

C) Trabajo colaborativo con GitHub

(Por alguna razón me da error al instalar git en mi máquina virtual, por lo que esta práctica está hecha desde la terminal de windows)

1. Ahora imagina que llega una nueva programadora a tu equipo, y tiene que descargarse todo el proyecto del repositorio base(*) de GitHub en su ordenador, ¿qué comando debe usar? Haz una simulación, como si fueras esa nueva programadora, y descargalo a tu local pero en otra ubicación (por ejemplo, si tu repo local lo tienes en /Documentos, pues haz la prueba de clonación en /Escritorio o viceversa). Incluye captura.

```
C:\Users\Marcos>git clone https://github.com/Marek8007/HelloGit pruebasGit
Cloning into 'pruebasGit'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 17 (delta 2), reused 14 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (2/2), done.
```

(*) Los términos "cabeza" y "base" se utilizan normalmente en Git. La cabeza o HEAD es la rama en la que te encuentras; es decir, la rama con tus cambios. La base es la rama en la que se basan estos cambios.

Ten en cuenta que el comando **git clone** crea una copia local de un repositorio remoto. Por defecto, se clona en el directorio desde el que se hace git clone y se crea una carpeta local con el nombre del repositorio remoto. Sin embargo, puedes especificar cualquier carpeta local como destino.

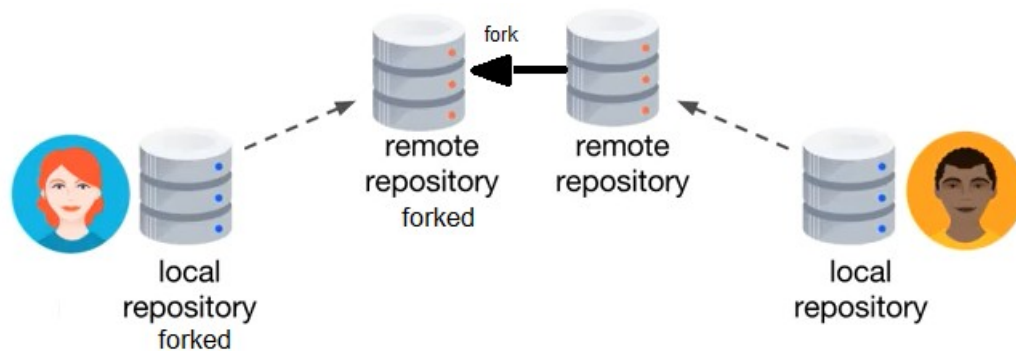
`git clone <repo>` o `git clone <repo> <directory>`

Ejemplo: `git clone https://github.com/tuusuario/tu repositorio`

2. Esta nueva programadora ya se ha clonado el repositorio en su local, después hace cambios de código o añade ficheros y finalmente intenta hacer un git push de los cambios comiteados para subirlos al remoto. GitHub no le dejará porque el repositorio remoto pertenece a otro usuario (es tuyo) y esta nueva programadora no tiene permisos.

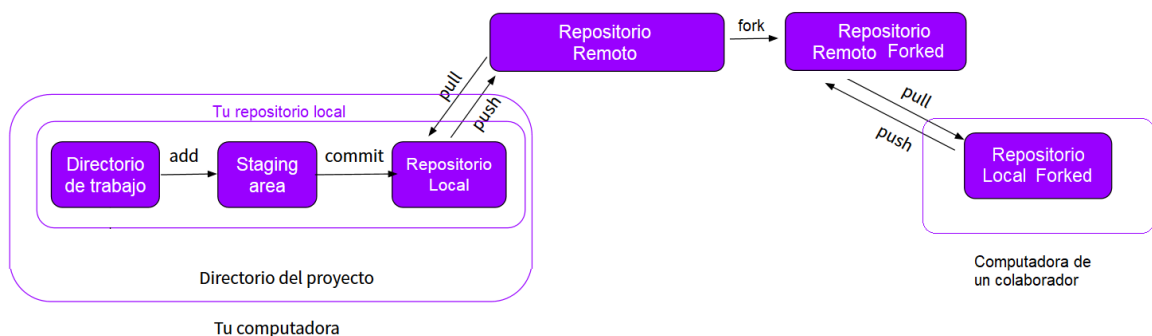
Vamos a ver como funciona el flujo colaborativo de GitHub, esta nueva programadora te debería indicar a tí, que eres el propietario del repositorio remoto, que quiere hacer cambios, ¿cómo te lo indica?. Ponte en el papel de esta nueva programadora y haz lo que ella debería hacer para indicártelo, que es lo siguiente:

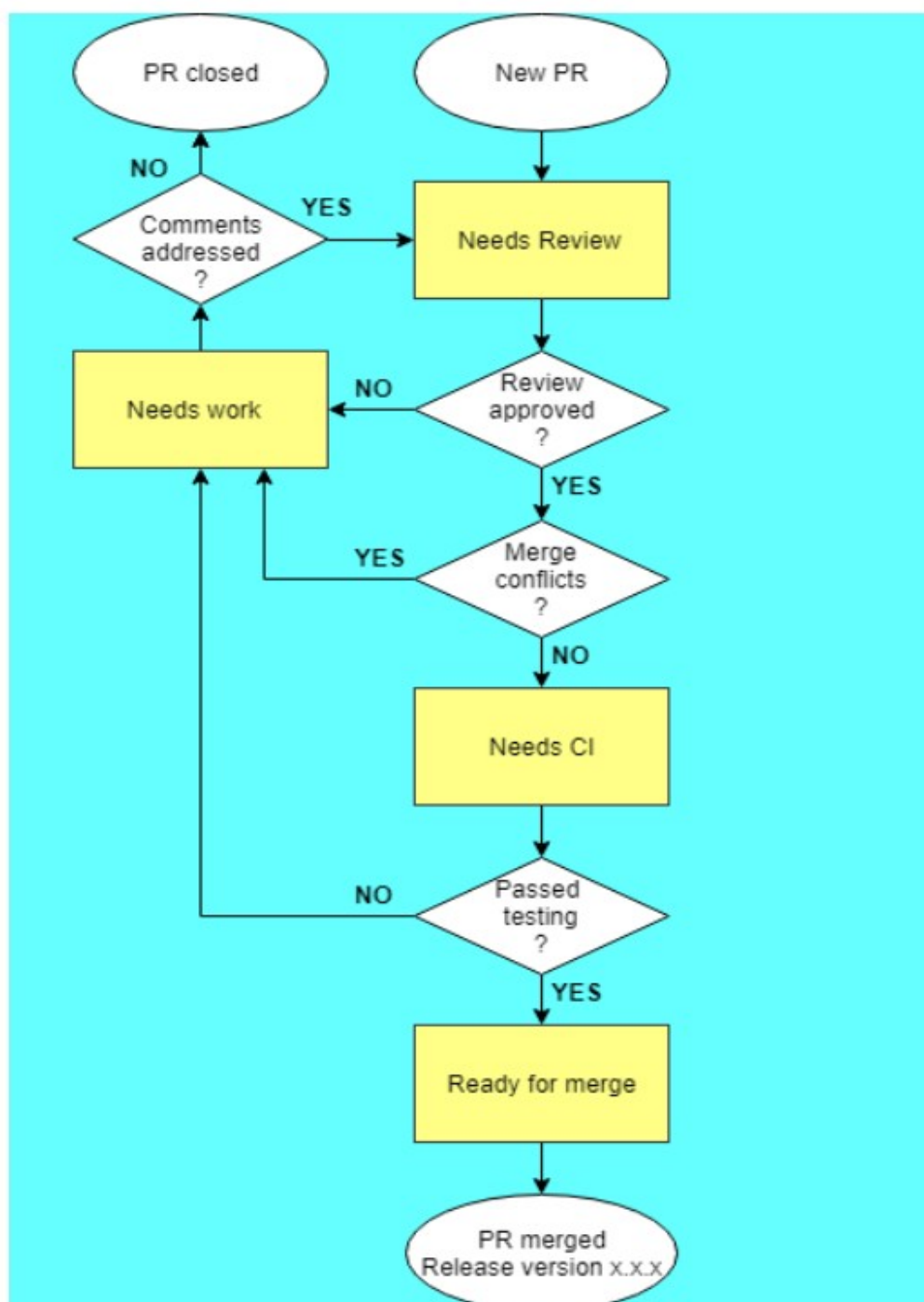
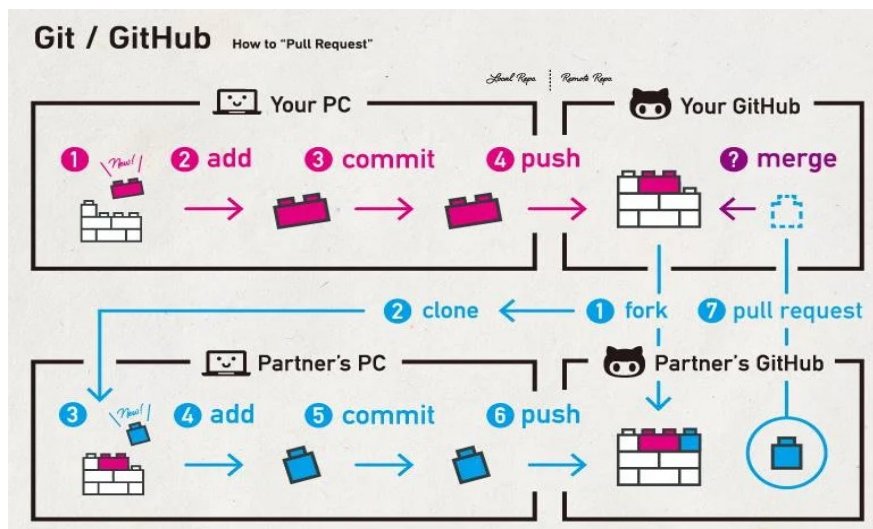
1º) El primer paso que tendría que haber hecho es un **fork** en GitHub del repositorio base, esto es “como crear un nuevo repositorio remoto clonado pero con otro nombre” (por ejemplo, con el nombre: tuusuario/HelloGit-suusuario), después tiene que descargarse en local ese nuevo “repositorio forked”, entonces ahí sí puede hacer cambios, probarlos, comitearlos y subirlos a ese “repositorio forked”.



2º) El segundo paso que tiene que hacer es solicitar al propietario del repositorio original (en este caso tu), que desea enviar cambios. Previamente, debe tenerlo todo bien sincronizado con el repositorio original, usando la opción “**Sync fork**” de GitHub. Hecho esto, desde la opción “**Contribute**” podrá abrir una petición con la opción de GitHub “**Open pull request**” (ya que tiene un commit hecho después del último commit del repositorio original) **con un comentario descriptivo** de lo que ha hecho, y terminar creando la petición con la opción “**Create pull request**”. El propietario del repositorio original (en este caso tu) ve la petición, lee el comentario descriptivo y revisa los cambios del código enviados. El propietario podrá, o bien dejar solo un comentario de repuesta, o bien aprobar los cambios, o bien pedirle algunos retoques. Entonces esta nueva programadora verá la respuesta del propietario.

- Cuando el propietario aprueba sus cambios, él mismo hará el “Merge pull request” en su repositorio original. Si hubieran conflictos, él tendría que solucionarlos y después hacer “**Mark as resolved**”, “**Commit merge**” y “**Merge pull request**”, por último, ya resueltos conflictos, haría “**Confirm request**” con lo que aceptaría los cambios y pasarían al repositorio original, con esto la pull request quedaría cerrada y el repositorio original con los cambios de la nueva programadora incluidos.
- Cuando el propietario no aprueba los cambios, puede hacer “**Comment**” para responder que tiene que corregir algo, o podría hacer directamente “**Close pull request**” no aceptando los cambios.

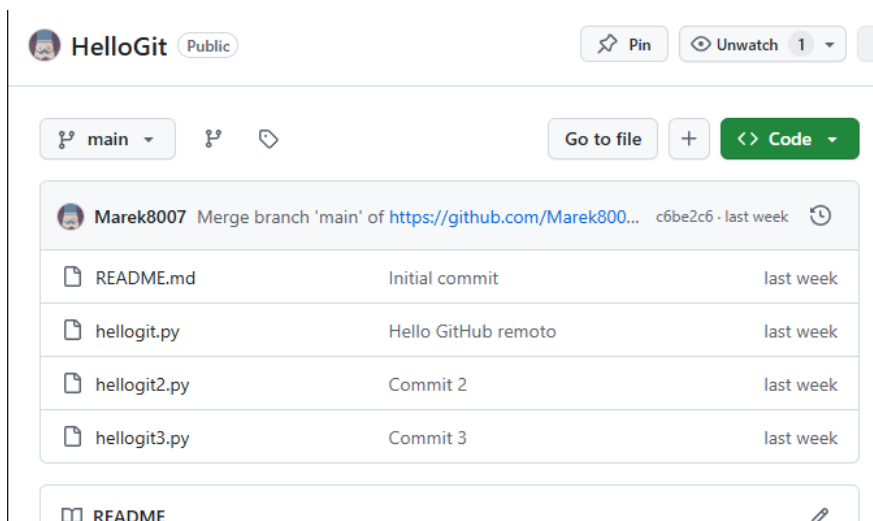




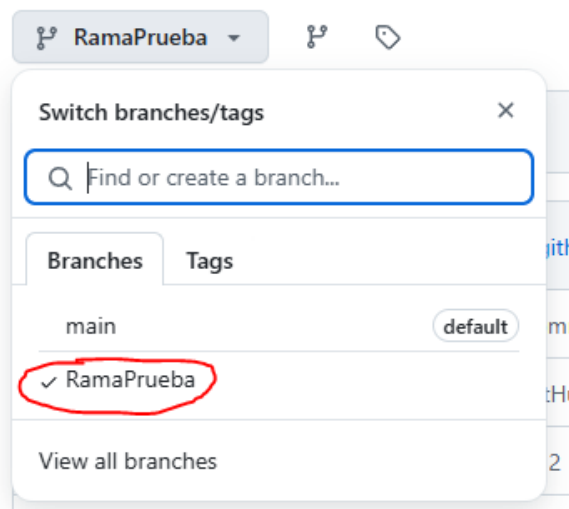
Pull request workflow

3. Realiza este pequeño ejercicio para entender el flujo de trabajo de solicitud de cambios de GitHub. El objetivo de este ejercicio es mostrar el flujo de solicitar a alguien que revise los cambios de mi rama para que los fusione en su rama: <https://docs.github.com/es/get-started/start-your-journey/hello-world>

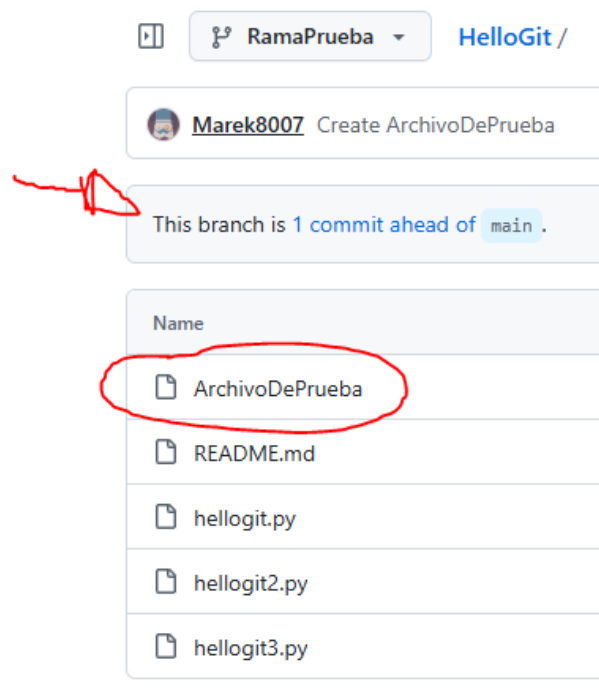
Paso 1 : crear repositorio en GitHub



Paso 2: crear una rama nueva




Paso 3: modificar algo en la rama nueva y commitear




Paso 4: Apertura de una solicitud de cambios


Create ArchivoDePrueba #1

 Open Marek8007 wants to merge 1 commit into `main` from `RamaPrueba` 

 Conversation 0

 Commits 1

 Checks 0

 Files changed 1



Marek8007 commented now

Owner

...

No description provided.



Create ArchivoDePrueba

Verified

b9c0268



No conflicts with base branch

Merging can be performed automatically.


Merge pull request


You can also merge this with the command line.

[View command line instructions.](#)


Paso 5: Fusionar la pull request


Create ArchivoDePrueba #1

 Merged

Marek8007 merged 1 commit into `main` from `RamaPrueba`  now

 Conversation 0

 Commits 1

 Checks 0

 Files changed 1



Marek8007 commented 1 minute ago

Owner

...

No description provided.



Create ArchivoDePrueba

Verified

b9c0268



Marek8007 merged commit aae5f5e into `main` now

Revert

Pull request successfully merged and closed

You're all set — the `RamaPrueba` branch can be safely deleted.

Delete branch

4.