

Comandos Avanzados Git - GitHub

Fork

Cuando hacemos un Fork en GitHub estamos creando una copia independiente de un repositorio ya existente en nuestro perfil. Cuando se hace un fork se trae una instancia completa del repositorio lo que incluye commits, ramas y archivos. El objetivo de realizar un fork es poder contribuir a un proyecto de un grupo o de código abierto, ya que al tener nuestra propia copia del repositorio, es posible modificar, solucionar problemas o agregar nuevas soluciones. Una vez realizados los cambios es posible realizar un pull request para que nuestros cambios se unan al repositorio original, pero esto dependerá del propietario del repositorio Original donde tiene la opción de revisar los cambios y aceptarlos o rechazarlos.

Pull Requests.

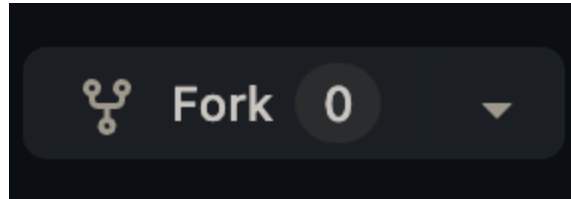
Los pull requests son una de las funciones principales que proporciona GitHub y otras plataformas que facilitan el trabajo colaborativo en los proyectos. Los pull requests no son más que propuestas de cambios a los repositorios, son propuestas ya que depende específicamente de los colaboradores con permisos de escritura o administrativos aceptar o rechazarlos.

Es importante que cuando se contribuya a un proyecto y se mande un pull request verificar los siguientes puntos:

- Verificar que código propuesto en el PR no contenga errores, y seguir las convenciones y estándares que especifican en los repositorios.
- Se resuelva el problema correctamente
- Que el PR no está en conflicto con otras partes del código base, ya que podría causar problemas de integración o comportamiento.
- Cumplir con la documentación ya que los proyectos pueden tener requisitos específicos sobre la documentación y pruebas unitarias

Flujo

1. Hacer un fork del proyecto que vamos a contribuir.



2. Especificar nombre del fork y la descripción

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk ().*

Owner *

Repository name *

DavidRamirez5

/ curriculum

✔ curriculum is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

The open curriculum for learning web development

☒ Copy the `main` branch only

Contribute back to TheOdinProject/curriculum by adding your own branch. [Learn more.](#)

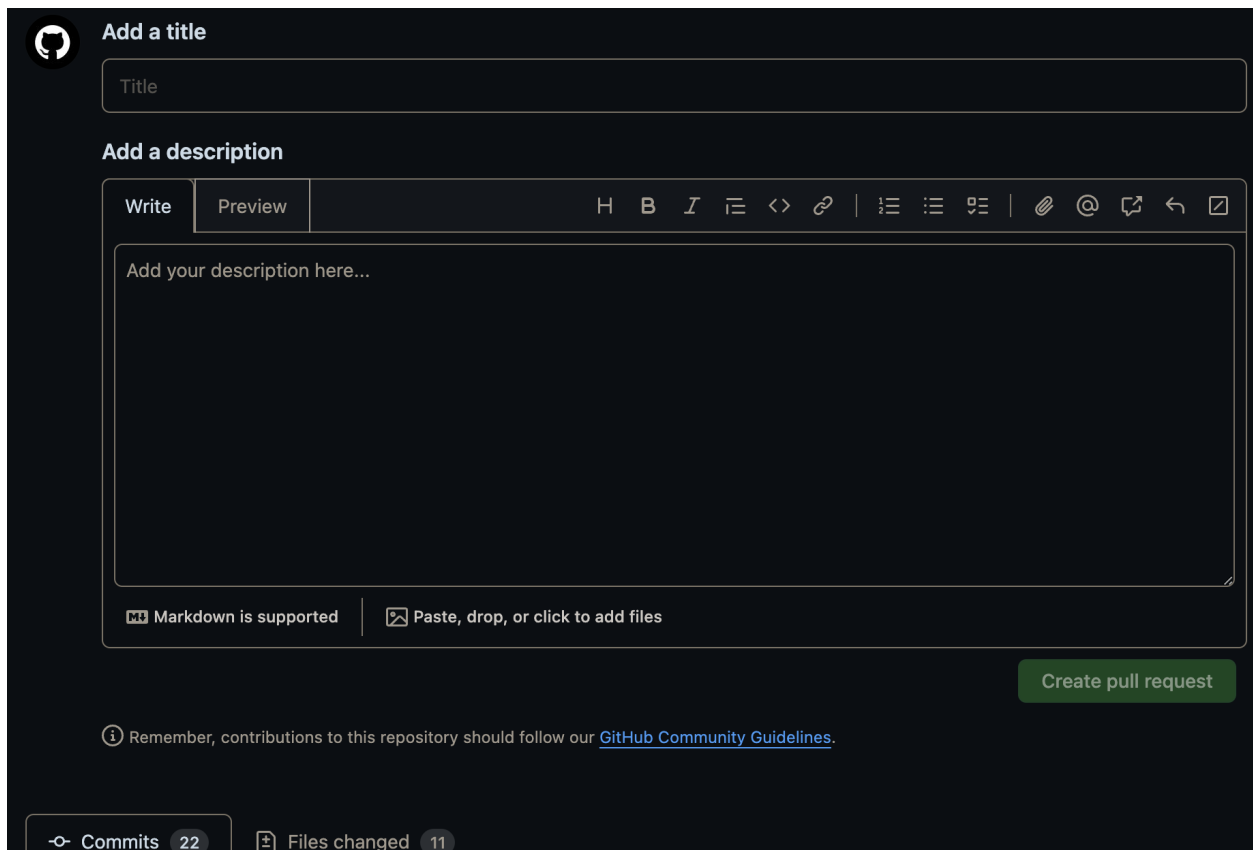
?

 You are creating a fork in your personal account.

Create fork

3. Una vez con la copia del repositorio podemos hacer un `git clone` , crear una rama `git branch` hacer los cambios al proyecto o soluciones para confirmarlos `git commit -m "..."` y posteriormente agregarlos `git push`
4. Enviar un pull request para que los cambios se agreguen al repositorio original donde agregaremos un titulo y una descripción de los cambios realizados. Los

cuales serán revisados y aceptados o rechazados por los propietarios del proyecto.



The image shows the GitHub interface for creating a pull request. At the top, there's a section 'Add a title' with a text input field labeled 'Title'. Below that is 'Add a description' with a 'Write' tab selected and a 'Preview' tab. The 'Write' tab has a rich text editor with various formatting options (H, B, I, etc.) and a large text area with the placeholder 'Add your description here...'. At the bottom of the description area, it says 'Markdown is supported' and 'Paste, drop, or click to add files'. A green button labeled 'Create pull request' is on the right. At the bottom of the form, there's a note: 'Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).' Below the form, there are two tabs: 'Commits' with a count of 22 and 'Files changed' with a count of 11.

Comandos Avanzados

Rebase

Este comando es utilizado para modificar el historial de commits en una rama. Es útil para mantener un historial de commits mas limpio y lineal.

```
git checkout feature-branch
git rebase main
# Este comando toma los commits de rama "feature-branch
# y los reubica sobre la punta de la rama main
```

Stash

El comando `stash` es utilizado para guardar temporalmente los cambios locales que aun no se han confirmado en un stage area.

```
# Guardando cambios aun no confirmados
git stash save "Cambios en progreso"

# Aplicar cambios guardados
git stash apply o git stash pop

# Eliminar el stash opcional
git stash drop
```

Clean

El comando `clean` es utilizado para eliminar archivos no rastreados del directorio de trabajo. Puede ser útil cuando sea necesario limpiar archivos generados por complicaciones o archivos temporales.

```
# comando que eliminara de forma forzada
# los archivos no rastreados
git clean -f

# Permite eliminar directorios no rastreados
git clean -d

# muestra que archivos se eliminarian
git clean -n
```

Cherry-pick

El comando `cherry-pick` es utilizado para aplicar un solo commit desde una rama a otra y es útil cuando se necesita aplicar un cambio específico a otra rama sin fusionar toda la rama.

```
git cherr-pick <hash-del-commit>
```

Reset

El comando `reset` es utilizado para mover la punta de una rama y el HEAD de la rama actual hacia otro commit. Existen diferentes opciones para resetear, como `--soft`, `--mixed` y `--hard` los cuales determinaran si los cambios en el area de trabajo y el indice se conservan o eliminar.

Bisect

El comando `bisect` se utiliza para encontrar el commit que causo un problema en el historial de commits. Es útil para realizar una búsqueda binaria eficiente en el historial para ubicar el commit especifico que introdujo un fallo.

```
#Inicia
git bisect start
# Marca el commit acutal como malo
git bisect bad
# Marca un commit conocido como bueno
git bisect good <commit>
```

David Patiño - Academia Java