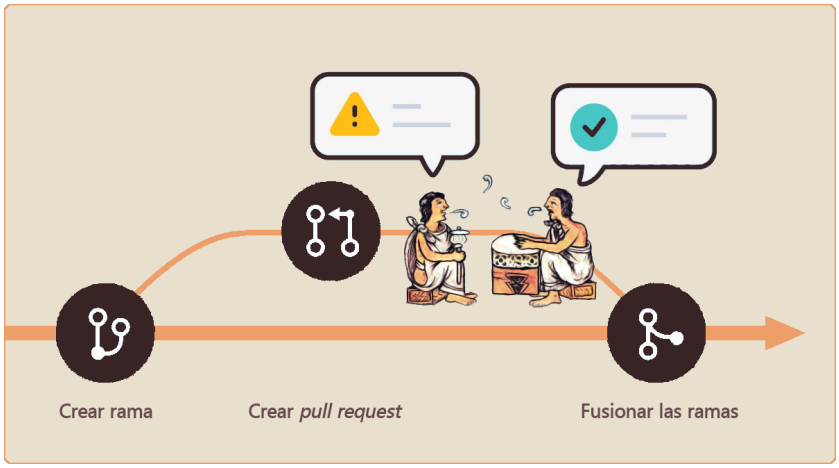
# Colaborar con ayuda de git

Miguel Equihua

Xalapa, Ver., 23 de mayo, 2025




La colaboración con otros miembros del equipo puede hacerse mediante lo que se llamma un *Pull request* en *Git*. Podríamos llamarlo una *Solicitud de Fusión*, aunque lo común es usar la frase en inglés. Es la base del uso de un repositorio *Git* como plataforma de colaboración, además de su papel en el control de versiones. Es el mecanismo seguro de solicitar la incorporación de cambios al proyecto. La ideas es tan simple como el nombre indica: se solicita al responsable del proyecto que incorpore una nueva rama (desde un repositorio que tiene cambios propuestos), en otra (del mismo o de otro repositorio).

Con las peticiones de fusión puedes:

- Comparar los cambios entre dos ramas
- Revisar y discutir las modificaciones propuestas
- Construir, probar y desplegar el efecto de la modificación en el proyecto.
- Evitar que la adición solicitada se aplique antes de ser adecuadamente revisada.

En `usethis` el subconjunto de funciones con el prefijo `pr_` se refieren a las operaciones relacionadas con *pull requests* en Github. Encontraras más información en Happy Git.

Una solicitud de fusión involucra a dos participantes, los llamaremos: contribuyente y revisor. Como las funciones `pr_` usan la infraestructura interactiva de Github (API) Necesitaras un *token personal de acceso*, recuerda que lo puedes hacer con este código:

```
usethis::create_github_token()
```

Al ejecutarlo te llevará a la página adecuada para la tarea en Github con la información recomendada ya registrada, sólo falta describir el uso que darás al token y la caducidad que prefieres. Copia el token y registralo en forma segura en *RStudio*

```
gitcreds::gitcreds_set()
```

Si ya ttenía un *token* registrado, el comando anterior te lo hará saber y te preguntará sobre si quieres verlo. Como probablemente estás aquí ¡porque tu *token* venció!, lo apropiado será remplazar lo que está registrado en este momento. Elige la opción apropiada para hacerlo.

## Copiar y clonar localmente

Para empezar a colaborar, lo primero es que cada participante tenga su propia copia (*fork*) del proyecto. Esto se puede hacer con la instrucción siguiente:

```
usethis::create_from_github("equihuam/2025-ciencia-reproducible", fork = TRUE)
```

Este comando hará lo siguienteen tu cuenta de *Github*:

- Una copia del repositorio *2025-ciencia-reproducible* que tengo en mi cuenta personal.
- Clonará la copia del repositorio en tu equipo.
- Definirá el *orígen remoto* del repositorio vinculado a tu copia en Github.
- También definirá que t copia está vinculada a mi repositorio del mismo nombre.
- Configura el sistema para viggilar lo que ocurra en la rama principal de mi repositorio.
- Abrir una nueva ventana de RStudio con tu copia clonada.
- Algunas opciones adicionales en el comando de `usethis` que podrías querer usar:
  - controla si se debe copiar o no `fork = TRUE` o `fork = FALSE`.
  - Usa `destdir` para especifica en donde quieres poner tu clon del repositorio.
  - Podrías incluso usar la opción `usethis.destdir` para que los nuevos proyectos siempre vayan a un directorio específico de tu preferencia.

### Crea una rama y propon cambios

Para crear tu propuesta de cambio puedes usar la función `pr_init()` que crea una rama nueva en tu repositorio en preparación para hacer la solicitud de incorporación de cambios. Es buena práctica hacerlo así, en lugar de usar la rama principal (comunmente *main* o *master*). En este caso llamaremos a la rama nueva idea-1

```
usethis::pr_init(branch = "idea-1")
```

La ejecución de este comando crea la rama y la hace el espacio activo de trabajo. Ahora puedes hacer todo el trabajo que gustes, hacer tus *commits* en esta rama y demás.

Cuando estés lista para enviar tu contribución con las adecuaciones y cambios que deseas proponer usa el siguiente comando.

```
usethis::pr_push()
```

La ejecución del comando hará que se presente un ventana de tu navegador de Internet en la *URL* (Uniform Resource Locator) de interés. En la ventana podrás ver una comparación de lo que hay en el repositorio original (izquierda) y los cambios que propones (derecha). También verás un gran botón verde que dice **Create Pull Request**. Si estás lista, apriétalo para enviar la solicitud a la consideración del responsable. Al apretar el botón tendrás opción de indicar si es un borrador o una propuesta terminada. En el primer caso, no olvides marcarla como lista para ser revisada cuando hayas terminado de trabajarla y la consideres completa, momento en el que habrás de lanzar un nuevo `pr_push()`

GitHub enviará un mensaje al responsable, quien revisará tu solicitud. Podrás ver lo que está pasando con la instrucción:

```
usethis::pr_view()
```

Te llevará a la página adecuada en tu navegador de Internet en donde se está dando segimientto a tu petición.

En el proceso de revisión el responsable decidirá en algún momento llevar tu propuesta a su máquina. Lo podrá hacer con la instrucción

```
usethis::pr_fetch()
```

Al ejecutar este comando se crea una nueva rama en el equipo del responsable con un identificador que incluye el dato del usuario que remite la contribución. Una vez que está satisfecho con el contenido, que pdo haber ajustado según su perspectiva, deberá emitir un nuevo push:

```
usethis::pr_push()
```

que actualizará los cambios con sus sugerencias. Cuando todo se valora está en orden, el responsable emitirá la orden de incorporar los cambios y dar por terminado el proceso. Lo puede hacer con las siguiente instrucción.

```
usethis::pr_finish()
```

Para poner tu copia del repositorio al día deberás emitir una penúltima instrucción

```
usethis::pr_pull()
```

que incorporará la versión final de los cambios en tu copia del repositorio. Finalmente, podrás cerrar el proceso desde tu lado con:

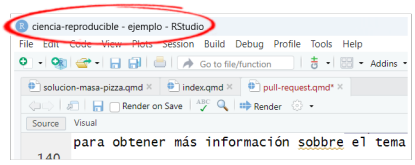
```
usethis::pr_finish()
```

lo que dará por terminado el proceso.

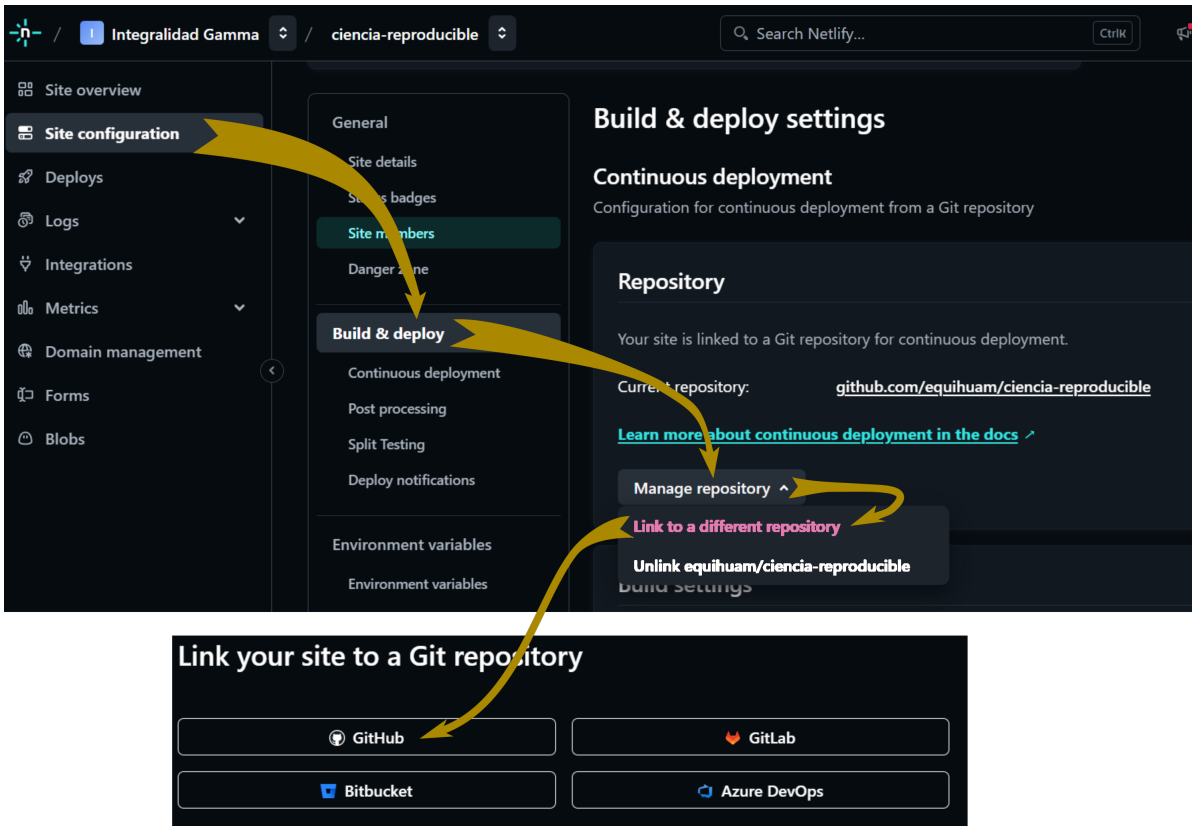
Puedes encontrar este texto útil para obtener más información sobbre el tema.

## Algunos problemas encontrados

1. Un problema que encontramos el año pasado fue la de operar *RStudio* fuera del proyecto de interés en forma inadvertida. Te sugiero siempre verificar que estés trabajando con el proyecto que deseas. La indicación la hace claramente *RStudio* en la equina superior izquierda. No sólo indica ahí el nombre del proyecto, si *Git* está habilitado, también indica la **rama** en la que estás trabajando. Si lo que se muestra es solamente **RStudio**, estás fuera de todo proyecto. Ve a **File → Open Project** o **File → Recent Projects** y abre el que deseas sea tu base de trabajo.



2. Después de hacer el *fork* con `usethis::create_from_github("equihuam/ciencia-reproducible", fork = TRUE)`, se hizo correctamente la copia, se obtuvo un clón del nuevo repositorio, etc. Sin embargo, el acceso al nuevo repositorio apareció inhabilitado (*pull* y *push* grises). Es seguro que esto se deba a un problema de configuración del proyecto. My probablemente tenga que ver con los permisos con los que estás operando y eso lo controla el token que tienes registrado. Pudieran ser también algunas otras cosas. Para empezar a explorar el problema lo mejor será usar `git_sitrepo` como sugiere Matthew Carson, que además de mostrarte detalles de configuración te sugerirá en dónde puede haber problemas y la forma de buscar solución al problema detectado.
3. En el vínculo entre *Github* y *Netlify* anotaste la rama de interés para publicar que no es la que querías. La solución es redefinir el repositorio de origen. En la figura te sugiero como acceder a esto. Básicamente la idea es reconstruir el vínculo entre *Github* y *Netlify* eligiendo los datos correctos de **rama** y **directorio**.




4. Encontramos inconsistencia con la definición de la rama que toma el papel *principal* (*default*) en tu **repo**. Sugiero resolver esto con el comando `git_default_branch_rename` de la biblioteca `usethis`. Úsalo en la pestaña de **consola** de *RStudio*. En el siguiente ejemplo estoy suponiendo que quieres cambiar de *master* a *main*. También estoy suponiendo que todo lo que te interesa está actualmente en *main*.

```
usethis::git_default_branch_rename(from = "master", to =  "main")
```

Una vez hecho esto, podría ocurrir que recordaras que hay cosas en *master* que quieres tener en *main*. Esto requiere un `git merge`.

Para hacer la fusión de *master* con *main*. No debe haber problema porque *main* la originaste a partir de *master*. Bajo estas condiciones puedes usar las operaciones siguientes y ejecutar algunos comandos más en la pestaña **terminal** de *RStudio*:

1. Cambia a la **rama** *main*, asegúrate que no tienes trabajo pendiente por registrar con un `commit`.
2. Navega en la **terminal** hasta asegurarte que estás en el directorio de tu proyecto.

```
cd "c:\[anota aquí la ruta a tu proyecto]"
```

3. Ubicada en tu directorio de interés invoca el siguiente comando de `git`

```
git merge master
```

Ahora *master* y *main* hacen referencia al mismo estado del proyecto, por lo tanto podrías considerar que uno de los dos sobra. En este caso, puede convenir borrar *master*. Lo puedes hacer con el comando siguiente en la pestaña **terminal** y desde el directorio de tu proyecto.

```
git branch -d master
```

Si resultara que la **rama** en cuestión, en realidad no es redundante, *git* te lo hará saber. Podrías obligar a *git* a eliminar *master* de cualquier manera, sólo cambia "-d" por "-D", pero hazlo con mucho cuidado, pues podrías perder algo que vas a echar de menos más adelante. Eric C. Anderson preparó esta explicación del tema, podría ser de tú interés.