

GIT: TRABAJO REMOTO

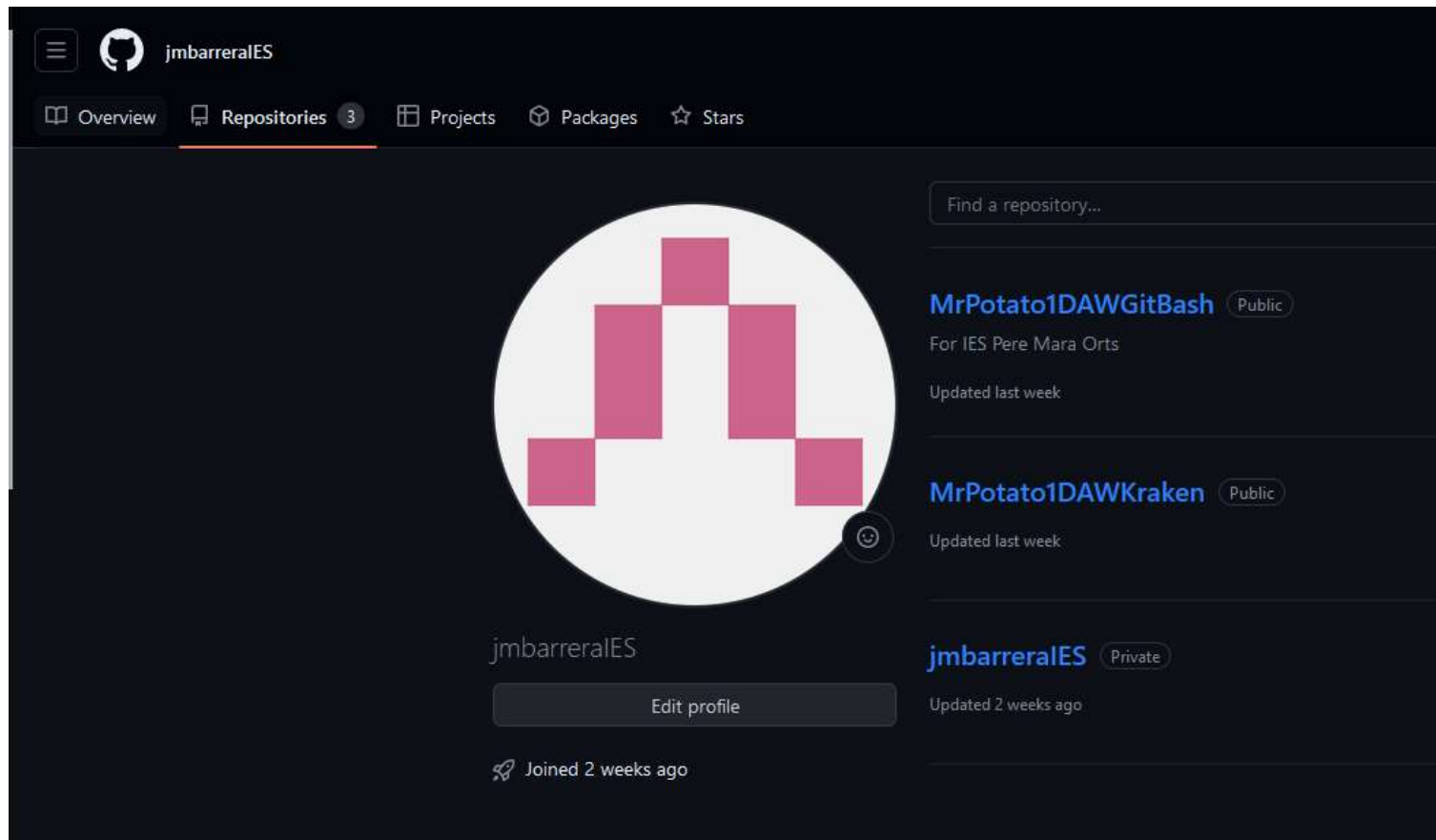
1r DAW

Jose Manuel Barrera Arroyo

GIT: REPOSITORIOS

- Vamos a crear dos repositorios:
 - Uno para SCV de Kraken
 - Otro para SCV de GIT Bash

GIT: REPOSITORIOS



GIT: LINKADO DE REPOSITORIOS

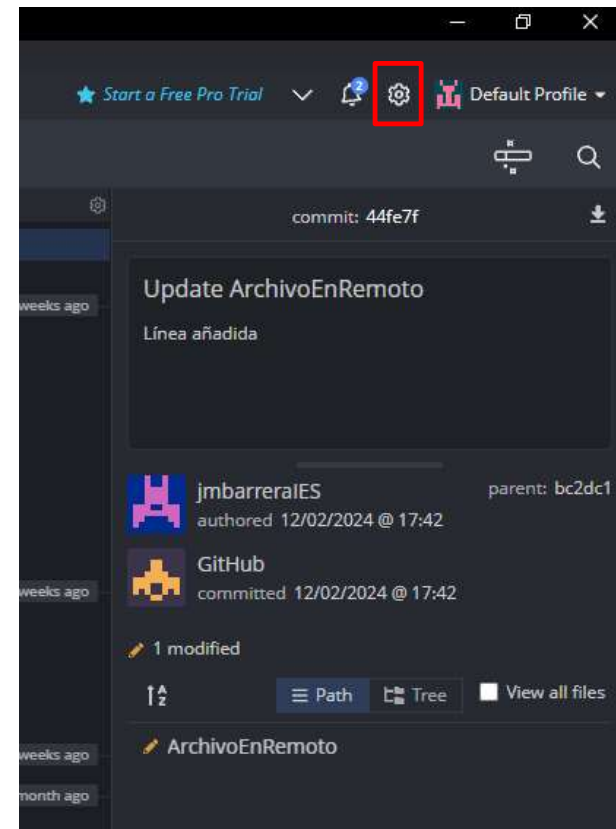
- Una vez se tienen los repositorios, se pueden hacer dos cosas:
 - Subir nuestro trabajo local al remoto (nosotros subimos trabajo al equipo)
 - O bajar del remoto a nuestro local (bajamos trabajo del equipo a nuestro pc)

GIT: LINKADO DE REPOSITORIOS

- Independientemente de lo que queramos hacer, lo primero es linkar el repositorio local con el remoto.
- Para ello, se debe de linkar:
 1. La cuenta de GIT Hub
 2. Añadir el repositorio al SCV

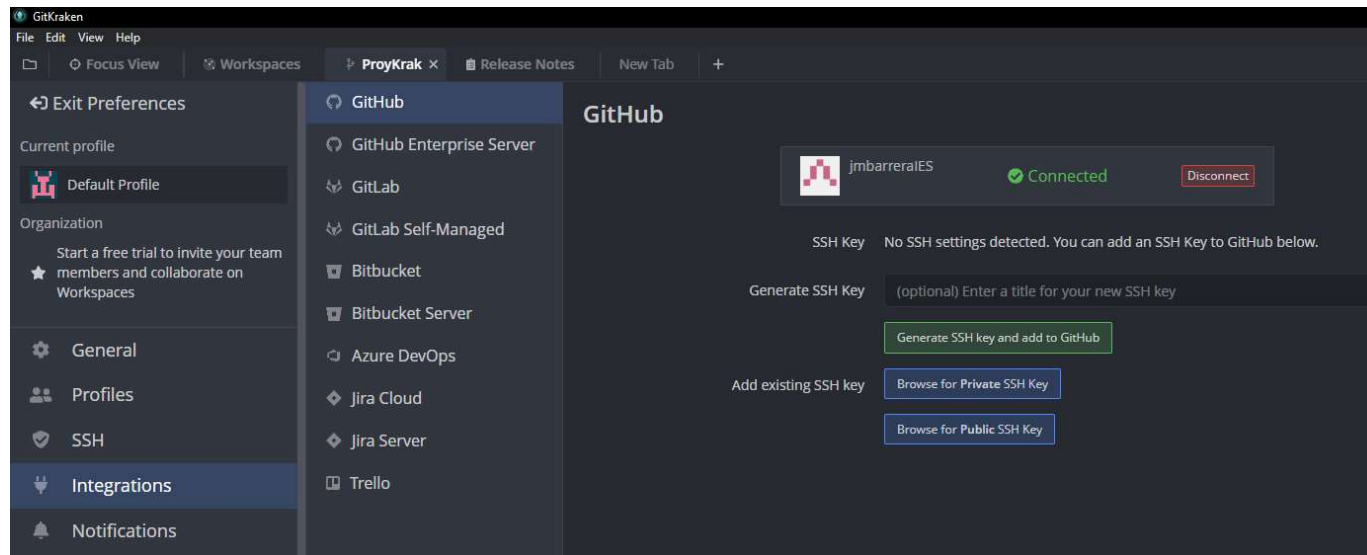
GIT: LINKADO DE REPOSITORIOS

- GIT Kraken: Linkado de cuenta
 - En Preferencias (al lado del perfil) → Integrations → GitHub



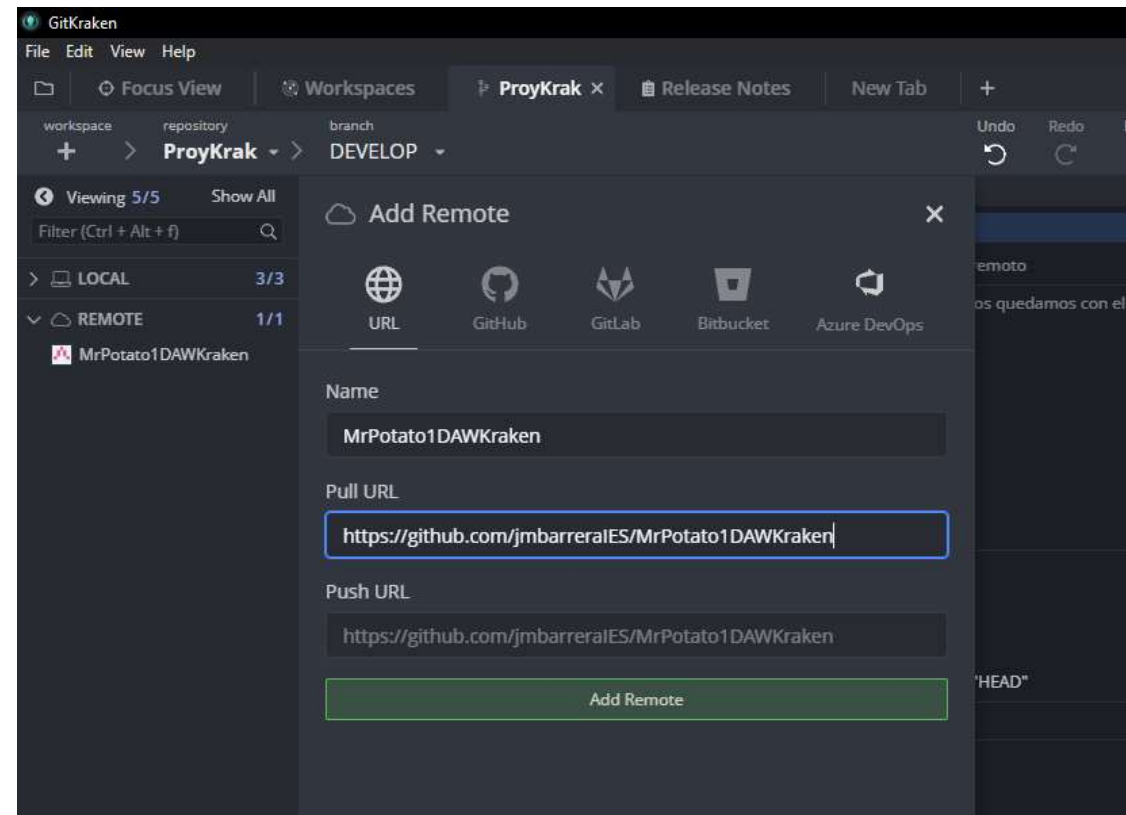
GIT: LINKADO DE REPOSITORIOS

- GIT Kraken: Linkado de cuenta
 - La cuenta debe de haber sido configurada correctamente por SSH



GIT: LINKADO DE REPOSITORIOS

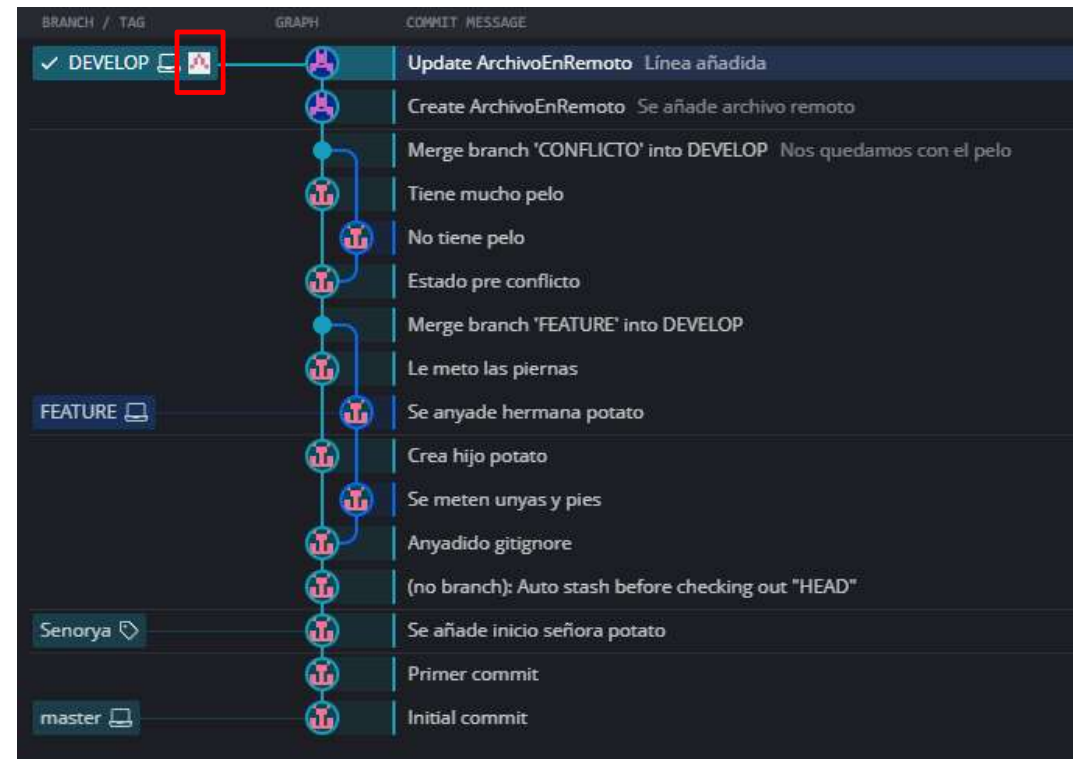
- GIT Kraken: Añadir repositorio
- En la barra lateral Remote → URL → Copy & Paste la dirección del repo remoto



GIT: LINKADO DE REPOSITORIOS

- GIT Kraken: Linkado de cuenta

- A partir de este momento, aparece icono de perfil de GitHub en la rama remota

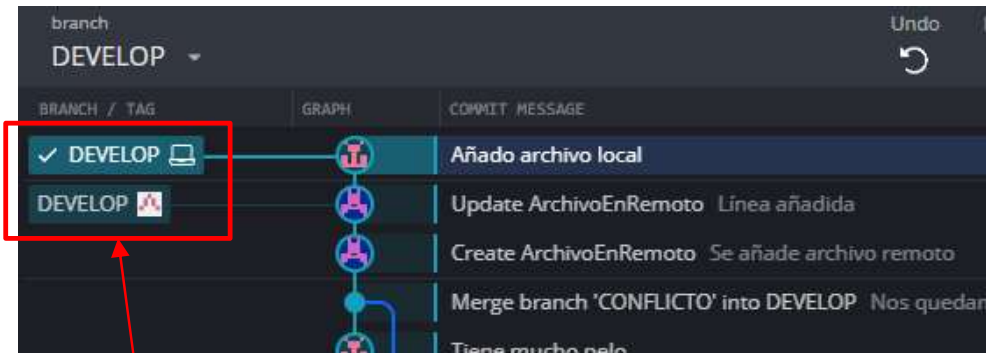


GIT: CREACIÓN DE LA CUENTA

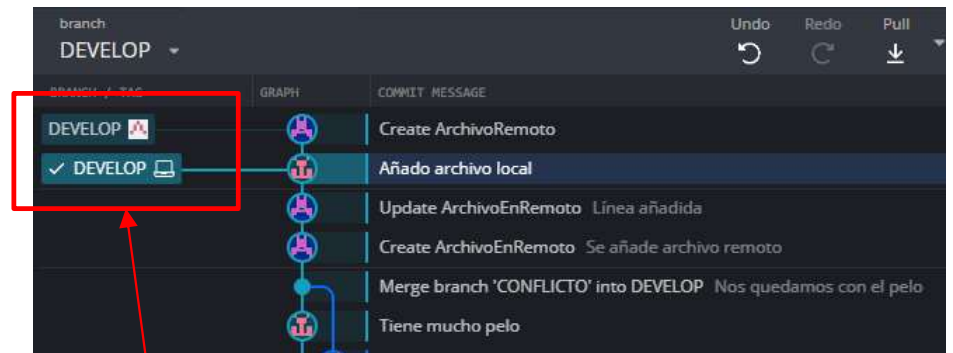
- GIT Kraken: Linkado de cuenta
 - A partir de ahora, podemos añadir archivos al repo local (hacemos trabajo nosotros) para subirlos posteriormente.
 - O añadir archivos al repo remoto (lo hacen los compañeros)
- Cuaquiera de las dos cosas, se nos indicará en el hub de manera visual

GIT: LINKADO DE REPOSITORIOS

- GIT Kraken: Linkado de cuenta



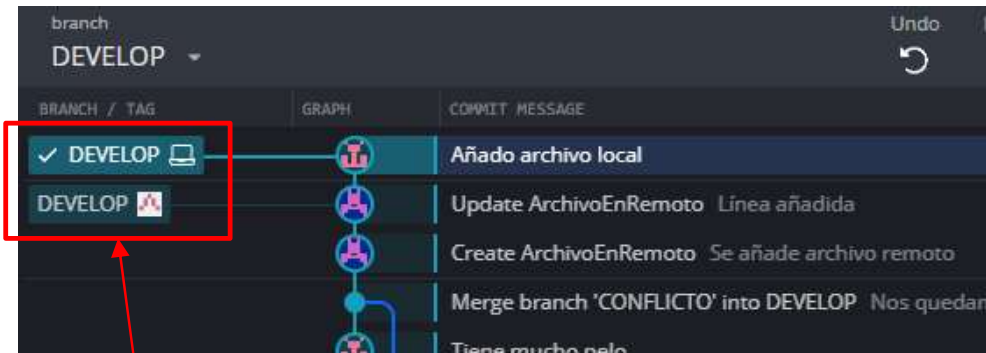
Añadido archivo local. Repo remoto está por detrás



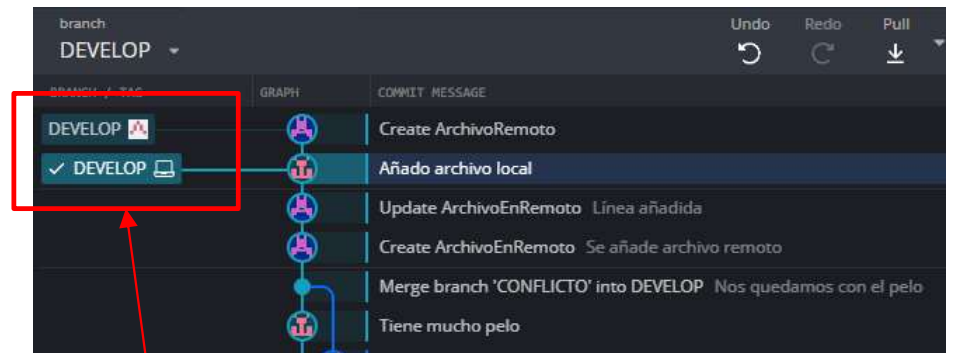
Añadido archivo en remoto. Repo local está por detrás

GIT: LINKADO DE REPOSITORIOS

- GIT Kraken: Sincronización



Click derecho: push



Click derecho: pull

GIT: LINKADO DE REPOSITORIOS

- GIT Bash: Linkado de cuenta y repo.
 - `git remote add origin direcciónwebdelrepositorio`
 - Cuando se ejecuta no hace nada especial

```
AzureAD+JOSEMANUELBARRERAARR@DESKTOP-UGDSMIK MINGW64 ~/Desktop/ProfeBenidorm/A  
ntes EDD/GIT/Proyecto/Proy1 (master)  
$ git remote add origin https://github.com/jmbarreraIES/MrPotato1DAWGitBash  
  
AzureAD+JOSEMANUELBARRERAARR@DESKTOP-UGDSMIK MINGW64 ~/Desktop/ProfeBenidorm/A  
ntes EDD/GIT/Proyecto/Proy1 (master)  
$ |
```

GIT: LINKADO DE REPOSITORIOS

- GIT Bash: Linkado de cuenta y repo.
 - Si hacemos un:
 - `git branch --all`
 - Aparecerán las ramas remotas

```
AzureAD+JOSEMANUEL BARRERAARR@DESKTOP-UGDSMIK MI
ntes EDD/GIT/Proyecto/Proy1 (master)
$ git branch --all
* master
  ninio
  remotes/origin/main
  remotes/origin/master
```

GIT: TRABAJO REMOTO

- GIT Bash: Linkado de cuenta y repo.
 - A partir de este momento, ya podemos subir y bajar archivos, mediante los comandos de sincronización
 - `git push`
 - `git pull`

GIT: TRABAJO REMOTO

- GIT Bash: Subida de archivos
 - Al igual que Kraken, si hacemos un git status nos avisa si estamos por detrás o por delante del repo remoto.
 - Para saber si estamos por detrás del repo remoto:
 - `git fetch` → y luego → `git status`

GIT: TRABAJO REMOTO

- GIT Bash: Subida/descarga de archivos

```
AzureAD+JOSEMANUELBARRERAARR@DESKTOP-UGDSMIK MINGW64 ~/Desktop/Proyecto/Proy1 (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Nos avisa que estamos por delante
(tendríamos que subir el trabajo)

```
AzureAD+JOSEMANUELBARRERAARR@DESKTOP-UGDSMIK MINGW64 ~/Desktop/ProfeBenidorm/Apuntes/Proyecto/Proy1 (master)
$ git fetch
AzureAD+JOSEMANUELBARRERAARR@DESKTOP-UGDSMIK MINGW64 ~/Desktop/ProfeBenidorm/Apuntes/Proyecto/Proy1 (master)
$ git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
(use "git pull" to update your local branch)
```

Nos avisa que estamos por detrás
(tendríamos que descargar el trabajo)

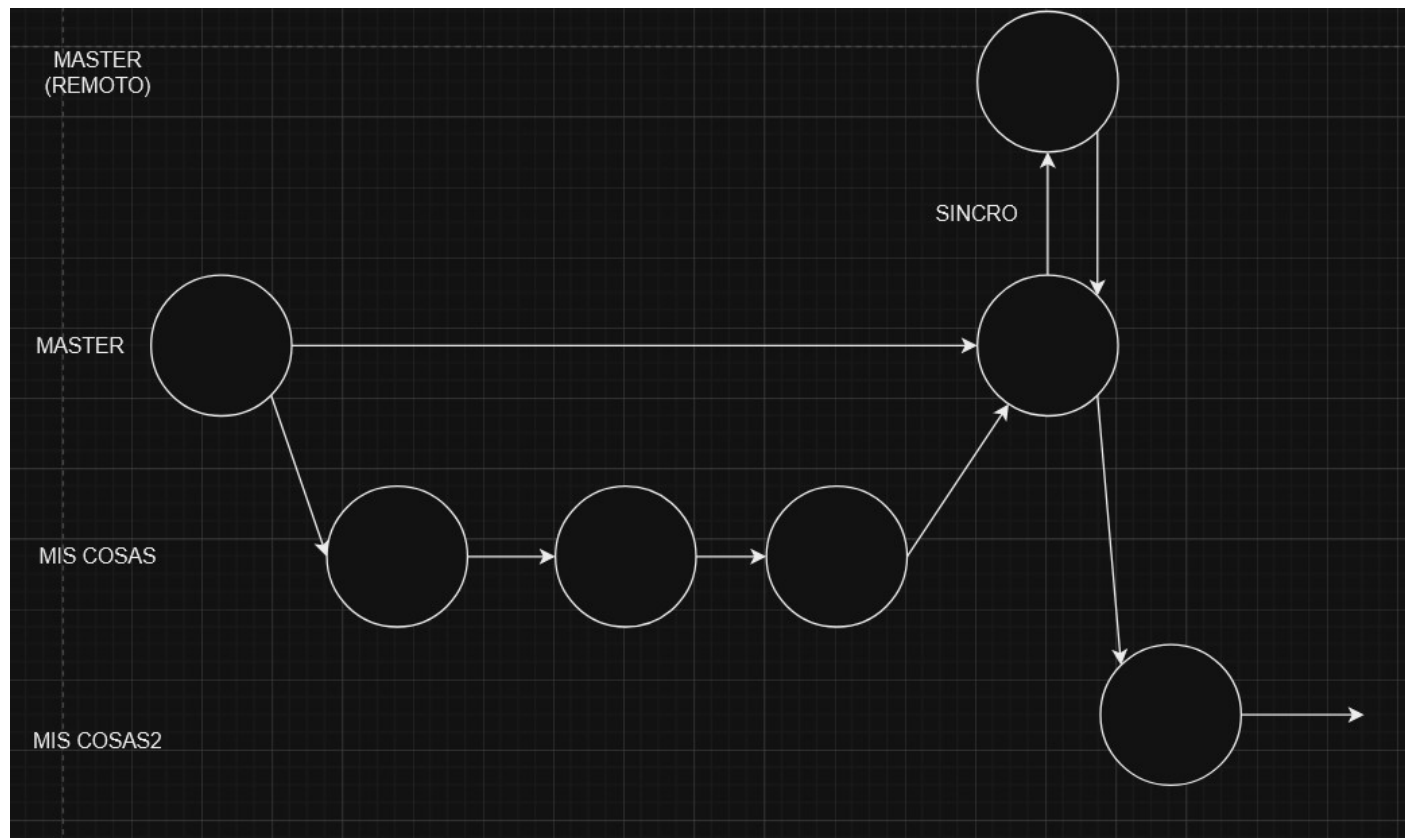
GIT: TRABAJO REMOTO

- Git nunca nos dejará subir un trabajo a una rama, si no estamos actualizados.
- Si estoy trabajando en una rama, y tengo archivos nuevos, y en esa misma rama remota hay archivos que no tengo, me obligará a bajarlos primero antes de subirlos.

GIT: TRABAJO REMOTO

- Aquí es muy importante la filosofía de ramas.
- Si mi jefe quiere únicamente una rama de sincronización, pero a mi me gusta hacer muchos commits, usaré una rama local no remota, y luego mergearé en la rama remota para subir el trabajo.

GIT: TRABAJO REMOTO

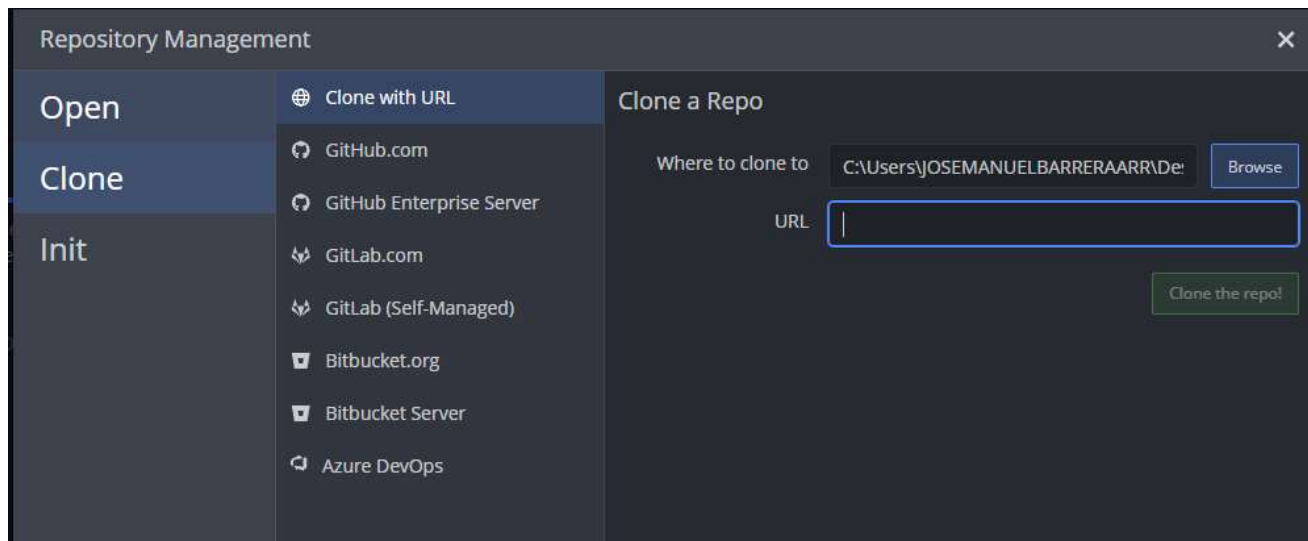


GIT: ENTRANDO A UN PROYECTO

- Cuando entro a un proyecto, lo normal es acceder al repo con el historial de lo que han hecho hasta ahora.
- Para eso se usa el comando
 - `git clone`

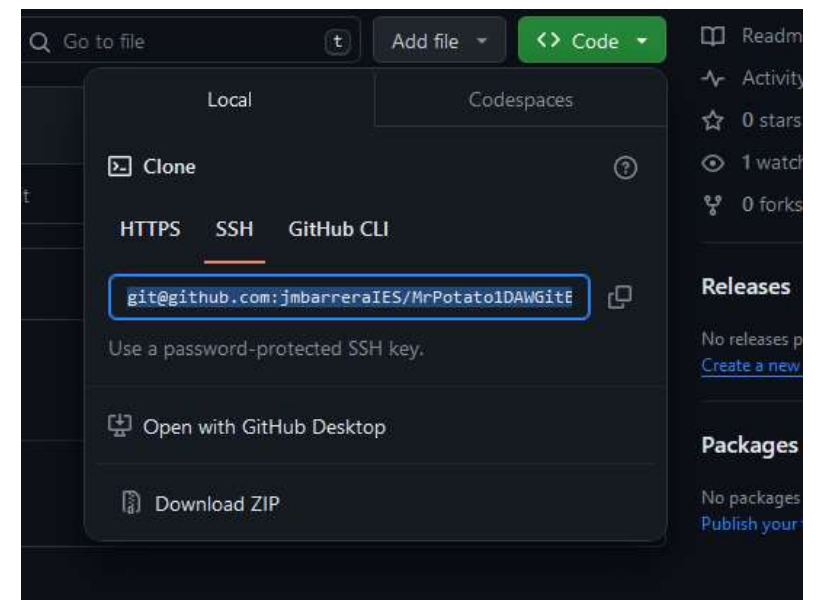
GIT: ENTRANDO A UN PROYECTO

- Git Kraken
 - Tengo que crear la carpeta con el proyecto



GIT: ENTRANDO A UN PROYECTO

- Git Bash
 - Tengo que crear la carpeta con el proyecto
 - `git clone direcciondecopiado`
 - La dirección la podemos encontrar en el repo remoto (o nos la puede dar el equipo)



GIT: ENTRANDO A UN PROYECTO

- Git Bash

```
AzureAD+JOSEMANUEL BARRERAARR@DESKTOP-UGDSMIK MINGW64 ~/Desktop/ProfeBe
ntes EDD/GIT/Proyecto/PB1
$ git clone https://github.com/jmbarreraIES/MrPotato1DAWGitBash.git
Cloning into 'MrPotato1DAWGitBash'...
remote: Enumerating objects: 53, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 53 (delta 13), reused 40 (delta 10), pack-reused 0
Receiving objects: 100% (53/53), 7.31 KiB | 3.66 MiB/s, done.
Resolving deltas: 100% (13/13), done.
```


GIT: TRABAJO REMOTO

- GIT Bash: Subida de archivos
 - `git push -u nombredelaramaprincipal`
 - El nombre de la rama principal es la local que estemos usando

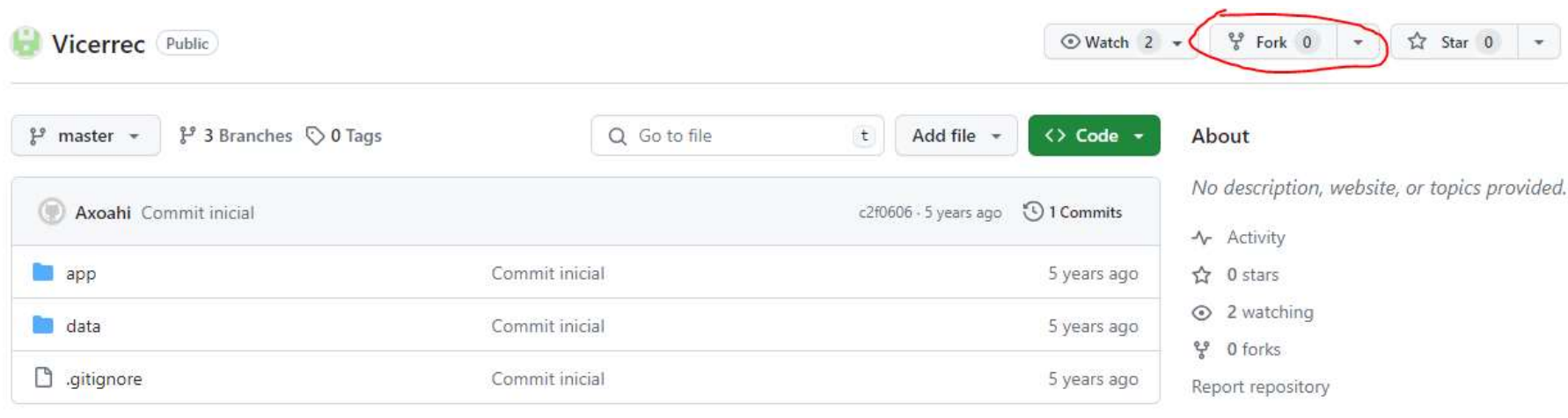
```
AzureAD+JOSEMANUELBARRERAARR@DESKTOP-UGDSMIK MINGW64 ~/Desktop/ProfeBenidorm/Apu
ntes EDD/GIT/Proyecto/Proy1 (master)
$ git push -u origin master
Everything up-to-date
branch 'master' set up to track 'origin/master'.

AzureAD+JOSEMANUELBARRERAARR@DESKTOP-UGDSMIK MINGW64 ~/Desktop/ProfeBenidorm/Apu
ntes EDD/GIT/Proyecto/Proy1 (master)
$ :
```

GITHUB: FORK

- El fork es la clonación de un repo en otro usuario.
- Sirve para controlar el acceso (y el contenido) al código depurado.
- Sigue la filosofía de “primero experimenta con una copia del proyecto en tu GitHub, y luego ya veremos si lo integro en el final”

GITHUB: FORK



The screenshot shows the GitHub interface for a repository named 'Vicerrec' (Public). At the top, there are buttons for 'Watch' (2), 'Fork' (0), and 'Star' (0). The 'Fork' button is circled in red. Below the repository name, there are tabs for 'master' (3 Branches, 0 Tags), a search bar 'Go to file', and buttons for 'Add file' and '<> Code'. The main content area shows a commit history table with one commit by 'Axoahi' titled 'Commit inicial' (c2f0606, 5 years ago, 1 Commit). The table lists files: 'app', 'data', and '.gitignore', all committed initially 5 years ago. On the right, the 'About' section states 'No description, website, or topics provided.' and lists repository statistics: Activity, 0 stars, 2 watching, 0 forks, and a 'Report repository' link.

Vicerrec Public

Watch 2 Fork 0 Star 0

master 3 Branches 0 Tags

Go to file Add file <> Code

Axoahi	Commit inicial	c2f0606 · 5 years ago	1 Commits
app	Commit inicial	5 years ago	
data	Commit inicial	5 years ago	
.gitignore	Commit inicial	5 years ago	

About

No description, website, or topics provided.

Activity

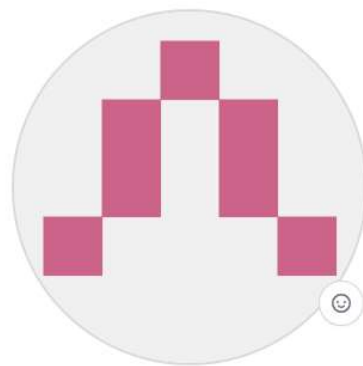
0 stars

2 watching

0 forks

Report repository

GITHUB: FORK



jmbarreraLES

Edit profile

Joined 3 weeks ago

Find a repository...

Type ▾

Language ▾

Sort ▾

New

MrPotato1DAWGitBash Public

For IES Pere Mara Orts

Updated last week

☆ Star ▾

MrPotato1DAWKraken Public

Updated last week

☆ Star ▾

jmbarreraLES Private

Updated 3 weeks ago

☆ Star ▾

Vicerrec Public

Forked from [Axoahi/Vicerrec](#)

Python Updated on Apr 21, 2020

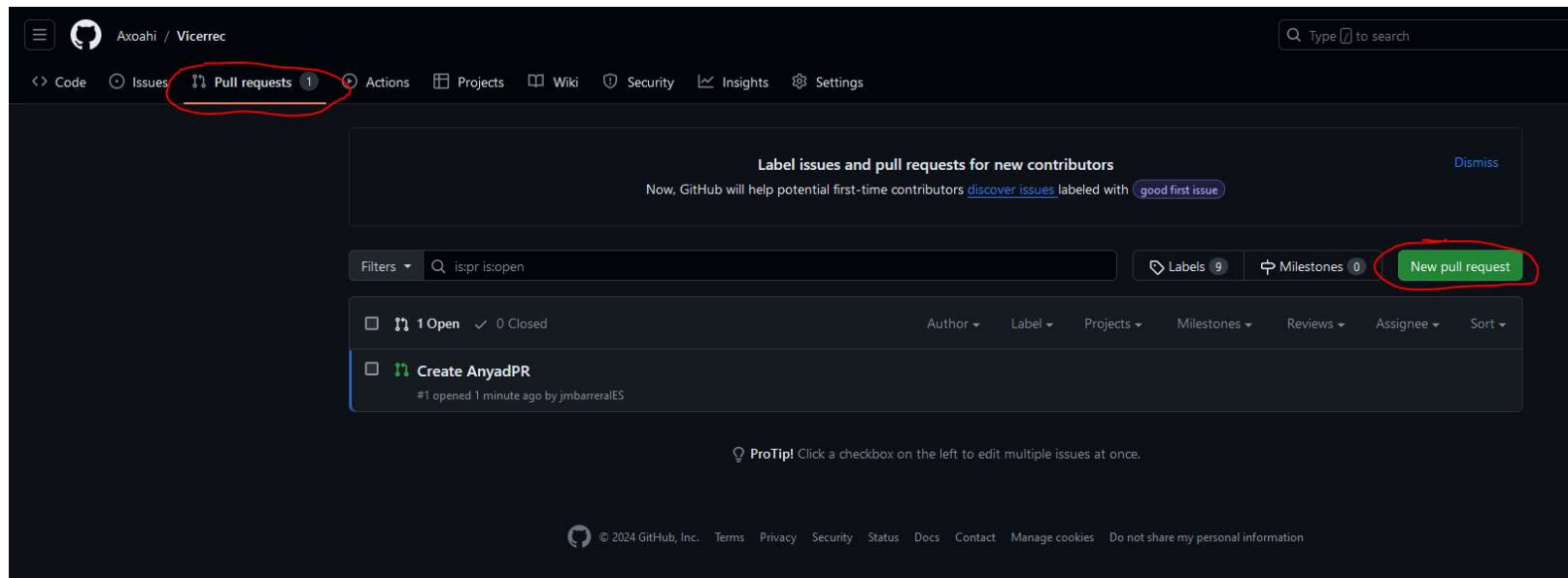
☆ Star ▾

GITHUB: PULL REQUEST (PR)

- Es el siguiente paso del fork: he modificado algo, y quiero que el dueño del proyecto lo meta.
- Es un merge entre el mismo proyecto de dos usuarios (y dos repos remotos) distintos.

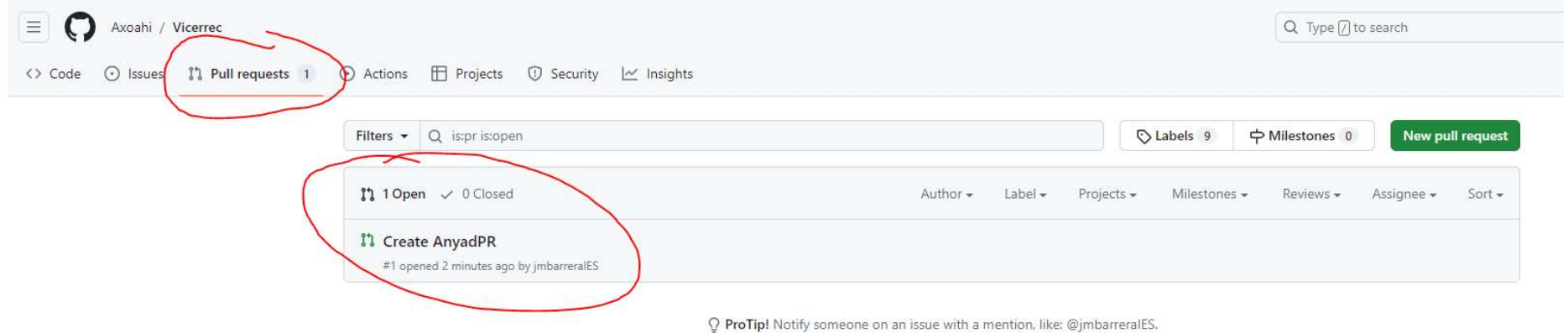
GITHUB: PULL REQUEST (PR)

- El que ha hecho la modificación en su repo, solicita al dueño que lo “mergee” mediante un PR.



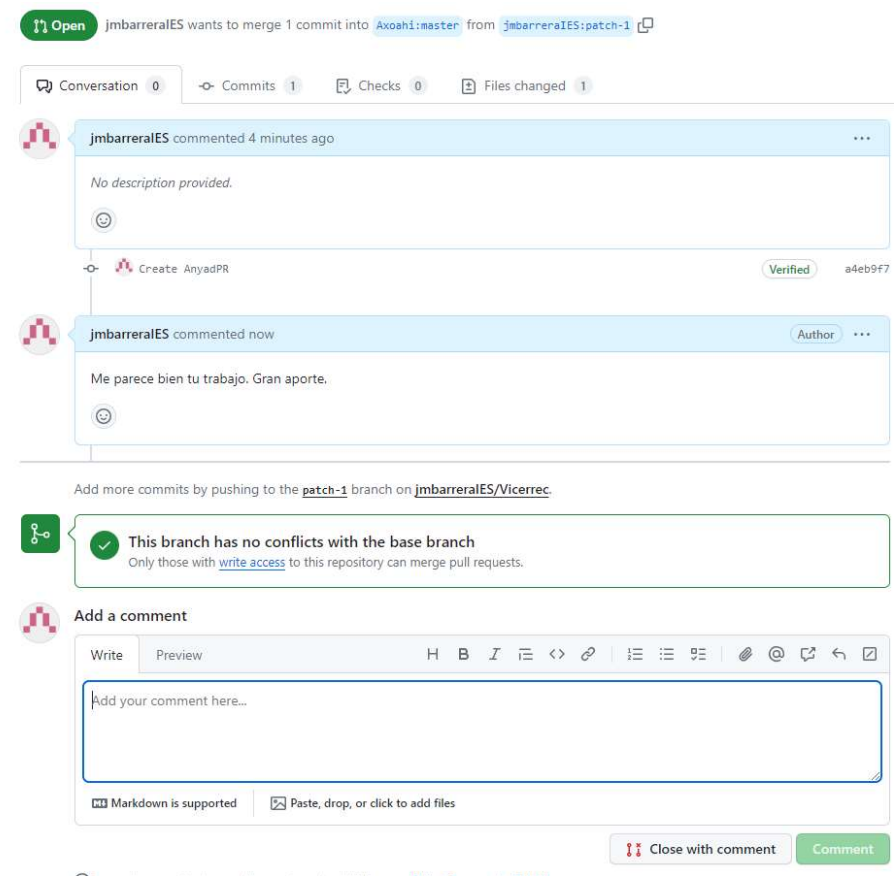
GITHUB: PULL REQUEST (PR)

- El dueño, recibe la solicitud de PR

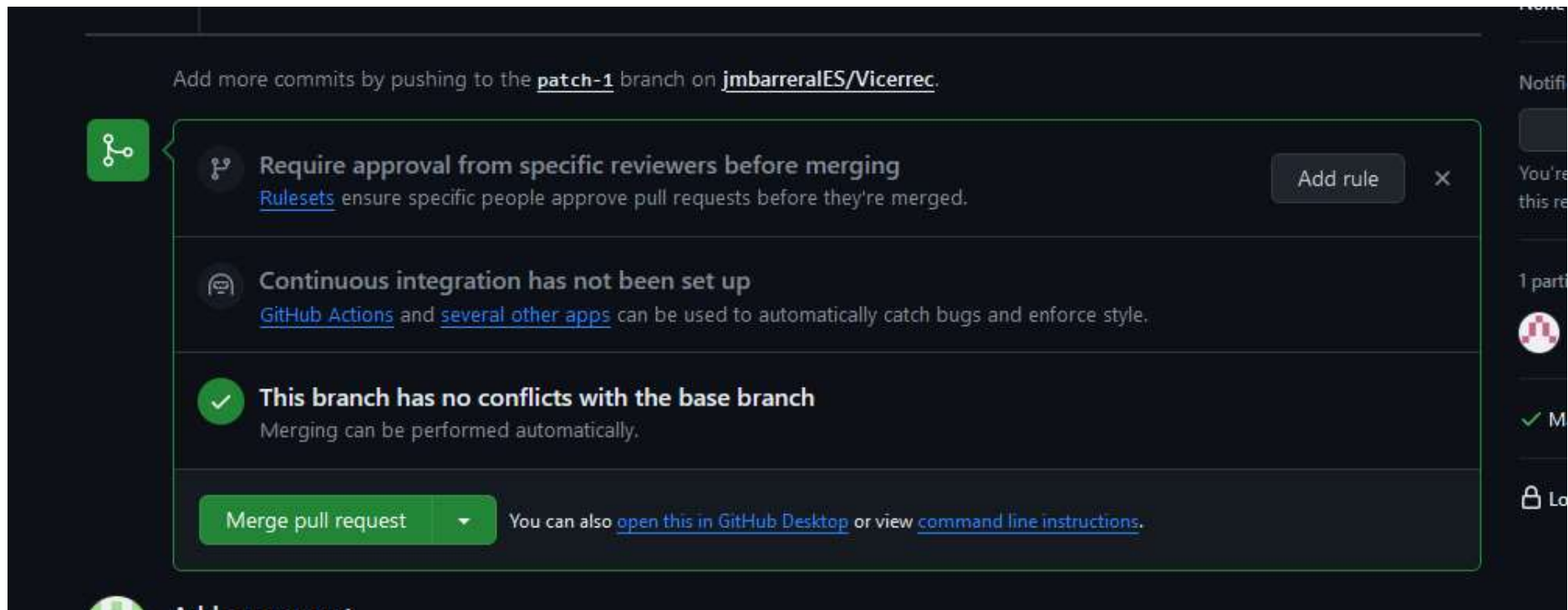


GITHUB: PULL REQUEST (PR)

- La visualiza, y decide si la acepta o no (Avisa si hay conflictos)



GITHUB: PULL REQUEST (PR)



GITHUB: PULL REQUEST (PR)

- Si se acepta el PR, el repo queda actualizado.
- El usuario que ha “forkeado” también tiene un botón de actualizar su fork, para que coincida con la versión del repo.

GITHUB: PULL REQUEST (PR)

The screenshot shows a GitHub repository named 'Vicerrec' (Public) forked from 'Axoahi/Vicerrec'. The repository has 2 branches and 0 tags. The current branch is 'master'. A pull request merge by 'Axoahi' is shown, merging 'Axoahi#2' from 'Axoahi/revert-1-patch-1' 45 minutes ago. The file history table shows the following files and their commit details:

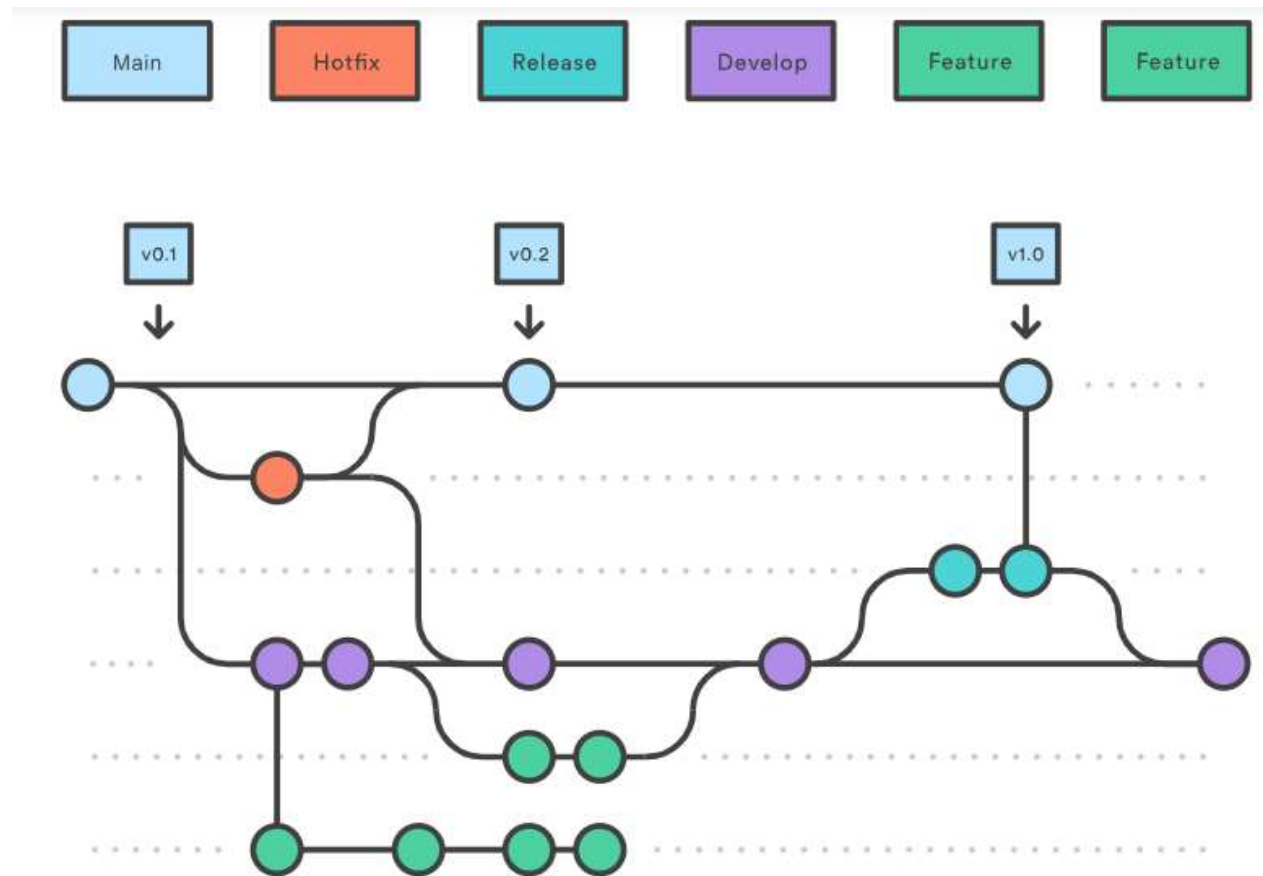
File	Commit	Time
app	Commit inicial	5 years ago
data	Commit inicial	5 years ago
.gitignore	Commit inicial	5 years ago
README.md	Se anyado el readme	1 hour ago

Below the file history, the 'README' file is shown with the text: 'Oye te anyado el Readme que esto no puede ser'.

GITHUB: PR. CONFLICTOS

- Cuando se es dueño de un repo, y se tiene PR, es muy probable que aparezcan conflictos.
- Estos conflictos se resuelven de la misma manera que en local, pero se suele usar la propia interfaz de GitHub para resolverlo.

GITHUB: GIT FLOW



GITHUB: GIT FLOW

- MAIN: Solo releases (con versión)
- DEVELOP: Rama de trabajo, con versiones testeadas.
- FEATURES: Ramas de trabajo.
- RELEASE: Preparación de release. Bloque de funcionalidades para main.
- HOTFIX: Arreglos urgentes (suelen venir de main)

GITHUB: GIT FLOW

- Mas info detallada:
 - <https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow>