
git commit # Hacer commit de los ficheros
```

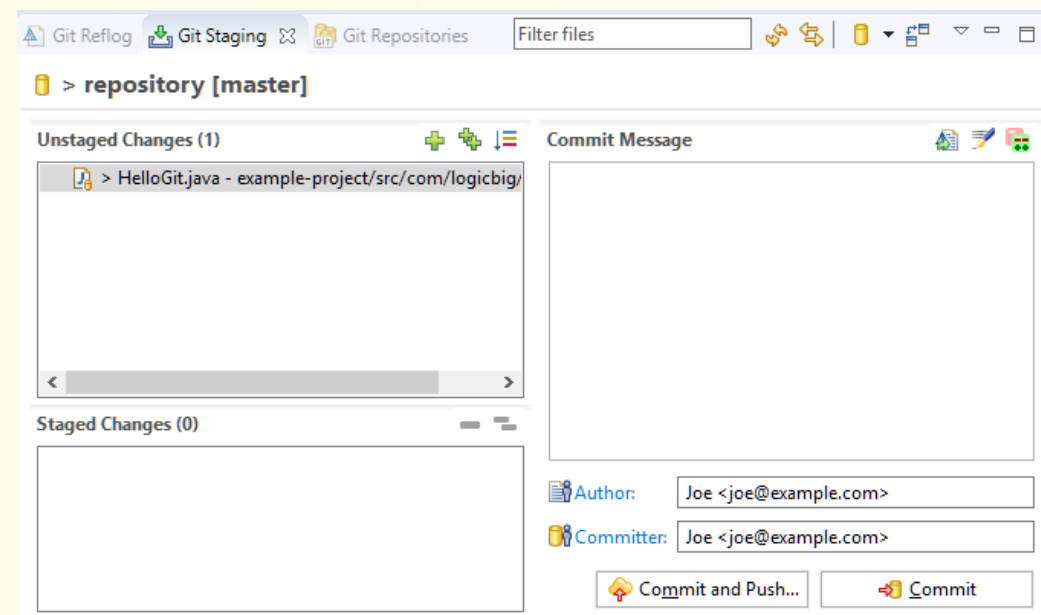
(a) Consola



(b) VSCode



(c) IntelliJ



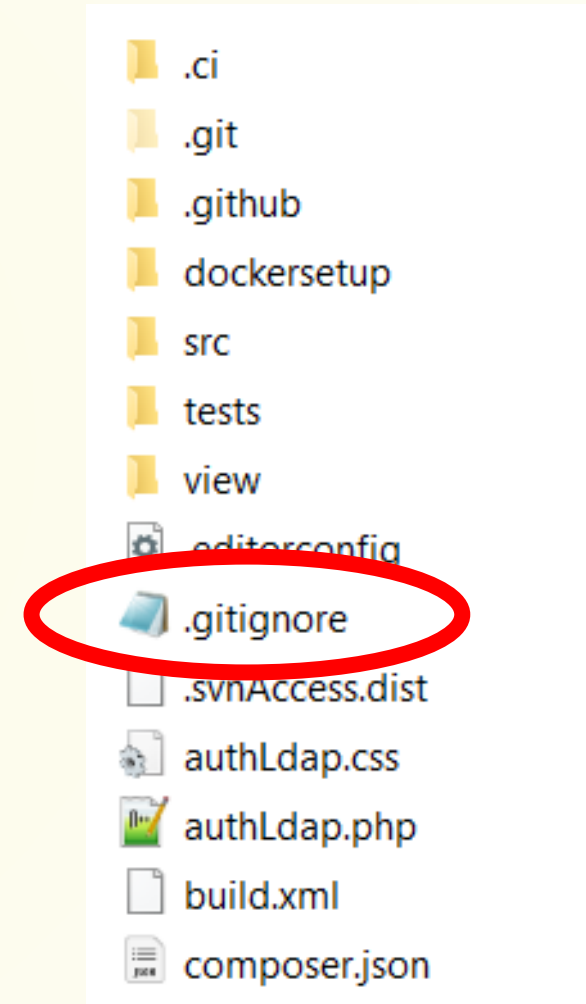
(d) Eclipse



# Gitignore

A veces necesitas no subir algunos archivos:

- Son de prueba
- Tienen contraseñas
- Simplemente es una copia que quieres guardarte



# Gitignore

Una excepción por  
línea.

El archivo debe  
llamarse .gitignore

```
1  /dist
2  /gh-pages
3  /node_modules
4  /test/coverage
5  /test/*.html
6  modernizr.js
7  metadata.json
8  .DS_Store
9  .idea
10 .project
11 *.log
12 tmp
13 /.nyc_output[EOF]
```

# Gitignore

Patrón	Ejemplos	Explicación
**/logs	logs/debug.log logs/monday/foo.bar build/logs/debug.log	Con dobles asteriscos puedes indicar cualquier directorio.
**/logs/debug.log	logs/debug.log build/logs/debug.log but not logs/build/debug.log	Después de ello puedes escribir más condiciones.
*.log	debug.log foo.log .log logs/debug.log	Con un asterisco incluyes cualquier tipo de caracteres (varios)
debug.log	debug.log logs/debug.log	El nombre de un archivo, sin más.



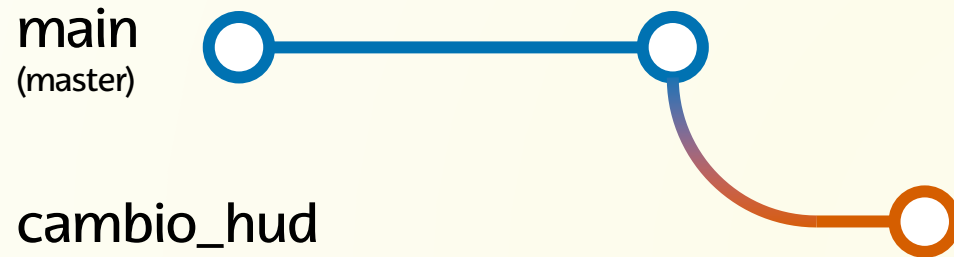
# Conceptos iniciales



main es la branch principal. Aquí hay que tener una versión funcional de tu proyecto en todo momento (la que entregas en eGela o a tu jefe).

1. Commits
2. Stages
3. Branches

# Conceptos iniciales



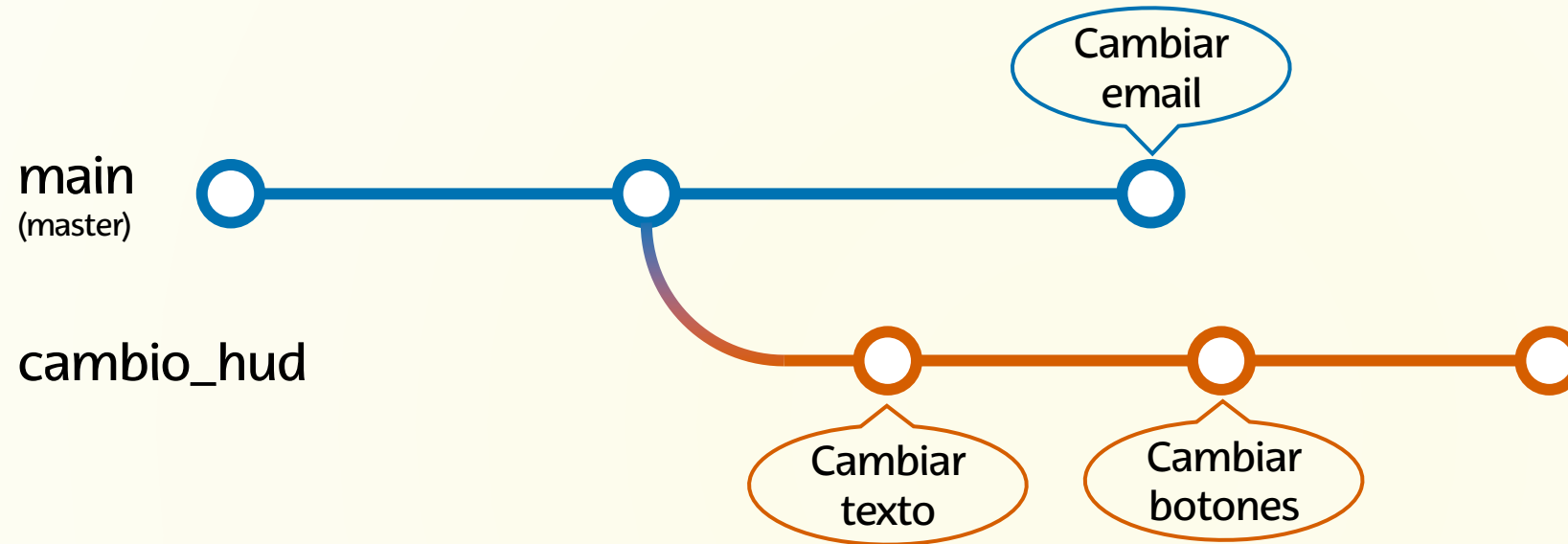
**¿Y si quieres hacer una modificación grande?**

Hacerlo sobre main sería contraproducente:

- Puede que no te de tiempo a terminarla
- Un(a) compañero/a trabaje sobre una versión no terminada.

1. Commits
2. Stages
3. Branches

# Conceptos iniciales

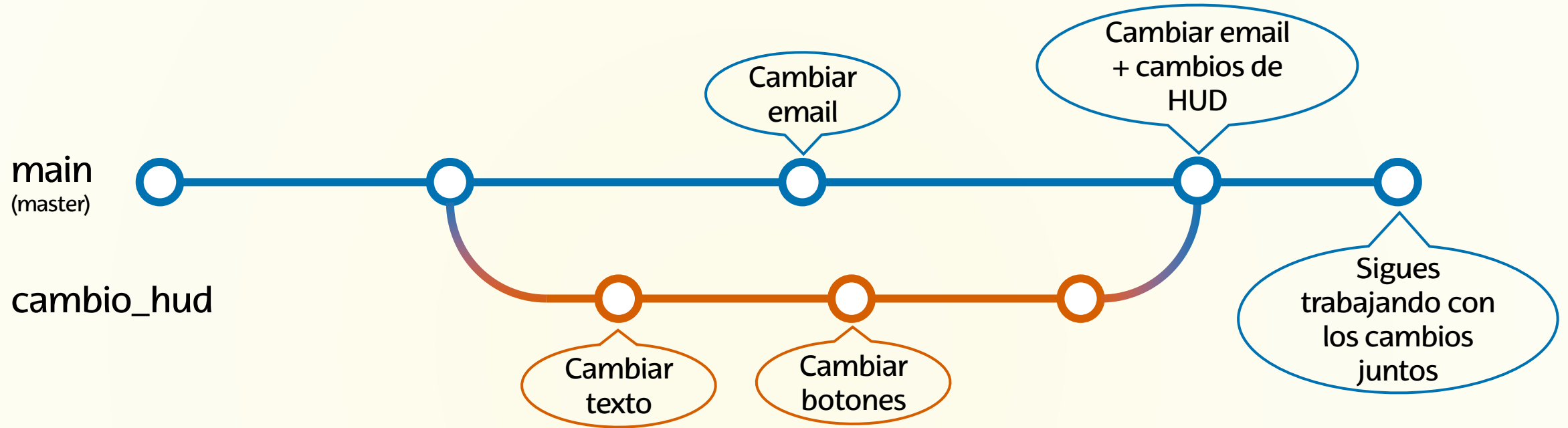


## Además:

- Puede que tu código no funcione (y main tiene que ser estable).
- ¿Cualquiera puede escribir en main? ¿Incluso la persona nueva de la empresa?

1. Commits
2. Stages
3. Branches

# Conceptos iniciales



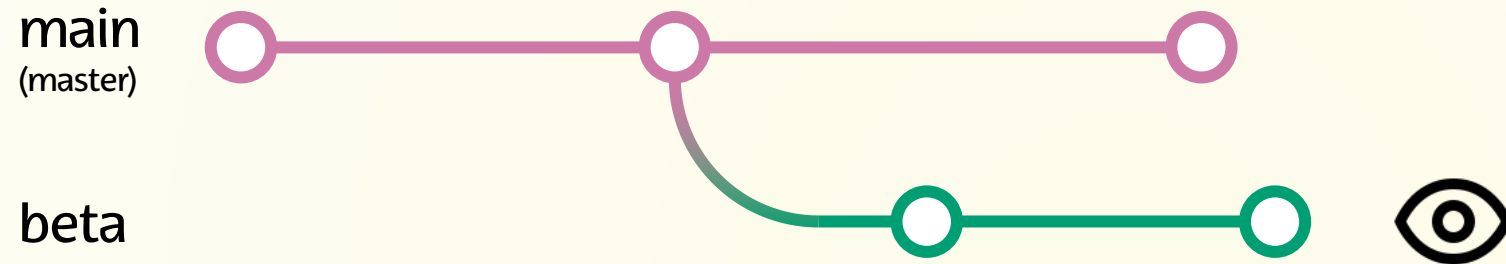
Una vez acabado el trabajo → Juntamos (merge) las ramas

El proceso de juntar las ramas es complicado,  
lo exploramos más adelante

1. Commits
2. Stages
3. Branches



# Cambiar de branch



Puedes ir cambiando de una rama a otra.  
Se le llama: **Checkout**

Explicación guay  
Explicación super deep

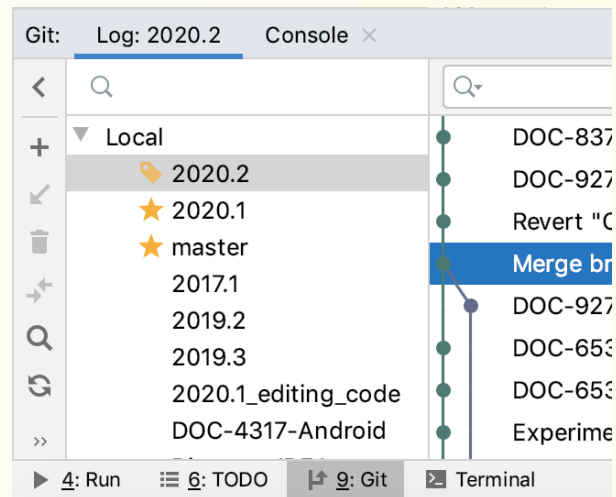
1. Commits
2. Stages
3. Branches



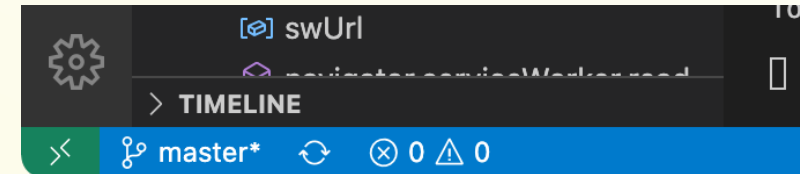
# Cómo hacer esto

```
git branch # Lista de ramas
git checkout <Nombre> # Cambiar
git checkout -b <Nombre> # Crear y cambiar
```

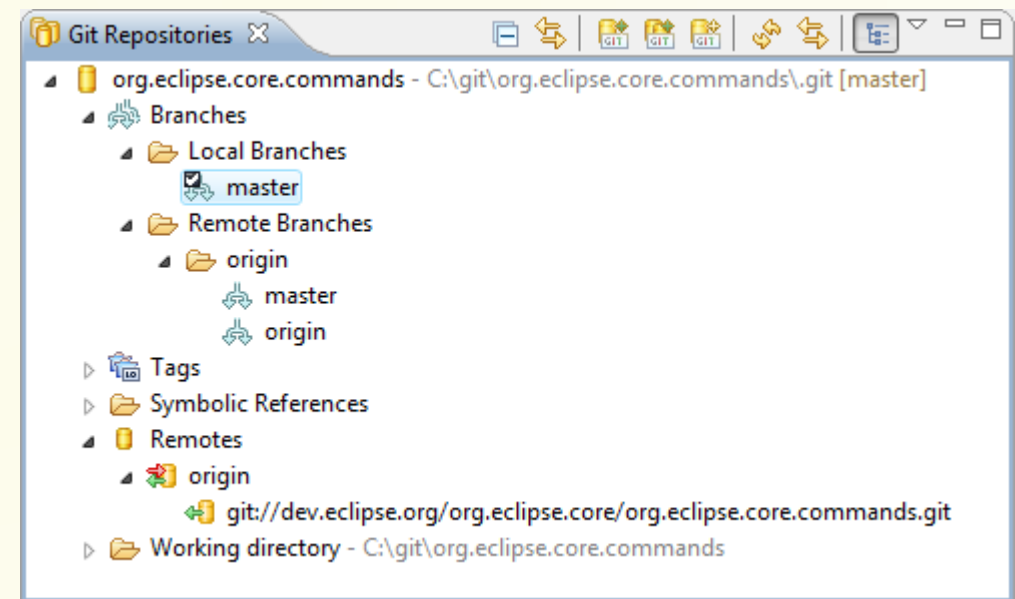
(a) Consola



(c) IntelliJ



(b) VSCode

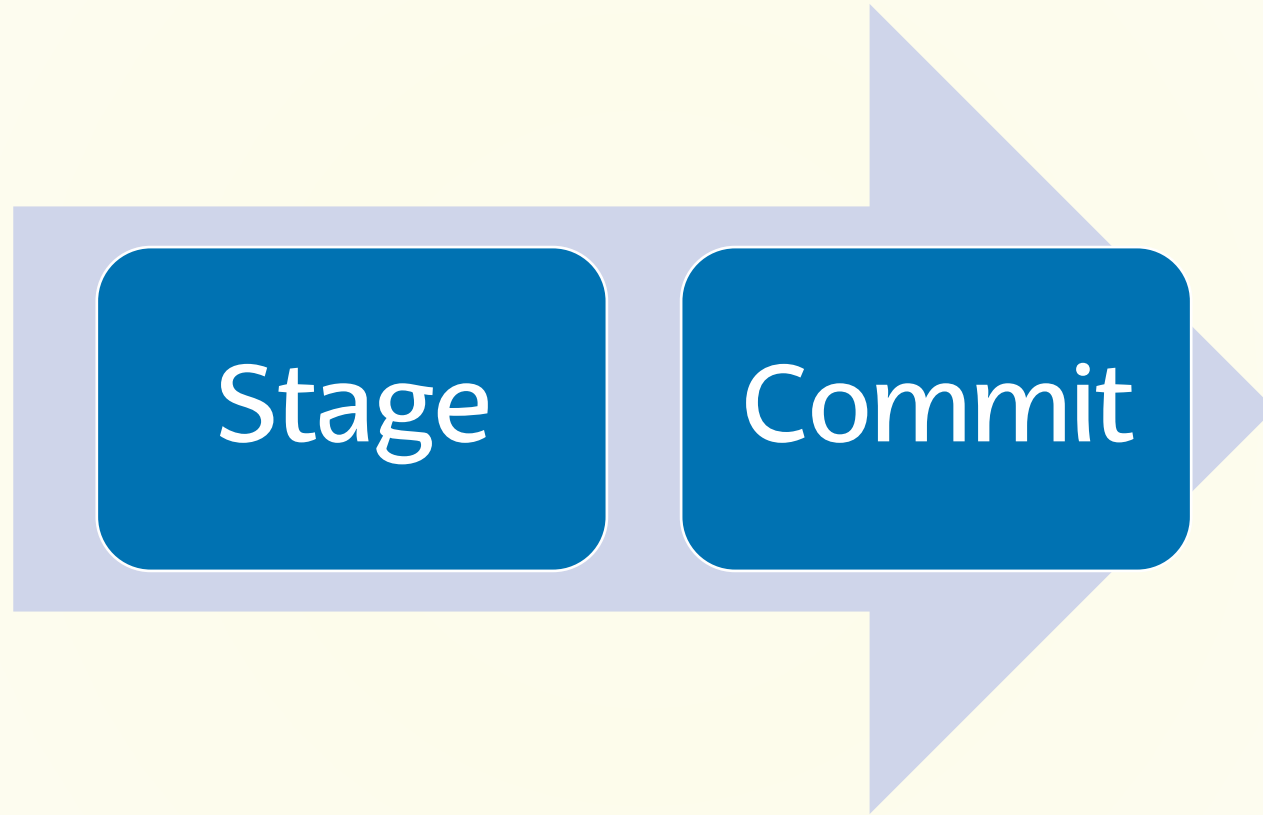


(d) Eclipse

¿Y qué debería ir en una rama?

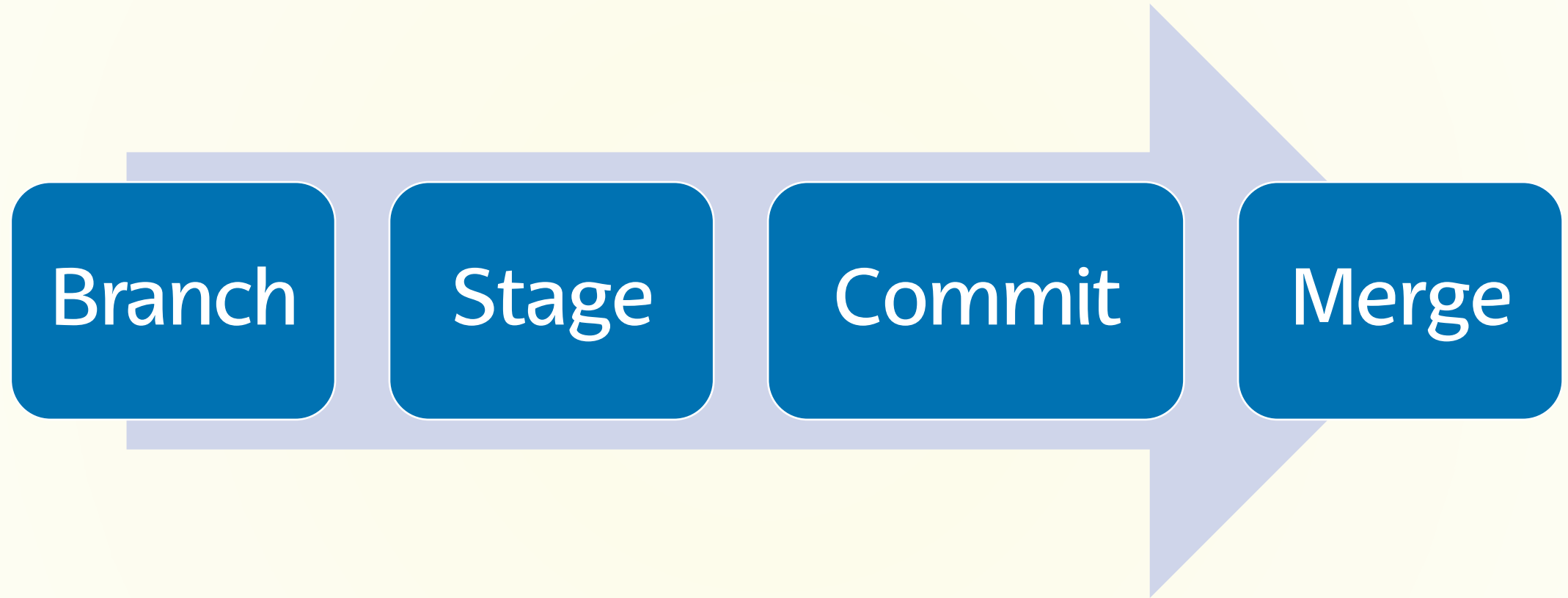


# El proceso






# El proceso




# Resumen

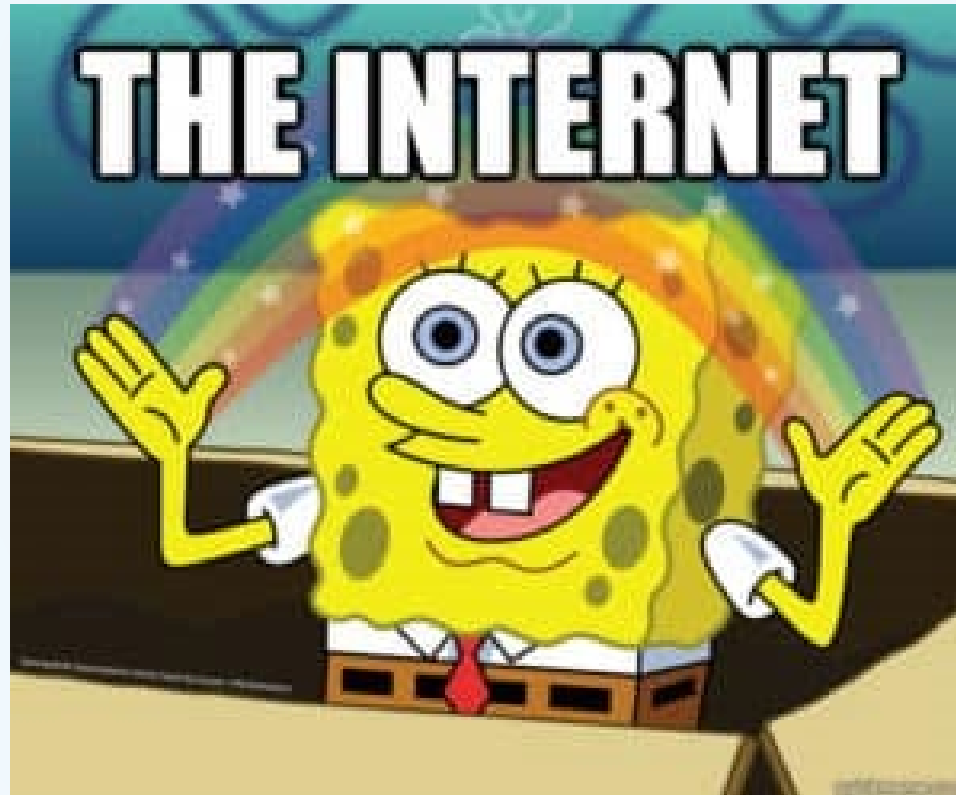
Repositorio: El proyecto (como una carpeta)

 (commit): Un punto específico en el repositorio (entre commit y commit hay cambios).

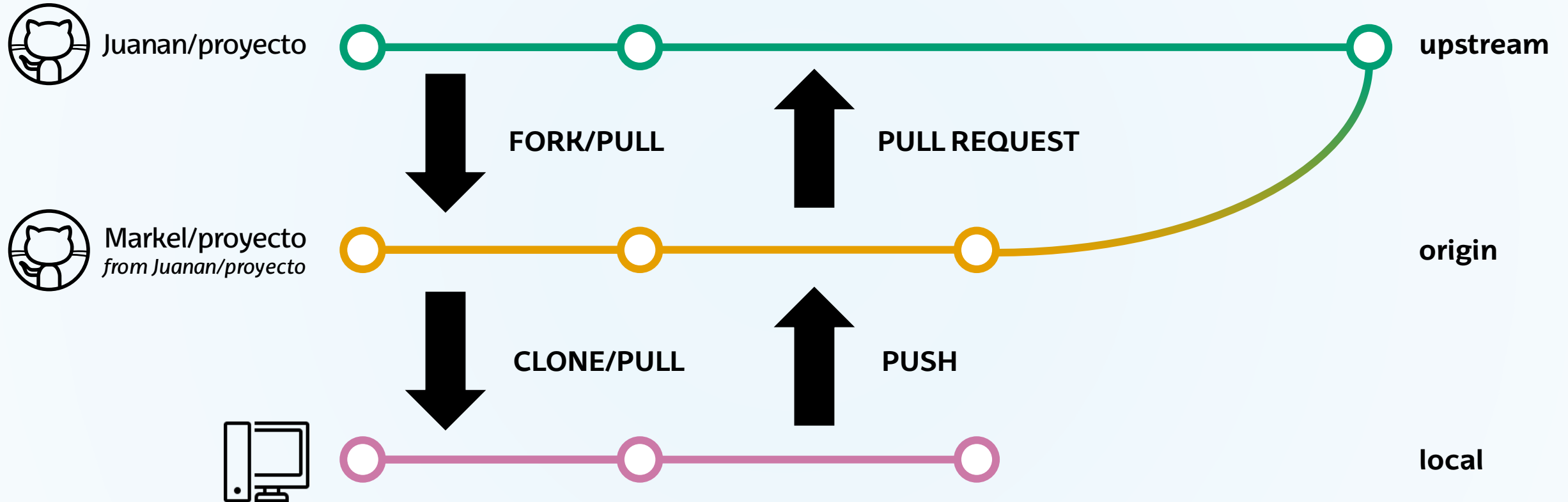
Stage: Elegir qué archivos van al commit

 (branch): Para trabajar en paralelo (un mundo paralelo).

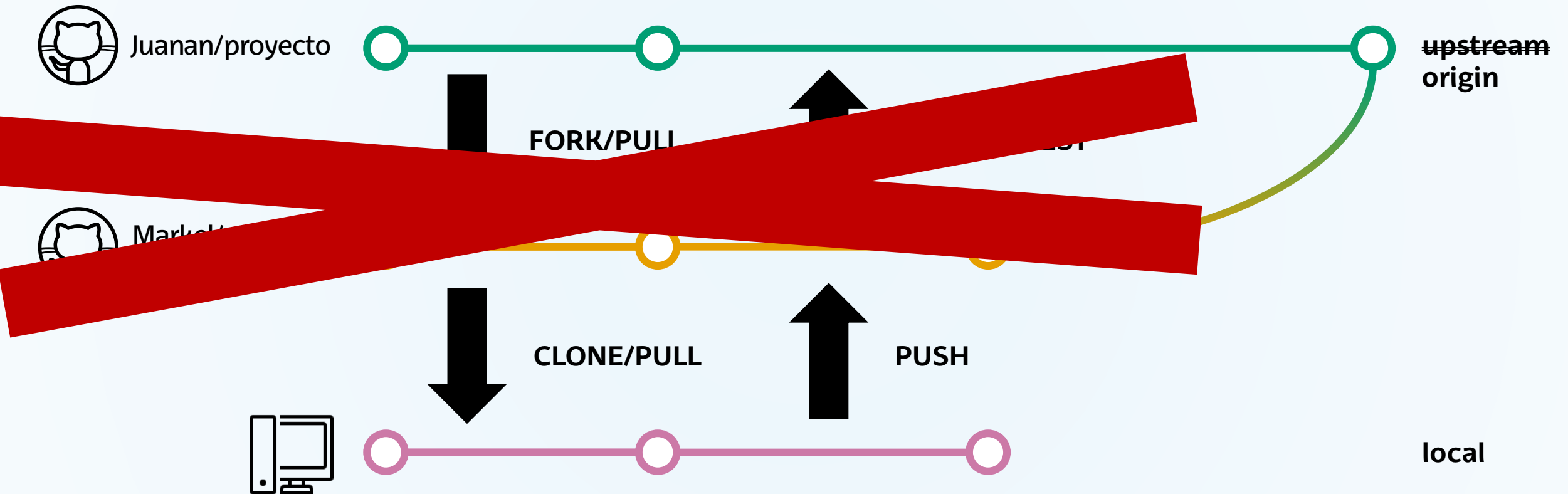
**Ahora entra:**



# Upstream vs origin vs local



# Cuando trabajas en tu propio repositorio





Cuando no hay upstream

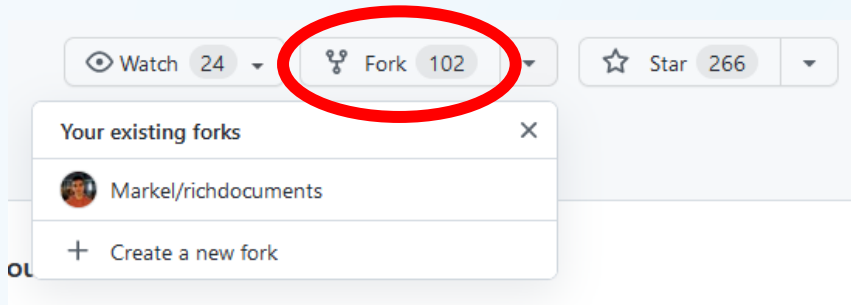
**Solución?**

En vez fork usar branches

# Operaciones

## Fork

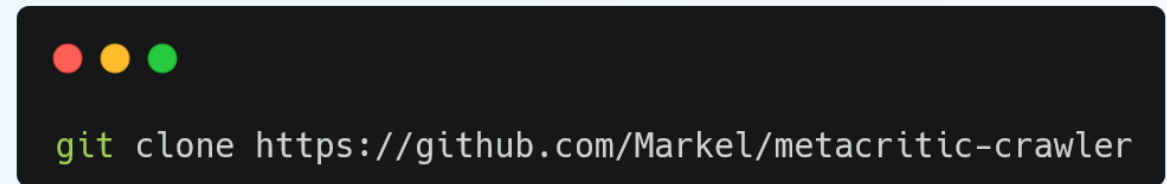
Copiar el repositorio de otra persona a tu propio perfil



El/La autor(a) evita que otra persona escriba en su repositorio.

## Clone

Descargar un repositorio (normalmente de tu perfil) a tu ordenador



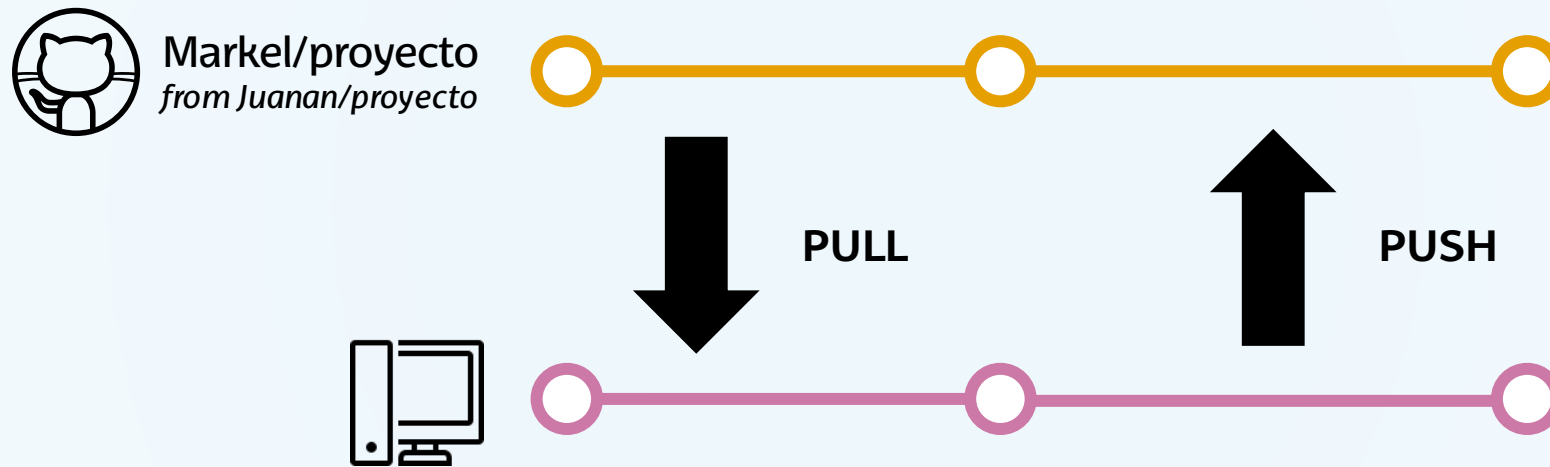
## Pull

Análogo de **descargar**

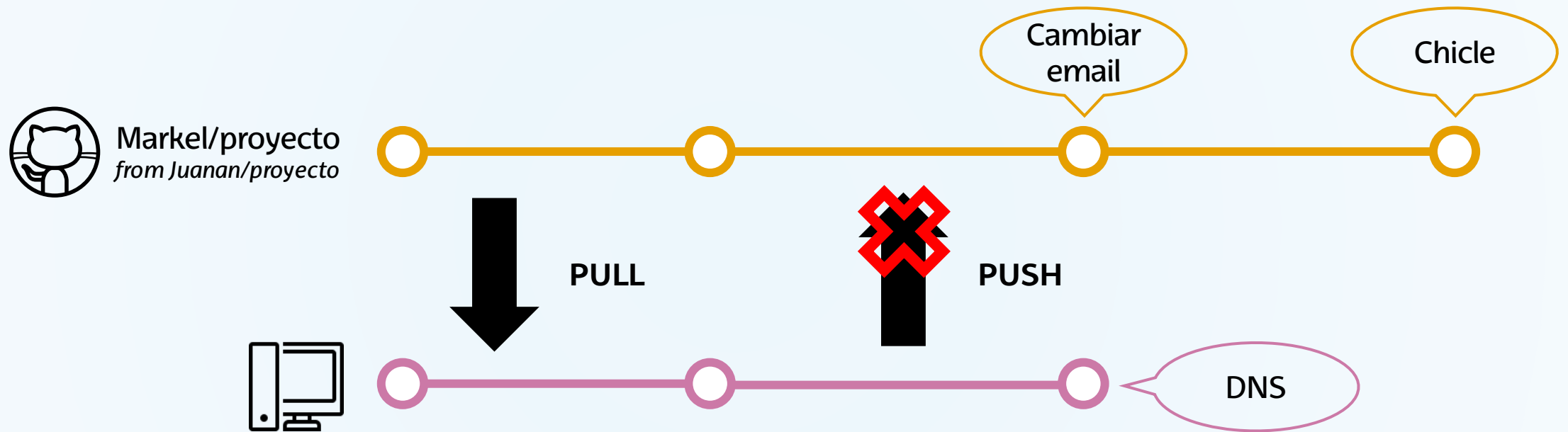
## Push

Análogo de **subir** (upload)

# Pull y push



# Pull y push

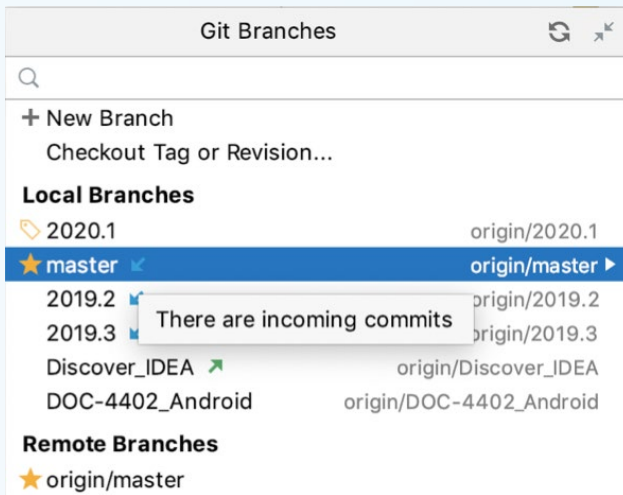


```
! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/proyecto.git'
```

# Sincronizar origin y local



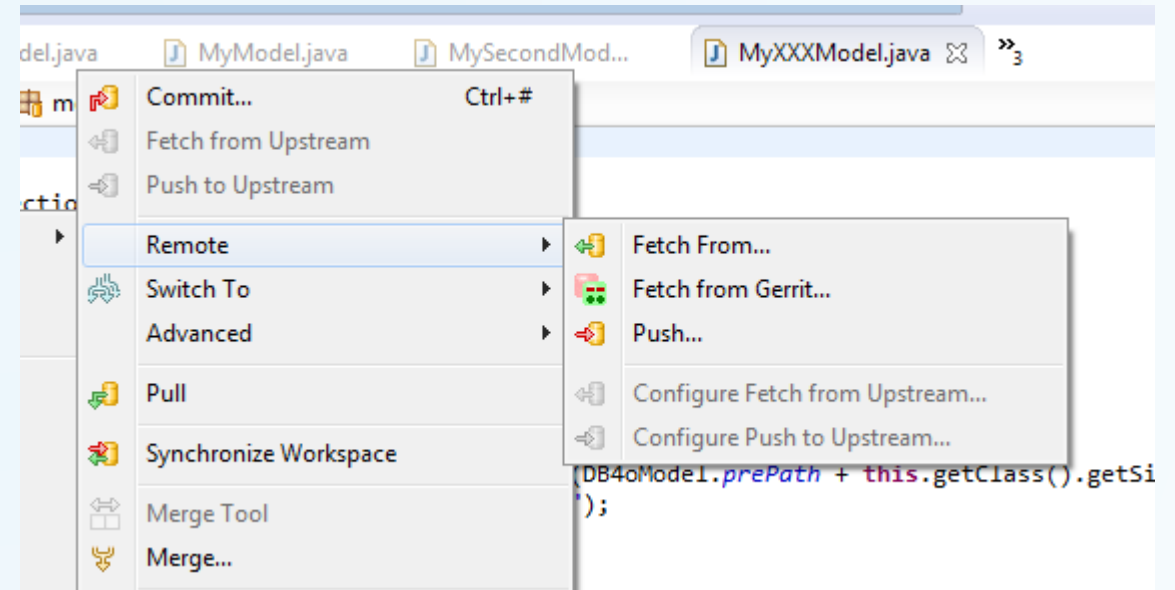
(a) Consola



(c) IntelliJ



(b) VSCode








(d) Eclipse


# Hacer pull de upstream

This branch is **1261 commits behind** nextcloud:main.

#350 Sync fork ▾

 **dependabot[bot]** Merge pull request [nextcloud#1779](#) from nextcloud/depe... ▾

 .github	Create fixup.yml
 .tx	Update backport
 appinfo	Bump version to 4.2.3
 assets	Very minimal document templates for Collab

 **This branch is out-of-date**

Update branch to keep this branch up-to-date by syncing 1261 commits from the upstream repository.

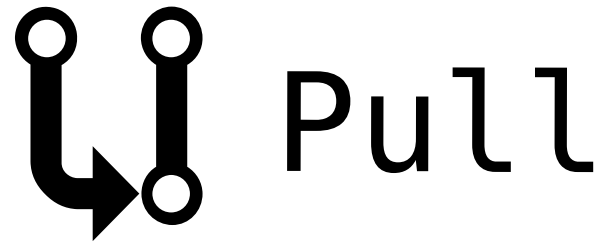
[Learn more](#)

Compare

Update branch



# Una solución para sincronizar upstream y origin



Keep your forks up-to-date with automated PRs

Incorporate new changes as they happen, not in 6 months.

[instalado](#)

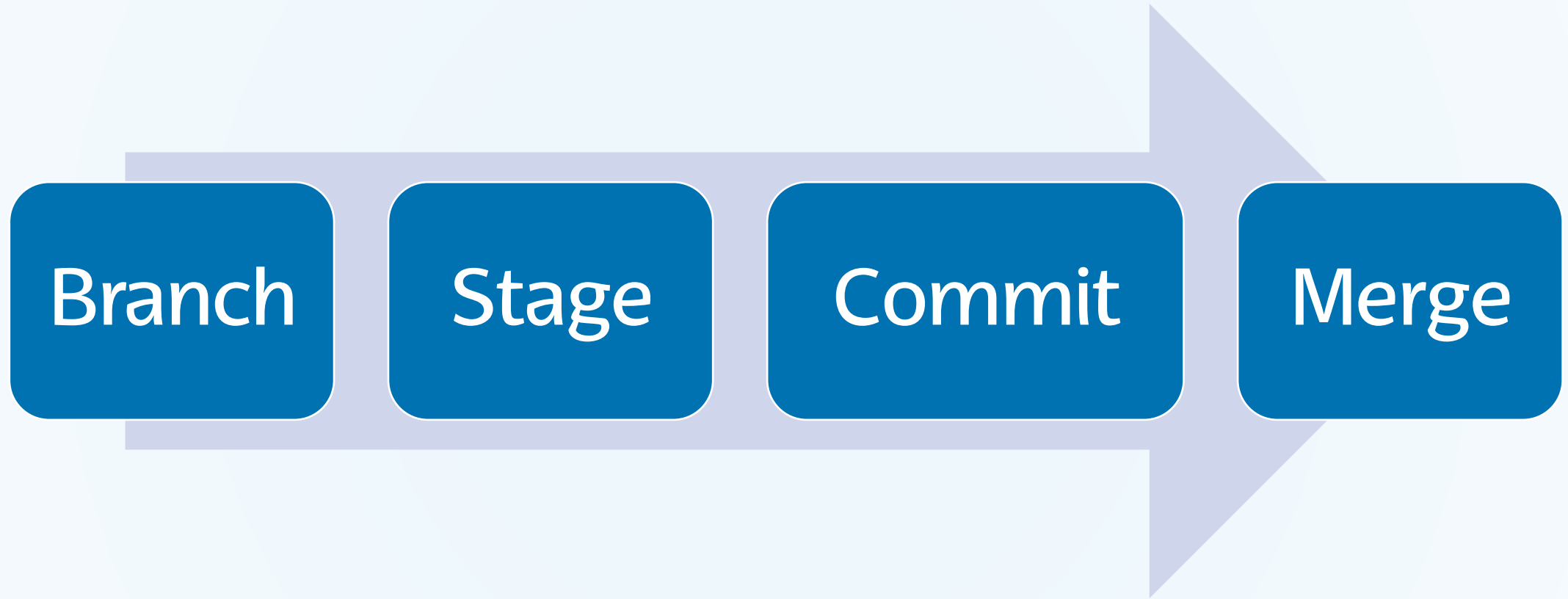
+

branching en origin

+

no tocar el main en origin

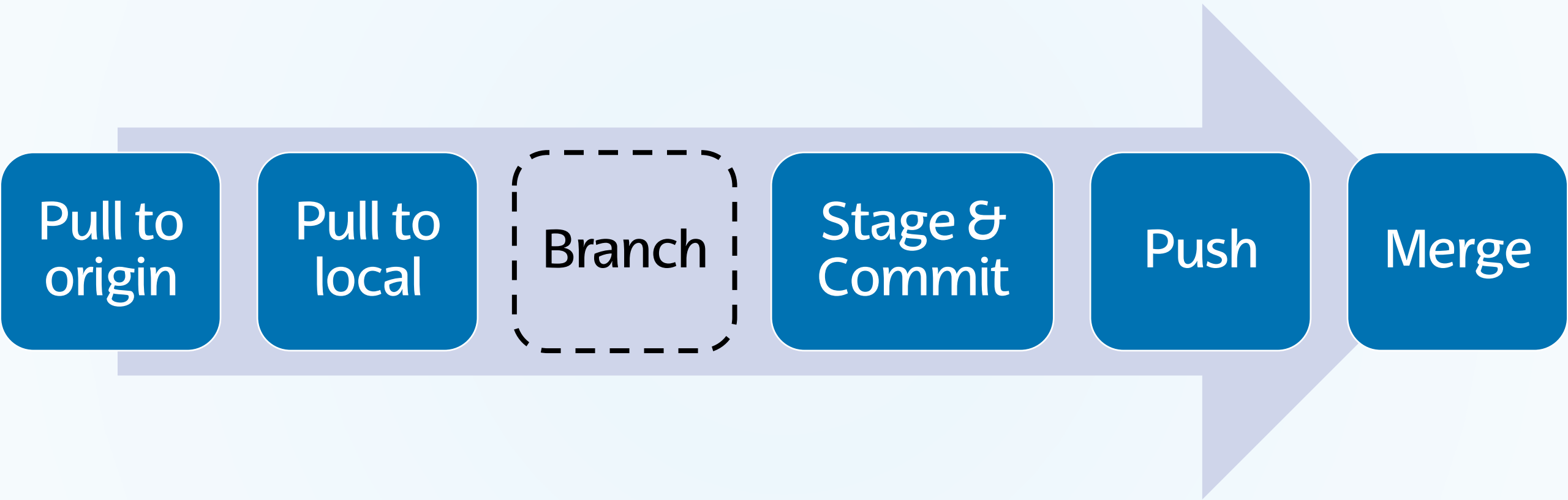
# El proceso







# El proceso

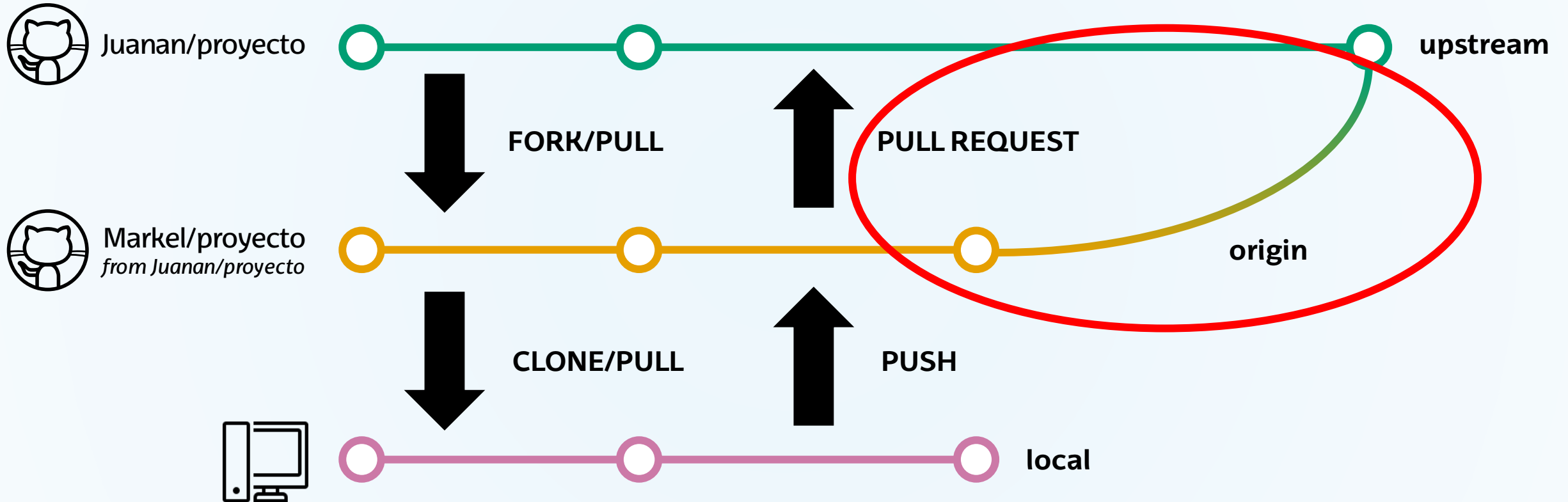




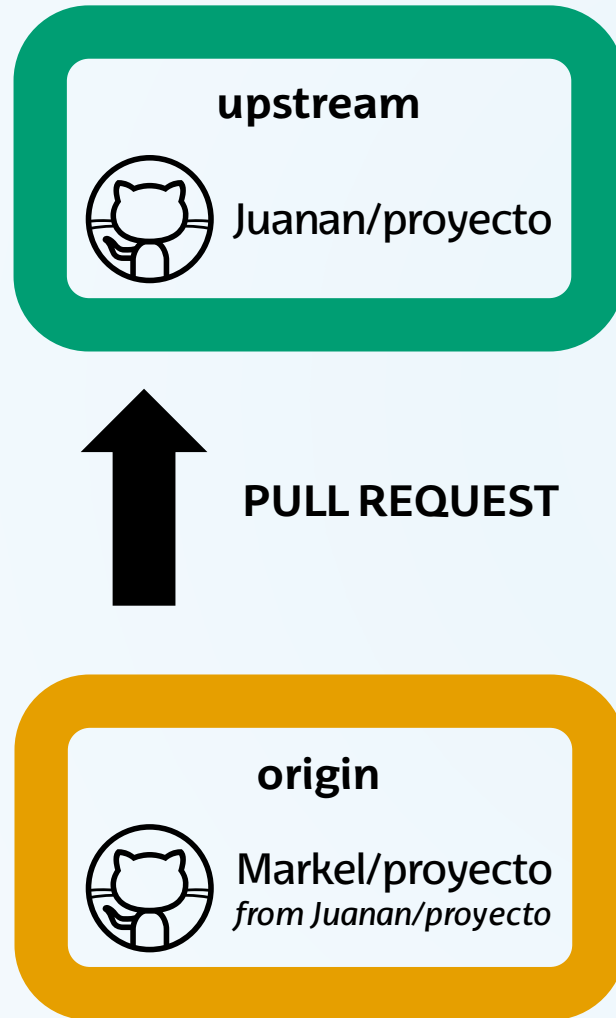
# Dena ondo?



# Upstream vs origin vs local

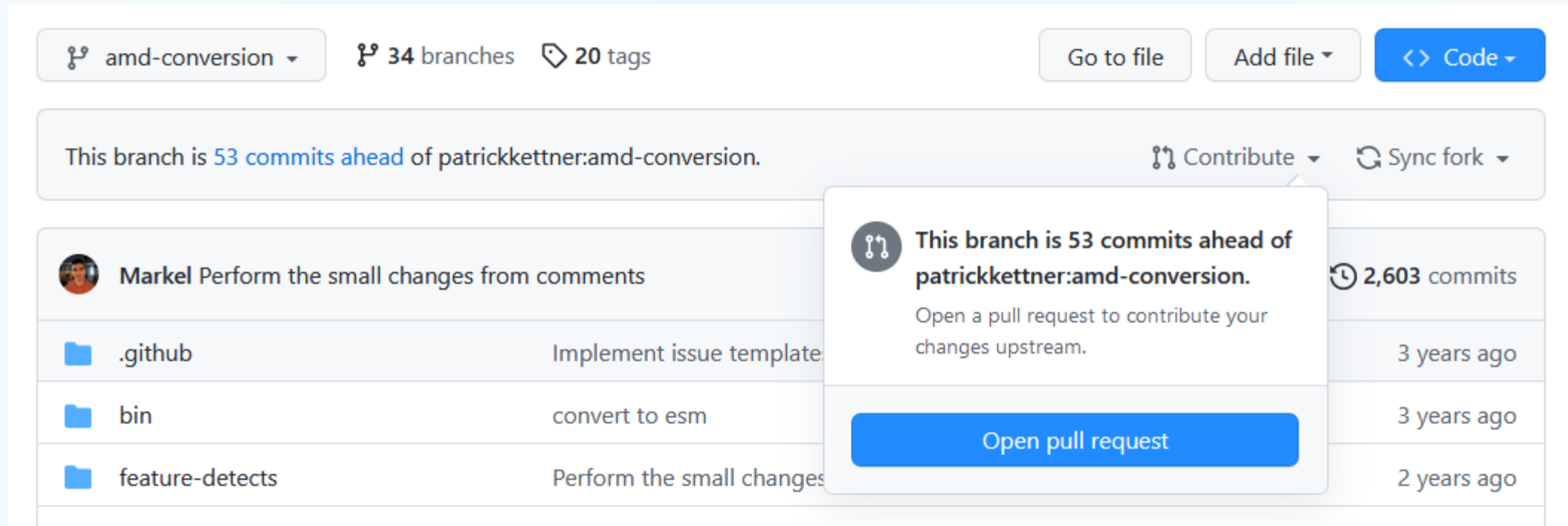


# ¿Por qué un proceso diferente?



- Democratizar
- Seguridad
- Validación

# Cómo hacer una PR



The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with a dropdown menu set to 'amd-conversion', showing '34 branches' and '20 tags'. To the right are buttons for 'Go to file', 'Add file', and 'Code'. Below this, a status bar indicates 'This branch is 53 commits ahead of patrickkettner:amd-conversion.' with 'Contribute' and 'Sync fork' options. A modal dialog is open, repeating the status and providing an 'Open pull request' button. The background shows a commit history table with three entries.


Commit	Author	Message	Time
2,603	patrickkettner	amd-conversion	3 years ago
2,602	patrickkettner	amd-conversion	3 years ago
2,601	patrickkettner	amd-conversion	2 years ago

Repository details: amd-conversion, 34 branches, 20 tags. Buttons: Go to file, Add file, Code. Status: This branch is 53 commits ahead of patrickkettner:amd-conversion. Actions: Contribute, Sync fork. Modal: This branch is 53 commits ahead of patrickkettner:amd-conversion. Open a pull request to contribute your changes upstream. Button: Open pull request. Commit history: 2,603 commits (3 years ago), 2,602 commits (3 years ago), 2,601 commits (2 years ago).

# Cómo hacer una PR

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

 base repository: patrickettner/Modernizr


base: amd-conversion

...

head repository: Markel/Modernizr-v4

compare: amd-conversion








✓ **Able to merge.** These branches can be automatically merged.




Amd conversion


Write

Preview


H B I  <>     @  


Leave a comment


Attach files by dragging & dropping, selecting or pasting them. 


☒ **Allow edits and access to secrets by maintainers** 

Create pull request

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

 Commits 53

 Files changed 77

 Commits on Apr 28, 2020

✓ **Create pull request**  
Open a pull request that is ready for review

**Create draft pull request**  
Cannot be merged until marked ready for review



# Cómo hacer una PR, técnicamente...



```
git request-pull [-p] <start> <URL> [<end>]
```

# Cómo queda

## Added option to remove partially watched videos from the 'Whats new' feed #9747

<> Code ▾



Jared234 wants to merge 5 commits into `TeamNewPipe:dev` from `Jared234:9126_remove_partially_watched_from_feed`



Conversation 9



Commits 5



Checks 5



Files changed 9

+105 -89



Jared234 commented 2 weeks ago

Contributor



### What is it?

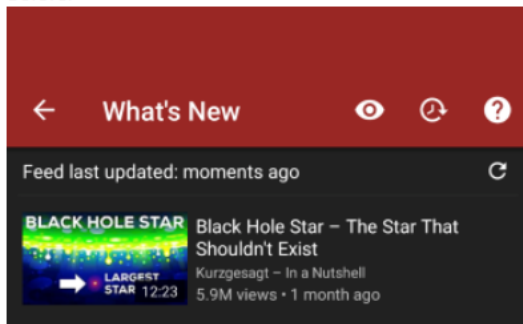
- ☐ Bugfix (user facing)
- ☒ Feature (user facing)
- ☐ Codebase improvement (dev facing)
- ☐ Meta improvement to the project (dev facing)

### Description of the changes in your PR

- added the option to remove partially watched streams from the "What's new" feed
- the options to hide played, partially played or future items have been merged into one checkbox menu

### Before/After Screenshots/Screen Record

- Before:



### Reviewers



Requested changes must be addressed to merge this pull request.

### Assignees

No one assigned

### Labels

feature request feed GUI

### Projects

None yet

### Milestone

No milestone

### Development

Successfully merging this pull request may close these issues.

Option to remove partially watched videos from...



# Resumen

**Upstream:** Proyecto original

**Origin:** Copia online del proyecto

**Local:** Copia offline de tu copia

**Fork**

Primera copia (upstream  
→ origin)

**Clone**

Primera copia (origin →  
local)

**Pull**

Copiar de arriba a abajo

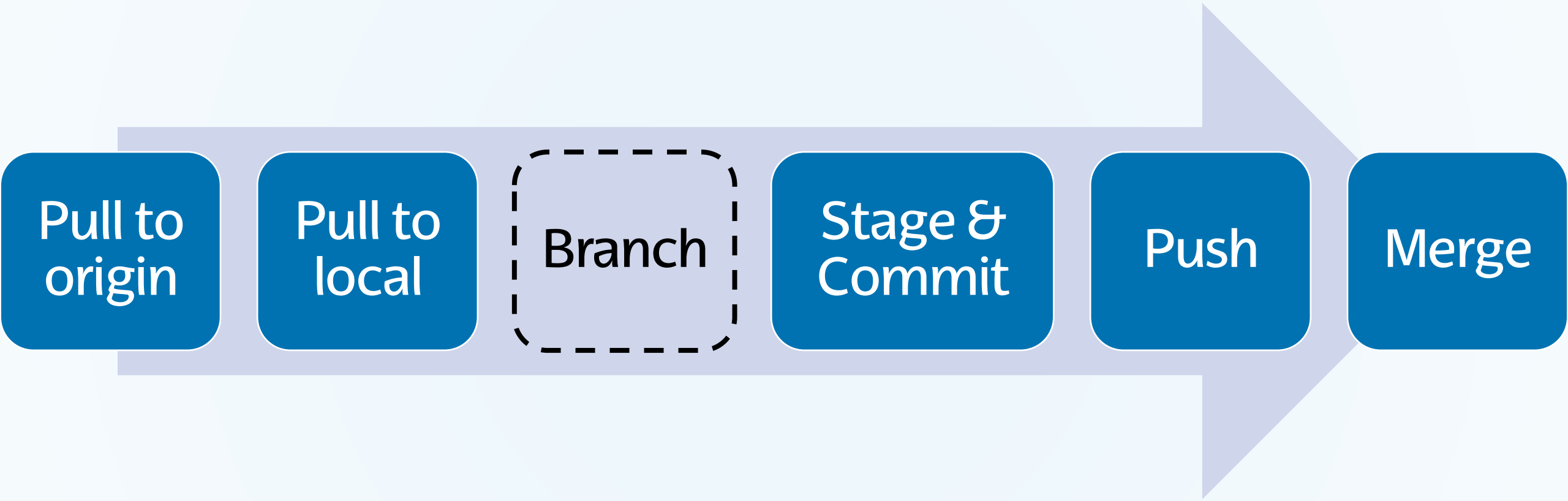
**Push**

Copiar de abajo a arriba

**Pull request**

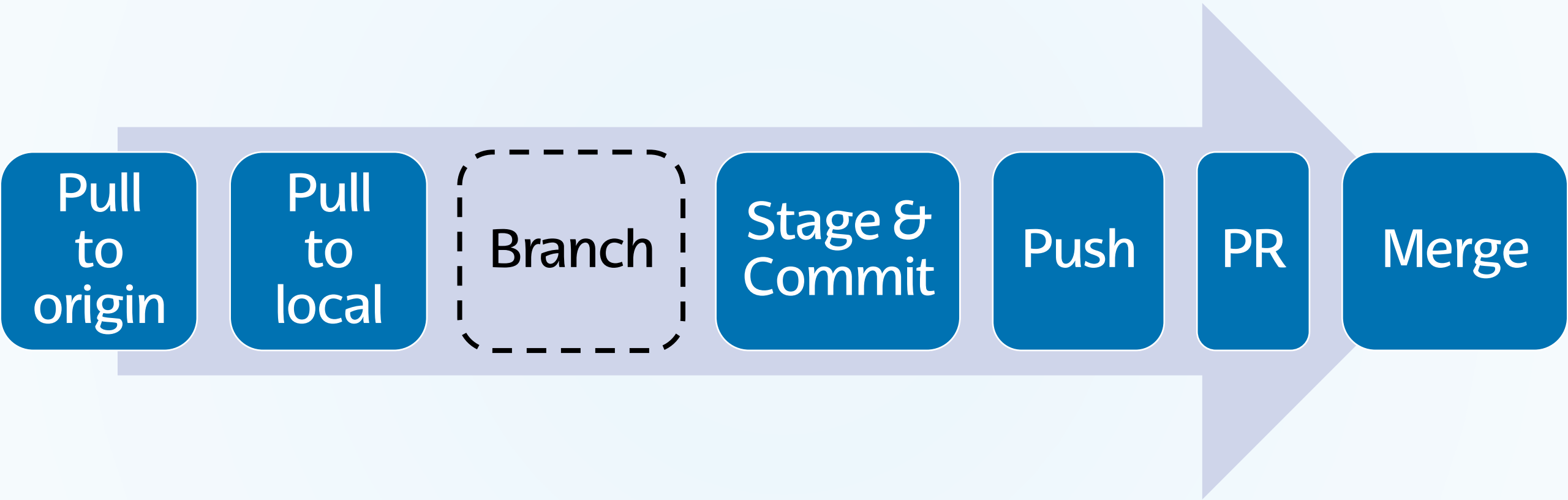
Copiar de origin a upstream

# El proceso



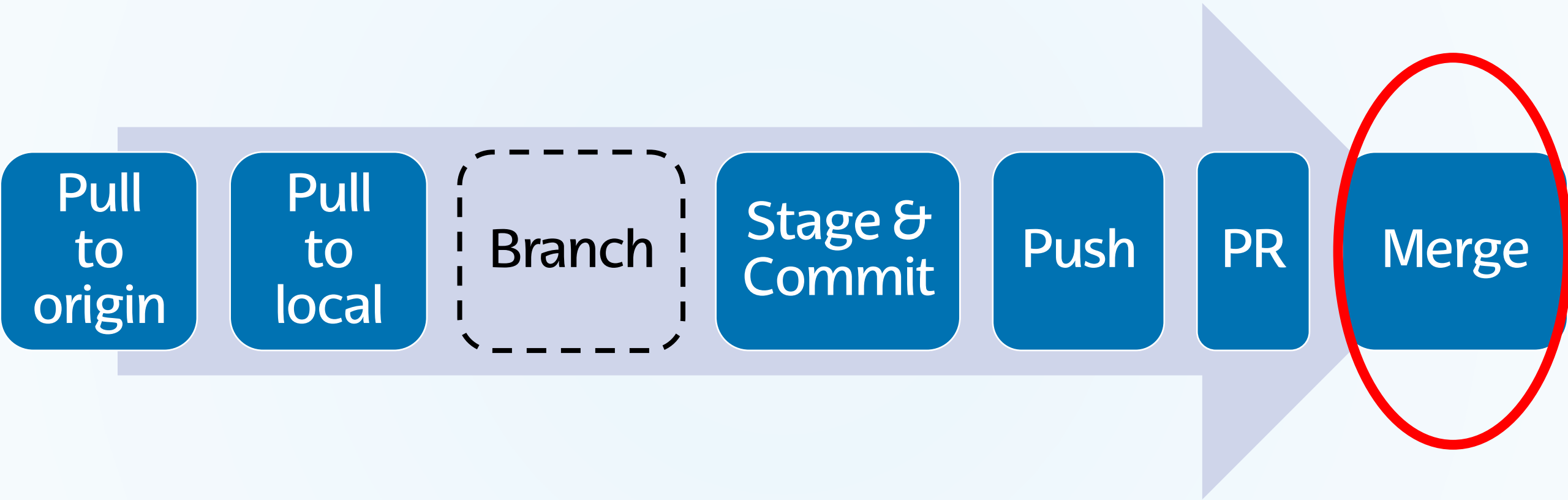


# El proceso





# El proceso



# Cómo hacerlo



**This branch has no conflicts with the base branch**

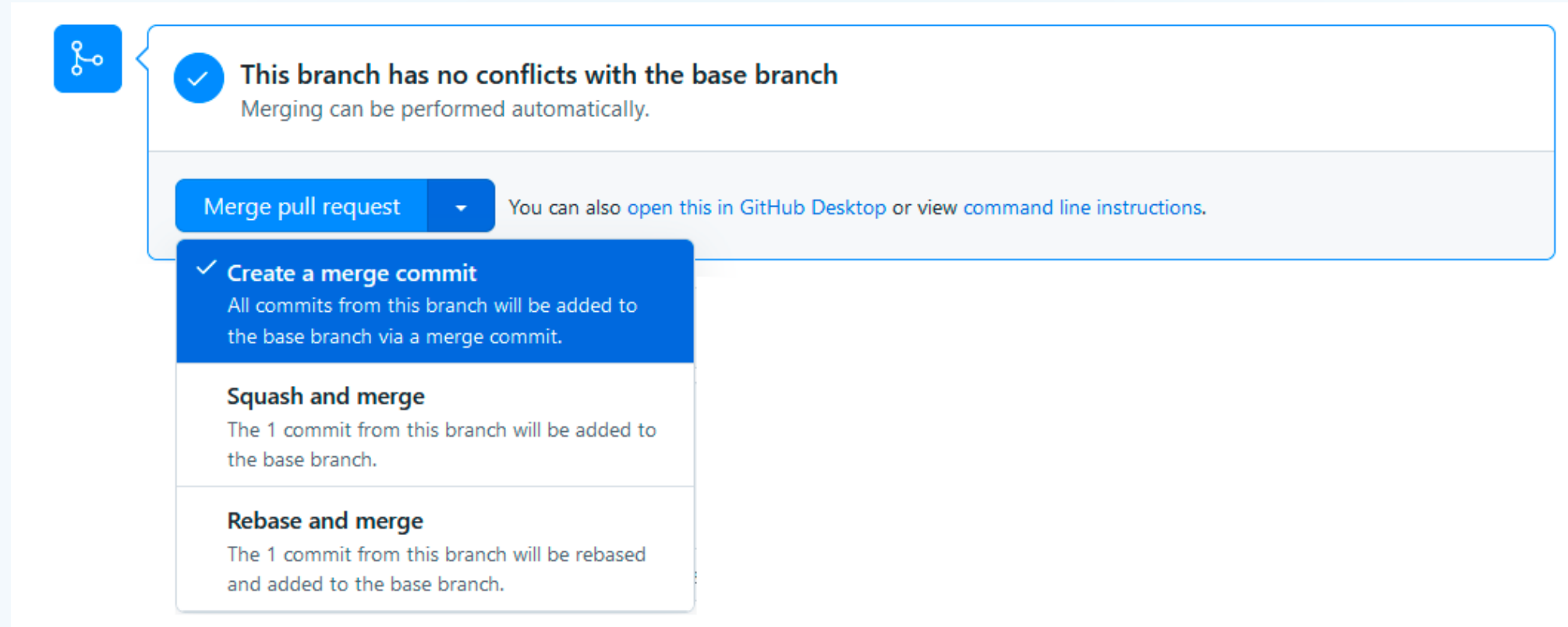
Merging can be performed automatically.

Merge pull request




You can also [open this in GitHub Desktop](#) or [view command line instructions](#).

# Tipos de merge



The image shows a GitHub interface for merging a pull request. At the top left is a blue square icon with a white branching diagram. To its right is a white box with a blue border containing a green checkmark icon and the text "This branch has no conflicts with the base branch" and "Merging can be performed automatically." Below this is a grey bar with a blue button labeled "Merge pull request" and a dropdown arrow, followed by the text "You can also open this in GitHub Desktop or view command line instructions." Below the grey bar is a blue box with a green checkmark icon and the text "Create a merge commit" and "All commits from this branch will be added to the base branch via a merge commit." Below the blue box are two white boxes with blue borders. The first is titled "Squash and merge" and contains the text "The 1 commit from this branch will be added to the base branch." The second is titled "Rebase and merge" and contains the text "The 1 commit from this branch will be rebased and added to the base branch."

 **✓ This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request** ▾ You can also open this in [GitHub Desktop](#) or [view command line instructions](#).

**✓ Create a merge commit**  
All commits from this branch will be added to the base branch via a merge commit.

**Squash and merge**  
The 1 commit from this branch will be added to the base branch.

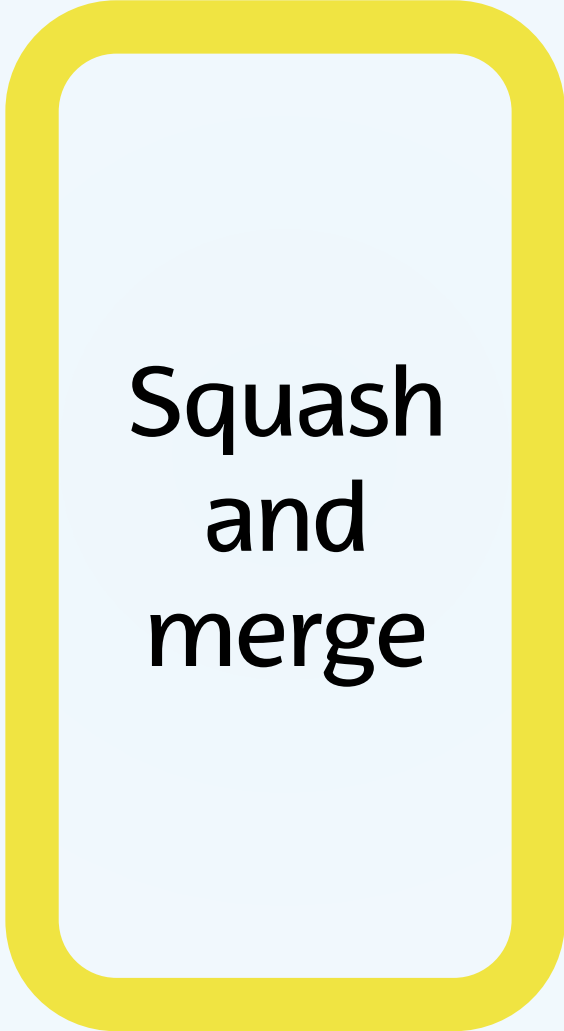
**Rebase and merge**  
The 1 commit from this branch will be rebased and added to the base branch.



# Tipos de merge

A large blue rounded rectangle with a thick border, containing the word "Merge".

Merge

A large yellow rounded rectangle with a thick border, containing the text "Squash and merge".

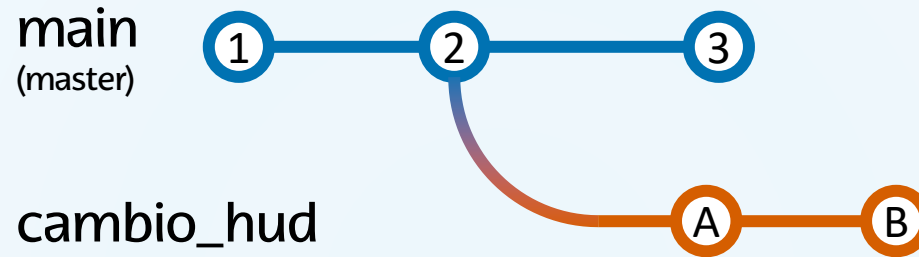
Squash  
and  
merge

A large pink rounded rectangle with a thick border, containing the word "Rebase".

Rebase

# Tipos de merge

Merge

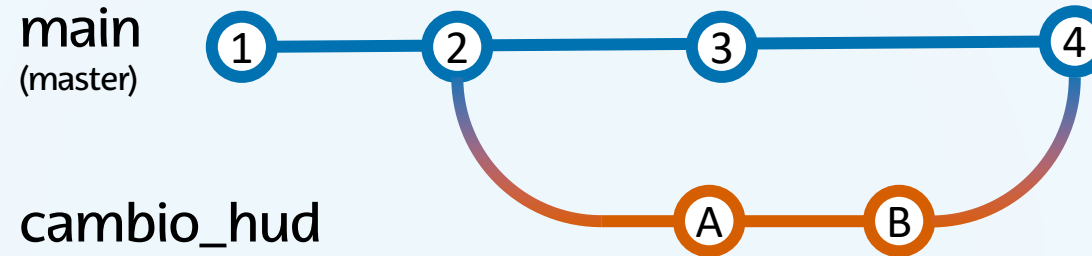






# Tipos de merge

## Merge

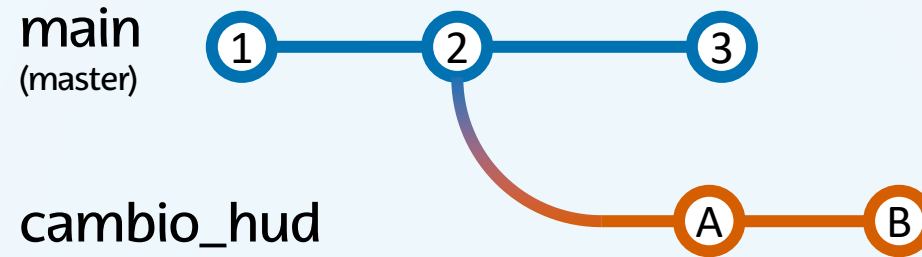


### Explicación:

- Crea un commit que hace referencia a los cambios de la otra branch.
- Deja un historial un poco lioso, pero puedes hacer referencia a todos los commits.

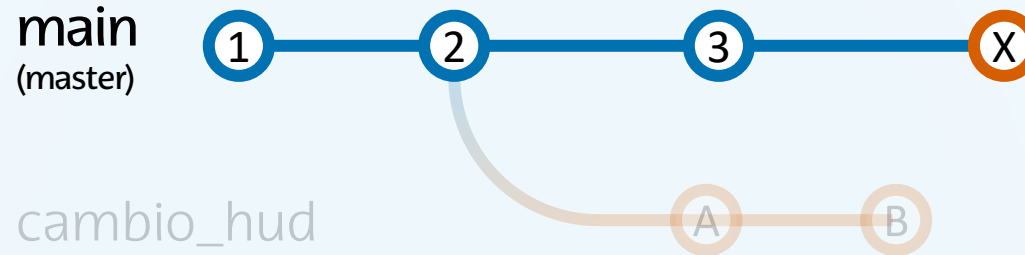
# Tipos de merge

Squash  
and  
merge



# Tipos de merge

## Squash and merge

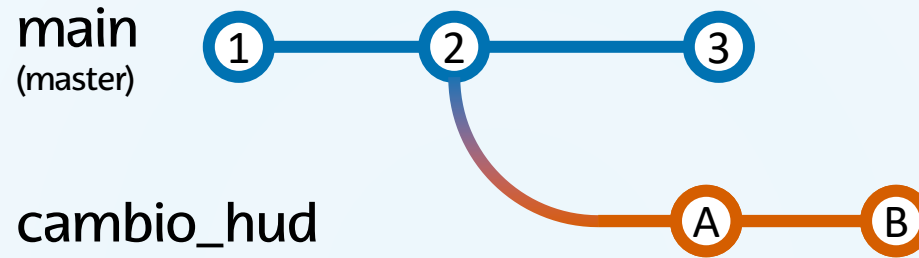


### Explicación:

- Junta todos los commits de la branch en un solo commit. Muy útil con compañeras que hacen 300 commits.
- Pierde esa granularidad de poder ir commit a commit.

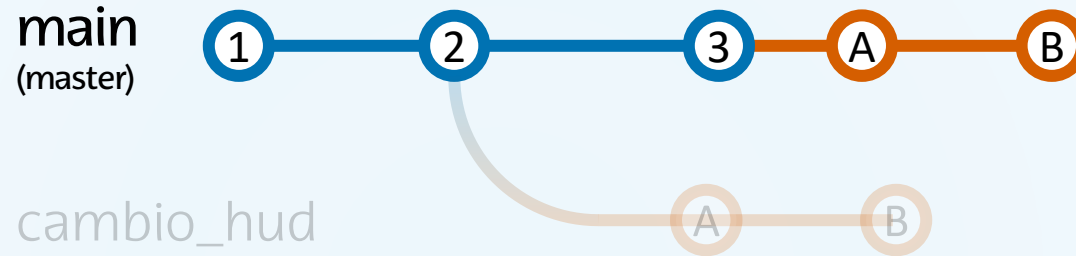
# Tipos de merge

Rebase



# Tipos de merge

## Rebase



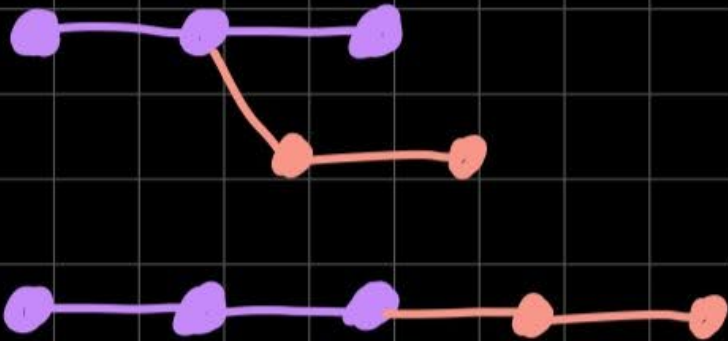
### Explicación:

- Coge los commits de la branch y los mueve a la otra branch.
- Muy útil para merges grandes y complejos, pero requiere aprender cómo hacer rebase correctamente.

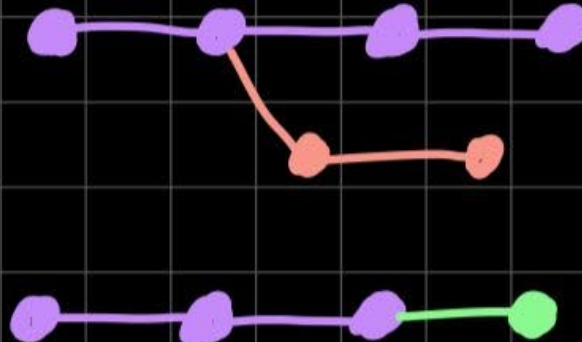


# Tipos de merge

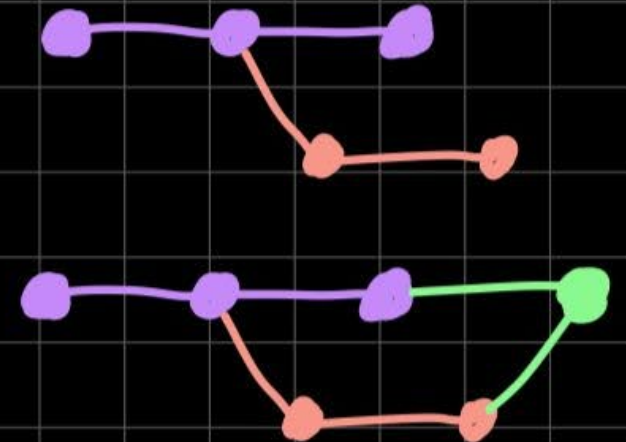
REBASE



SQUASH + MERGE



MERGE



Resumen de Matt Rickard

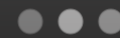
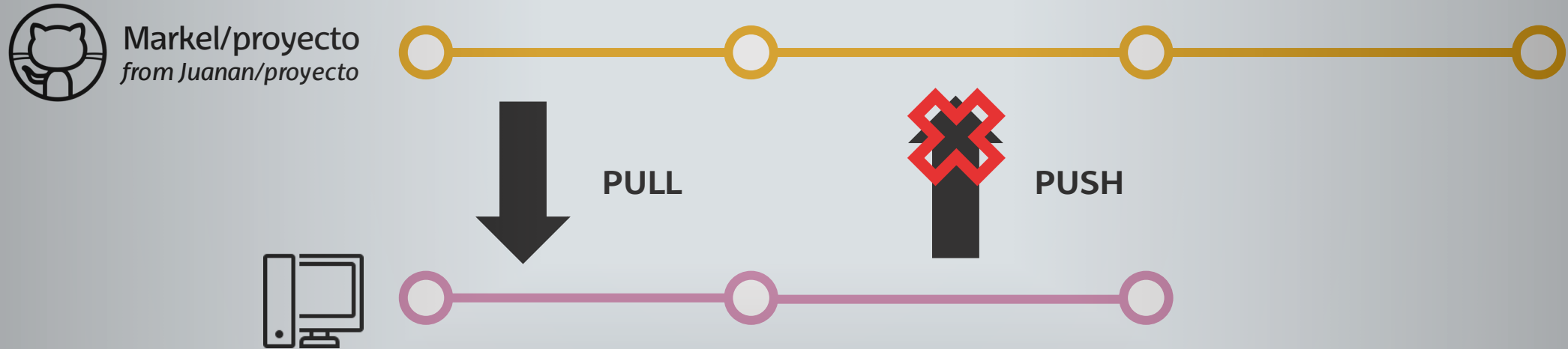


# Mi solución

**Squash and merge**

Porque trabajo en cosas pequeñas

# Pull y push



```
! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/proyecto.git'
```





Cuando ambos cambiáis el mismo archivo...

Merge conflicts

(Nota: Estas son las diapositivas por las que dejo la presentación a descargar)

# Merge conflicts



## This branch has conflicts that must be resolved

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Resolve conflicts

### Conflicting files

readme.md

Merge pull request



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



# Cuando las cosas salen mal...

```
myfile.html


<html>
<body>
  <h1>Izenburua</h1>


  <<<<<< cambio_hub
    <h2 id="hi">Nire subtitulua</h2>
  =====
    <h2>Nire subtitulua</h2>
  >>>>>> master

  <p>Eta textu pixka bat.</p>
</body>
</html>
```

Elegir que dejar y que quitar (o hacer algo intermedio)

# Cómo solucionarlos



 **This branch has conflicts that must be resolved**  
Use the [web editor](#) or the [command line](#) to resolve conflicts.

**Conflicting files**


`readme.md`

[Merge pull request](#) ▼ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

[Resolve conflicts](#)



# Cómo solucionarlos

 [simkimsia / resolve-merge-conflict](#)

Unwatch 1

Star 0

Fork 0

<> Code

Issues 0

**Pull requests 1**

Projects 0


Wiki

Insights

Settings

#2: we adding section 2 instead > Conflicts

1 conflicting file

 content.md

content.md

content.md

1 conflict

Prev ^

Next v

⚙

Mark as resolved

1

# Content

2

3

<<<<<< feature/add-section2

4

## Section 2

5

6

We adding section 2 deliberately instead.

7

=====

8

# Section 1

9

10

This is section 1

11

>>>>>> master

12



# Cuando no te deja



## This branch has conflicts that must be resolved

Only those with [write access](#) to this repository can merge pull requests.

### Conflicting files

feature-detects/cookies.js

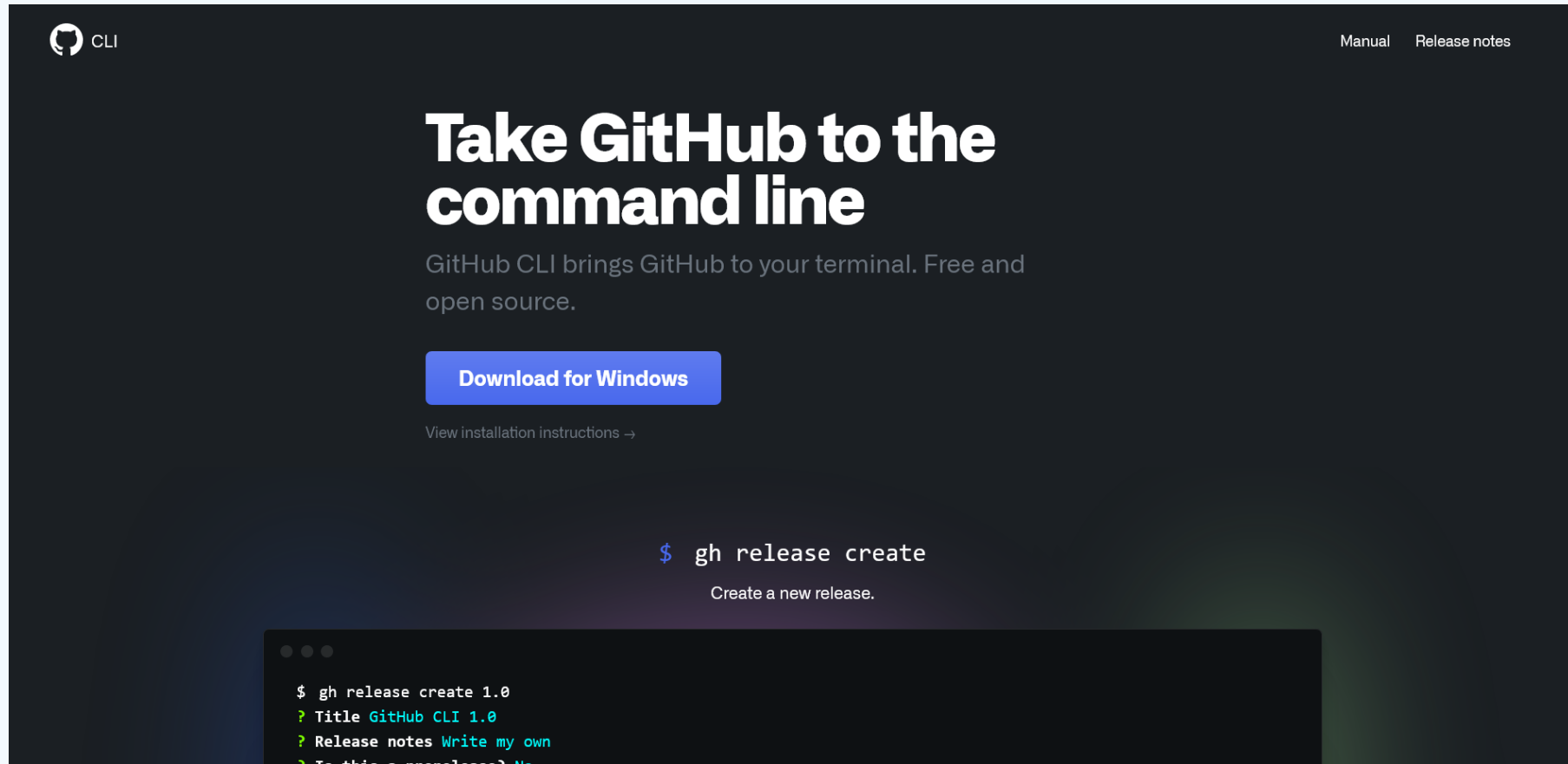
---

Resolve conflicts ?

# Paso 1



# Paso 2 (si quieres hacerte la vida más entendible)



Ventajas:  
Comandos más entendibles

Desventajas:  
Monopolios y el capitalismo



# Paso 3

Elegir en cuál de las dos está  
la PR



(Pista: mirar en la web si aparece un *from*)\*

# Paso 4 (la PR inversa)



**origin**



Markel/proyecto  
*from Juanan/proyecto*

**upstream**



Juanan/proyecto



```
# Para elegir entre origin y
upstream
gh repo set-default

gh pr checkout PR_NUMBER
git merge <origin|upstream>/main
# La branch que es objetivo
```



```
# Solo la primera vez
git config --add remote.origin.fetch +refs/pull/*/merge:refs
/remotes/origin/pr/*

# El resto
git pull
git checkout origin/pr/PR_NUMBER
git merge origin/main # La branch que es objetivo
```



```
# Solo la primera vez
git remote add upstream https://github.com/DUEÑO_ORIGINAL
/REPOSITORIO_ORIGINAL.git
git config --add remote.upstream.fetch +refs/pull/*/merge:refs
/remotes/upstream/pr/*

# El resto
git pull upstream
git checkout upstream/pr/PR_NUMBER
git merge upstream/main # La branch que es objetivo
```

Fuente (sí, Twitter)

# Paso 5

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
550 <<<<<<< HEAD (Current Change)
551 $meta_value = $wpdb->get_var("SELECT meta_value FROM {$wpdb->usermeta} WHERE meta_key = '{$wpdb->prefix}capabilities' AND user_id = {$
552
553 if (!$meta_value) {
554     return array( );
555 }
556
557 $capabilities = unserialize($meta_value);
558 $roles = is_array($capabilities) ? array_keys($capabilities) : array( );
559 =====
560 $usercapabilities = get_user_meta( $uid, "{$wpdb->prefix}capabilities", true);
561 if ( ! is_array( $usercapabilities ) ) {
562     return '';
563 }
564
565 $editable_roles = apply_filters('editable_roles', $wp_roles->roles);
566
567 $userroles = array_keys( array_intersect_key($editable_roles, $usercapabilities) );
568 $role = $userroles[0];
569 >>>>>> master (Incoming Change)
570
```

+ git add . + git commit + git push

# Paso 6 (congrats!)



**This branch has no conflicts with the base branch**

Merging can be performed automatically.

Merge pull request



You can also [open this in GitHub Desktop](#) or [view command line instructions](#).



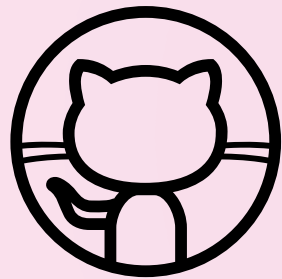
Habiendo  
sobrevivido lo  
duro ahora  
tocan:



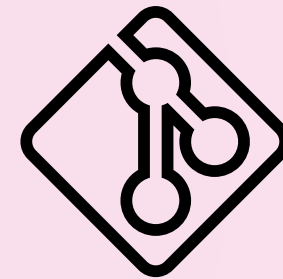
Hau da, cosas inconexas que son útiles



# Github vs Git



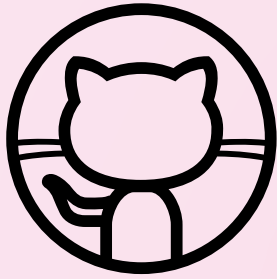
VS



1. Github vs Git
2. Iniciar sesión
3. Github Pro
4. Co-author
5. PR tricks
6. Markdown
7. Ctrl + Z
8. Protected branches



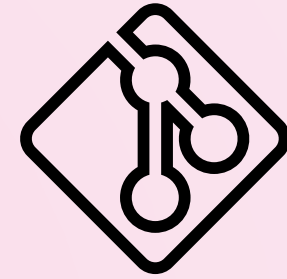
# Github vs Git



Es una compañía

Vende un servicio

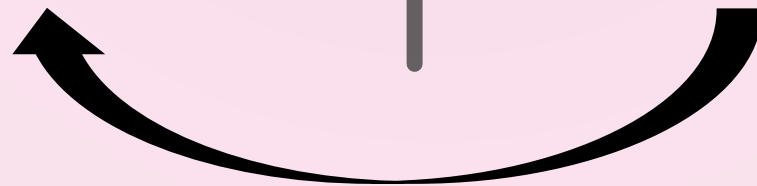
Guarda en la nube los  
resultados de aplicar Git



Es un proyecto

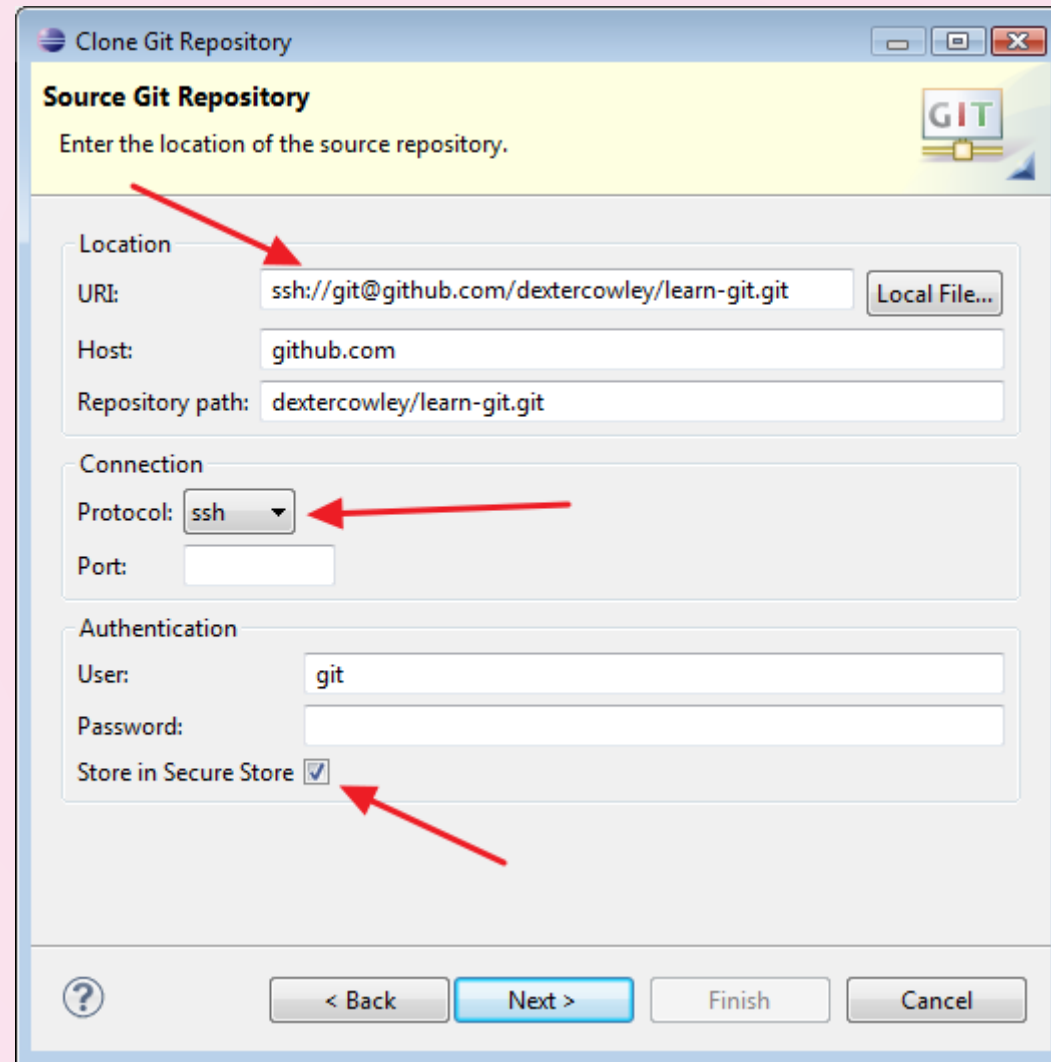
Es un sistema de gestión  
(gratis)

Es el programa que hace todo el  
control de versiones





# Evitar estar loggeandose todo el rato...



The screenshot shows the 'Clone Git Repository' dialog box. It has a yellow header bar with the title 'Clone Git Repository' and a subtitle 'Source Git Repository'. Below the subtitle is the instruction 'Enter the location of the source repository.' and a Git logo. The dialog is divided into three sections: 'Location', 'Connection', and 'Authentication'. In the 'Location' section, the 'URI' field contains 'ssh://git@github.com/dextercowley/learn-git.git', and the 'Repository path' field contains 'dextercowley/learn-git.git'. In the 'Connection' section, the 'Protocol' dropdown menu is set to 'ssh'. In the 'Authentication' section, the 'User' field contains 'git', and the 'Store in Secure Store' checkbox is checked. Red arrows point to the 'URI' field, the 'Protocol' dropdown, and the 'Store in Secure Store' checkbox. At the bottom of the dialog are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

Clone Git Repository

**Source Git Repository**

Enter the location of the source repository.

Location

URI:  Local File...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

Store in Secure Store ☒

? < Back Next > Finish Cancel

1. Github vs Git
2. Iniciar sesión
3. Github Pro
4. Co-author
5. PR tricks
6. Markdown
7. Ctrl + Z
8. Protected branches



# Evitar estar loggeandose todo el rato...

**Paso 1:** Lo primero es hacer que tu ordenador recuerde la contraseña

Windows (por defecto, creo)

```
git config --global credential.helper manager
```

Ubuntu-like

```
sudo apt-get install libsecret-1-0 libsecret-1-dev
cd /usr/share/doc/git/contrib/credential/libsecret
sudo make
git config --global credential.helper /usr/share/doc/git/contrib/credential/libsecret/git-credential-libsecret
```

Mac

```
git config --global credential.helper osxkeychain
```

Fedora-like

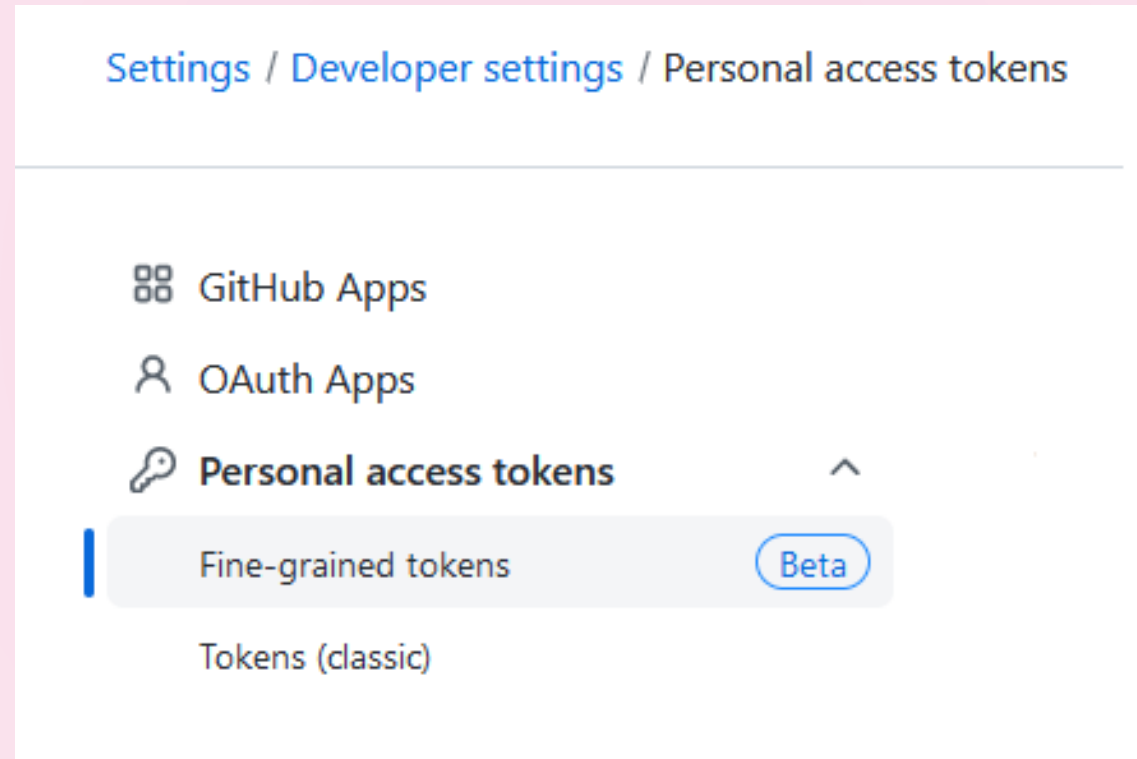
```
sudo dnf install git-credential-libsecret
git config --global credential.helper /usr/libexec/git-core/git-credential-libsecret
```

Guardar en texto plano (no recomendado, pero algunos lo hacemos jeje):

```
git config --global credential.helper store
```

# Evitar estar loggeandose todo el rato...

**Paso 2:** Ir a opciones y generar un token (una contraseña) de acceso personal



**Nota:** Algunas versiones de Git (mainly Windows) te permiten loggearte a través del navegador con lo que puedes saltar este paso e ir al 3

# Evitar estar loggeandose todo el rato...

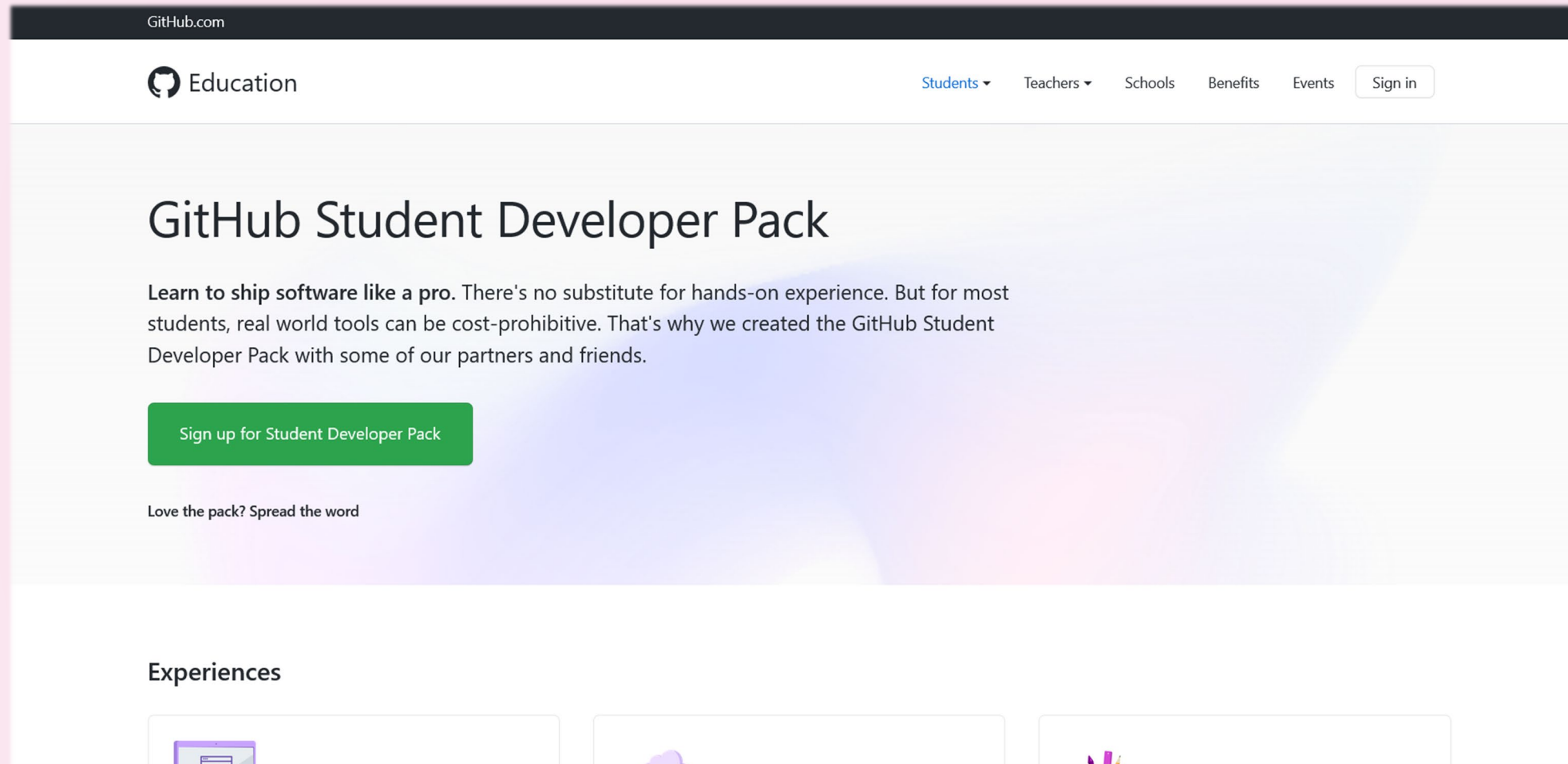
**Paso 3:** Clonar un repositorio privado o inexistente para que te pida la contraseña



```
git clone https://github.com/Markel/bum4-markel
Username: <type your username>
Password: <type your password>

[días después]
git clone https://github.com/Markel/bum4-markel
[no te pide la contraseña]
```

# GitHub Pro y más



<https://education.github.com/pack>

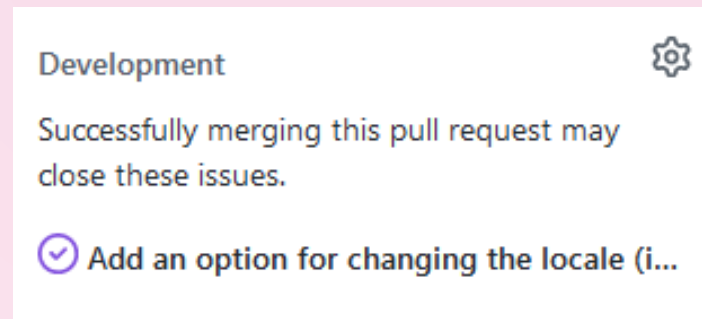
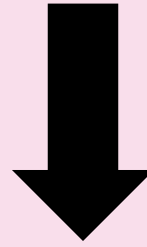
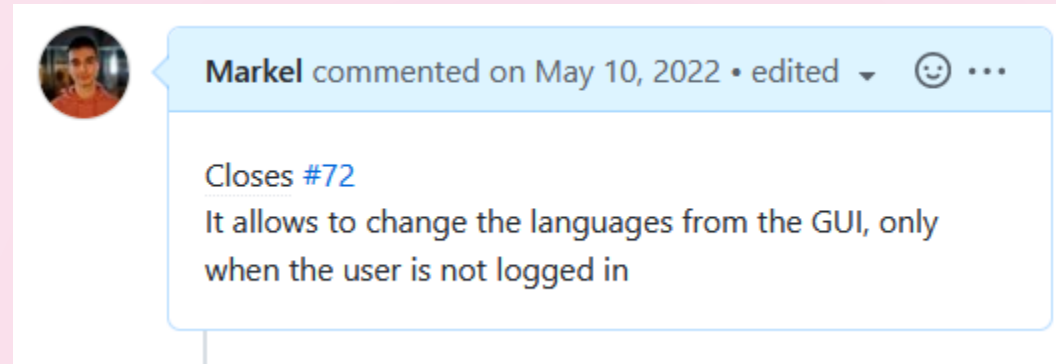


# Trabajo en grupo

Co-authored-by: juananpe <juanan.pereira@ehu.eus>

1. Github vs Git
2. Iniciar sesión
3. Github Pro
- 4. Co-author**
5. PR tricks
6. Markdown
7. Ctrl + Z
8. Protected branches

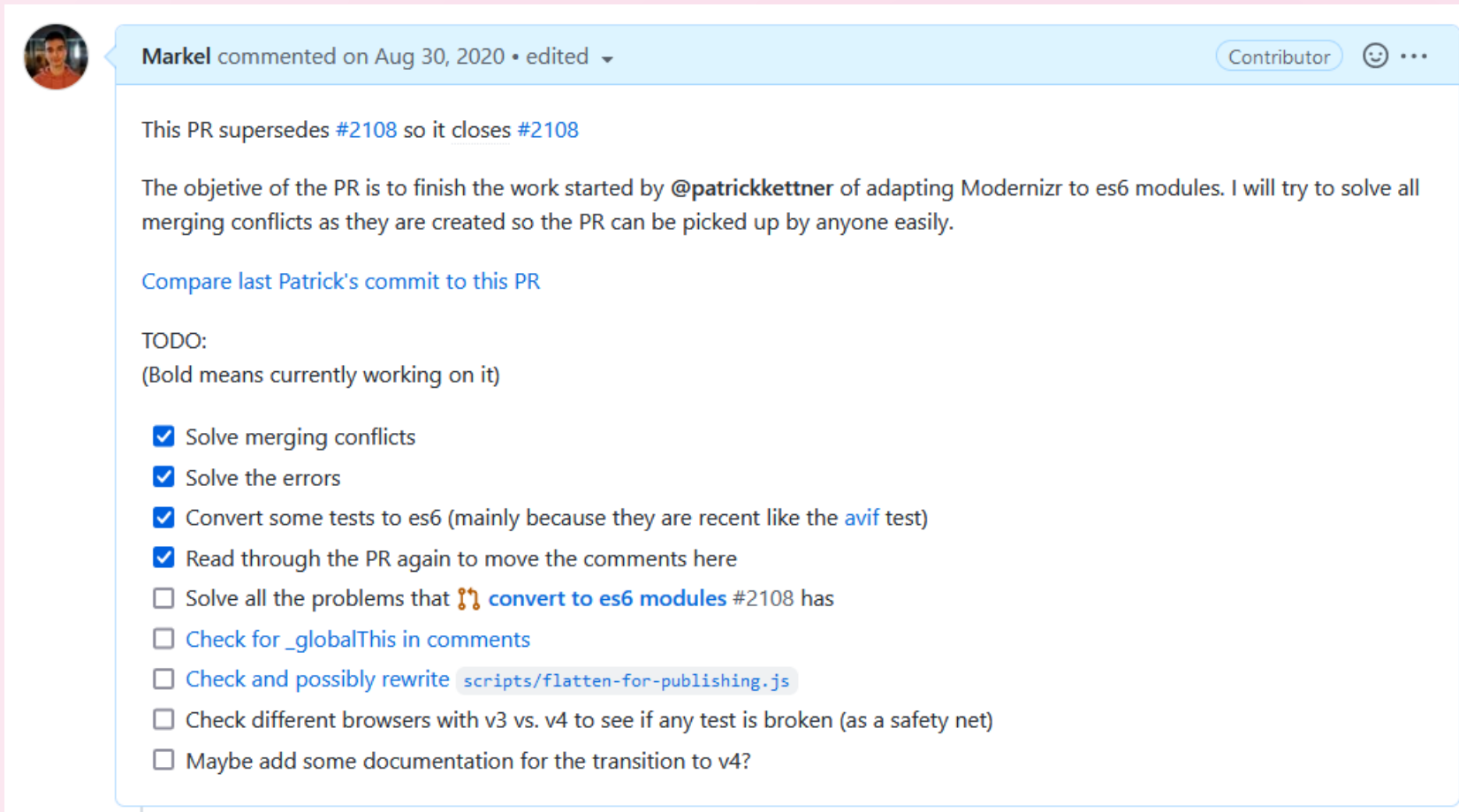
# Cosas guays de las PR (automatic close)



Lista de todas las keywords

1. Github vs Git
2. Iniciar sesión
3. Github Pro
4. Co-author
5. PR tricks
6. Markdown
7. Ctrl + Z
8. Protected branches

# Cosas guays de las PR (listas)



Markel commented on Aug 30, 2020 • edited ▾ Contributor 😊 ⋮

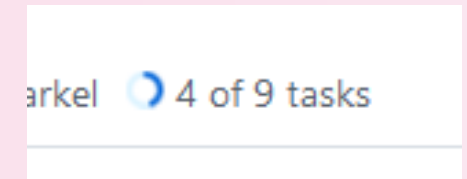
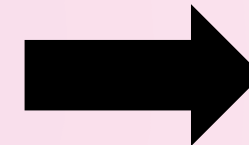
This PR supersedes #2108 so it closes #2108

The objective of the PR is to finish the work started by @patrickkettner of adapting Modernizr to es6 modules. I will try to solve all merging conflicts as they are created so the PR can be picked up by anyone easily.

[Compare last Patrick's commit to this PR](#)

TODO:  
(Bold means currently working on it)

- ☒ Solve merging conflicts
- ☒ Solve the errors
- ☒ Convert some tests to es6 (mainly because they are recent like the [avif](#) test)
- ☒ Read through the PR again to move the comments here
- ☐ Solve all the problems that 🐞 [convert to es6 modules](#) #2108 has
- ☐ Check for `_globalThis` in comments
- ☐ Check and possibly rewrite `scripts/flatten-for-publishing.js`
- ☐ Check different browsers with v3 vs. v4 to see if any test is broken (as a safety net)
- ☐ Maybe add some documentation for the transition to v4?



Markel 4 of 9 tasks

Simplemente usar – [ ] en Markdown



# ¿Quieres aprender Markdown?

<https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>

Porque se utiliza en:



# git commit --amend



```
git add myfile.html
git commit
```

1. Github vs Git
2. Iniciar sesión
3. Github Pro
4. Co-author
5. PR tricks
6. Markdown
7. Ctrl + Z
8. Protected branches

# git commit --amend



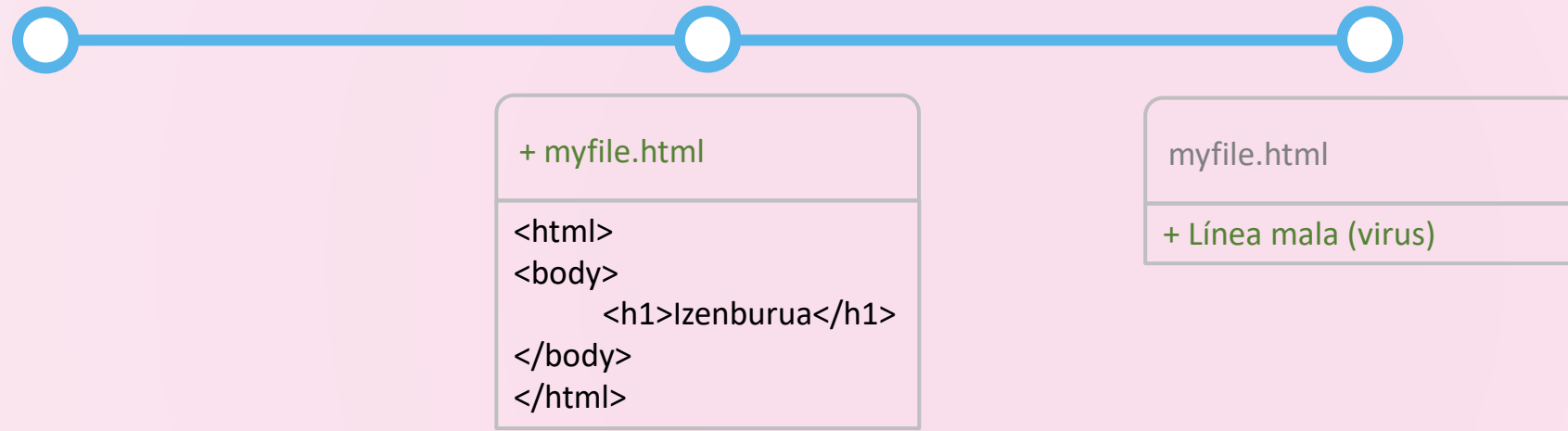
```
git add myfile.html  
git commit
```



```
git add myfile.html  
git commit --amend
```

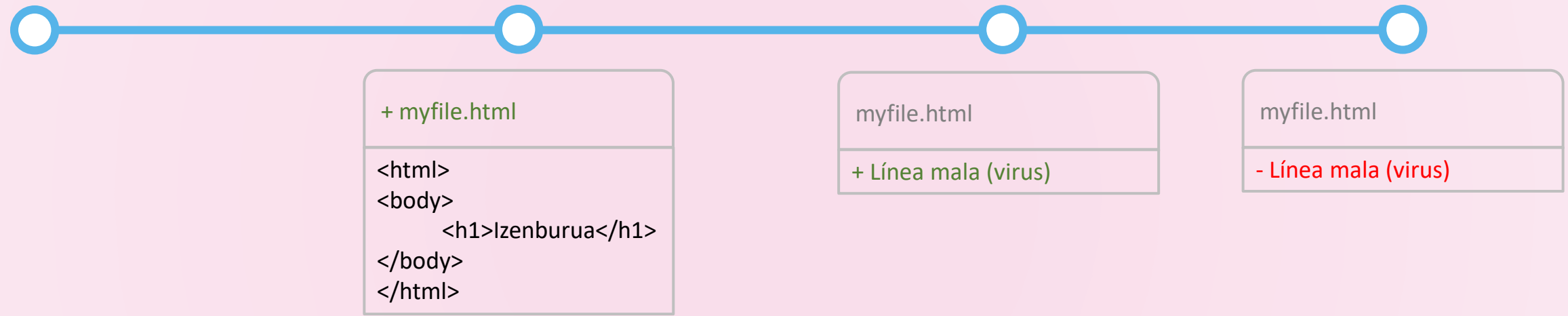
Evitar esto si ya habéis  
hecho push

# git revert



```
git add myfile.html  
git commit  
git push
```

# git revert



```
git add myfile.html  
git commit  
git push
```



```
git revert HEAD
```

Es una operación segura (crea un commit inverso sobre el que haces revert)

En vez HEAD puedes poner el nombre de una branch u otros aspectos

# git revert

```
commit 31ff840a72005754a6647ad017781870a70d6c3a (HEAD -> master)
```

```
Author: Markel <git@markel.dev>
```

```
Date: Tue Feb 14 17:04:07 2023 +0100
```

```
Revert "Revert "Revert "Revert "Revert "Revert "Revert "The inverse""""""""
```

```
This reverts commit 38874c5ea097faf83d3459d593df7a644db28012.
```

```
commit 38874c5ea097faf83d3459d593df7a644db28012
```

```
Author: Markel <git@markel.dev>
```

```
Date: Tue Feb 14 17:04:04 2023 +0100
```

```
Revert "Revert "Revert "Revert "Revert "Revert "Revert "The inverse""""""""
```

```
This reverts commit 1edeb63f40fb1eb5338ecd1d37b7a3e2c56e301c.
```

```
commit 1edeb63f40fb1eb5338ecd1d37b7a3e2c56e301c
```

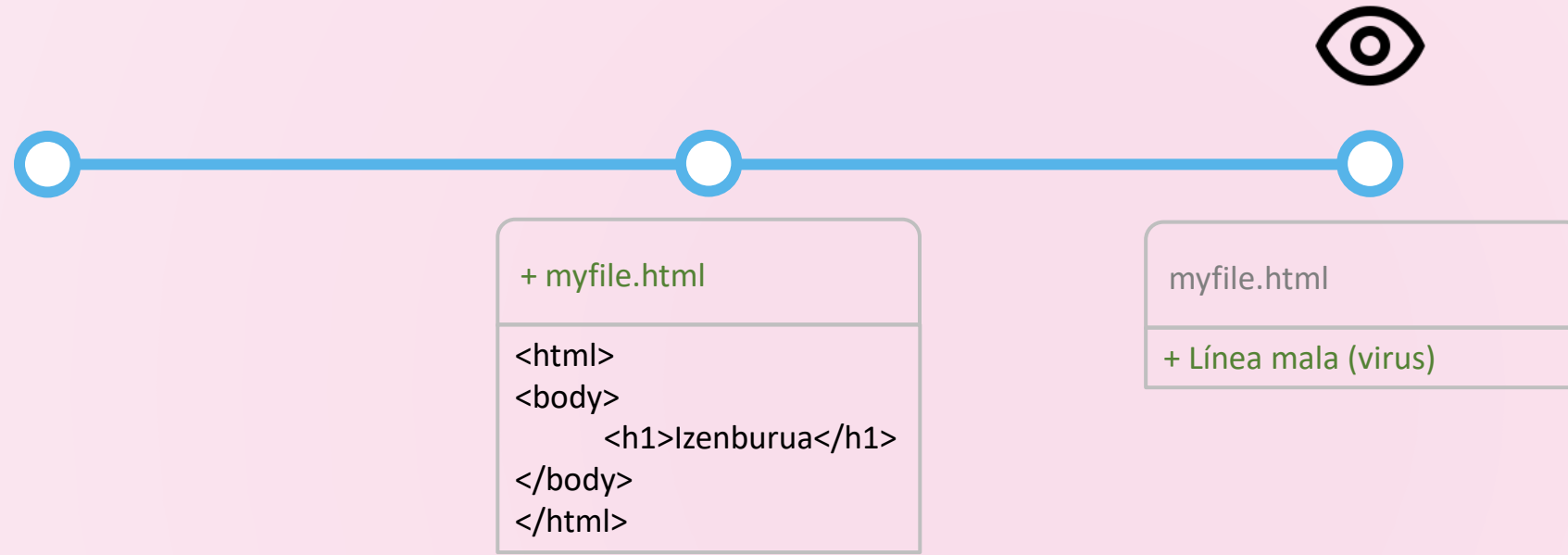
```
Author: Markel <git@markel.dev>
```

```
Date: Tue Feb 14 17:04:01 2023 +0100
```

```
Revert "Revert "Revert "Revert "Revert "Revert "The inverse""""""""
```

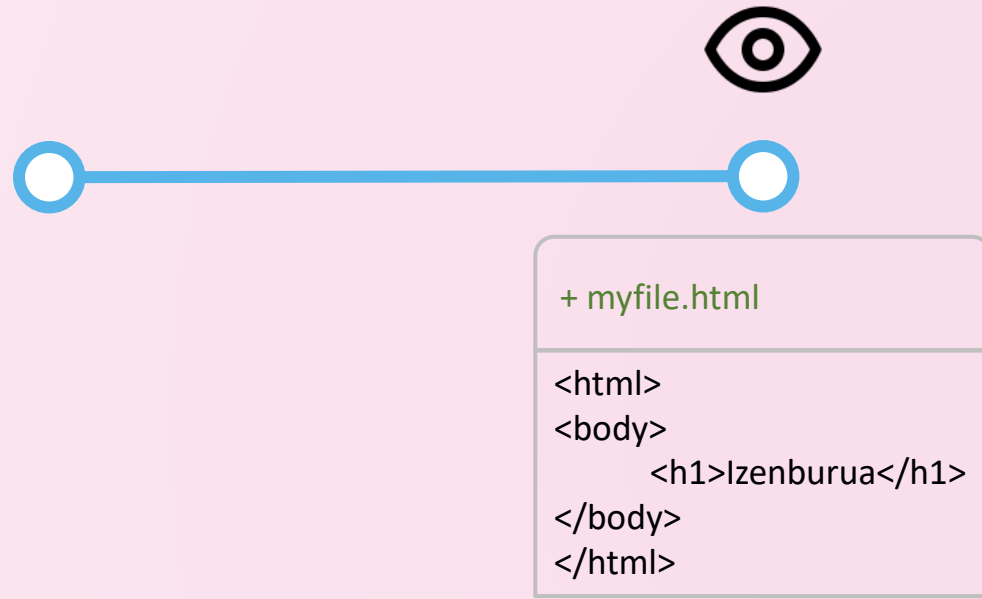
```
This reverts commit 4ab65de9b94d87f91b06a0a6fd01e8936b4362d3.
```

# git reset



```
git checkout HEAD~1
```

# git reset



```
git checkout HEAD~1
```



```
git reset --hard
```

Borra todo lo que haya por  
delante **para siempre**.

Es una operación insegura. Útil  
si has desvelado algún secreto  
o así.



# Resumen

**ammend**

Cambiar algo en tu último commit.

**revert**

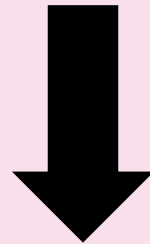
Crear un commit que es inverso al último commit.

**reset**

Borrar todo lo que haya delante.

Ahora vosotros/as sabéis Git

¿Y vuestras/os  
compañeras/os de  
grupo?



**Protected branches**



# Protected branches

[Security](#) [Insights](#) [Settings](#)

## Default branch

The default branch is considered the “base” branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

main

## Branch protection rules

main

Currently applies to 1 branch

Edit

Delete

Add rule



# Protected branches

## Protect matching branches

### ☐ Require a pull request before merging

When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

### ☐ Require status checks to pass before merging

Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

### ☐ Require conversation resolution before merging

When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule. [Learn more.](#)

### ☐ Require signed commits

Commits pushed to matching branches must have verified signatures.

### ☐ Require linear history

Prevent merge commits from being pushed to matching branches.

### ☐ Require deployments to succeed before merging

Choose which environments must be successfully deployed to before branches can be merged into a branch that matches this rule.

### ☐ Lock branch

Branch is read-only. Users cannot push to the branch.

### ☐ Do not allow bypassing the above settings

The above settings will apply to administrators and custom roles with the "bypass branch protections" permission.

## Rules applied to everyone including administrators

### ☐ Allow force pushes

Permit force pushes for all users with push access.

### ☐ Allow deletions

Allow users with push access to delete matching branches.

Create



# Eskerrik asko! Galderarik?

<https://labur.eus/gitmk>