

Comandos Avanzados de Git

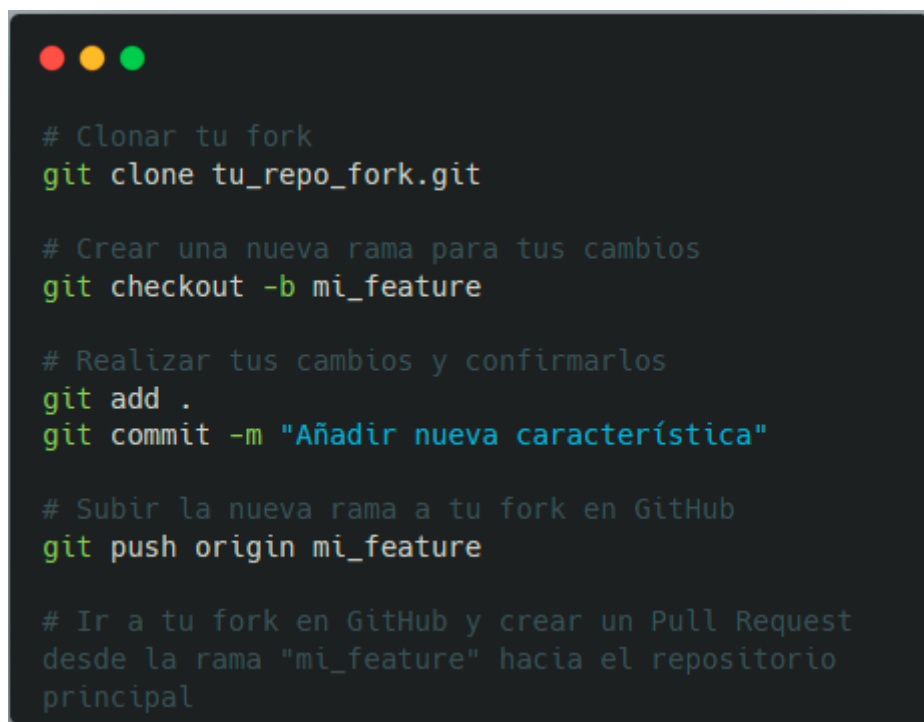
Pull Request

Un *Pull Request* (PR) es una función fundamental en plataformas como GitHub y GitLab. Se utiliza para proponer cambios en un repositorio y solicitar que estos cambios se integren en el proyecto principal.

-Creación de PR: Después de hacer cambios en tu propio repositorio (fork), puedes crear un PR para fusionar tus cambios en el repositorio principal. Esto facilita la colaboración y revisión de código.

-Revisión de código: Los PR permiten a otros colaboradores revisar tus cambios, comentarlos y aprobarlos antes de la fusión. Esto es esencial para mantener la calidad del código.

-Automatización de flujos de trabajo: Las plataformas de alojamiento de código suelen proporcionar herramientas para la integración continua (CI) y la integración continua de entrega (CD) que se activan automáticamente al crear un PR.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It displays a series of Git commands with comments in Spanish, explaining the steps to create a Pull Request. The commands are: cloning a fork, checking out a new branch named 'mi_feature', adding and committing changes with the message 'Añadir nueva característica', pushing the branch to the fork on GitHub, and a final instruction to create a Pull Request from 'mi_feature' to the main repository.

```
# Clonar tu fork
git clone tu_repo_fork.git

# Crear una nueva rama para tus cambios
git checkout -b mi_feature

# Realizar tus cambios y confirmarlos
git add .
git commit -m "Añadir nueva característica"

# Subir la nueva rama a tu fork en GitHub
git push origin mi_feature

# Ir a tu fork en GitHub y crear un Pull Request
desde la rama "mi_feature" hacia el repositorio
principal
```

Ejemplo pull request

Fork

Un *Fork* es una copia de un repositorio en el que puedes trabajar de forma independiente.

-Creación de Fork: Puedes realizar un Fork de un repositorio público para tener tu propia copia con la que trabajar.

-Contribución a proyectos de código abierto: Los Forks son especialmente útiles cuando deseas contribuir a proyectos de código abierto. Haces cambios en tu Fork y luego creas un PR para proponerlos al repositorio original.

-Sincronización con el repositorio original: Es importante mantener tu Fork actualizado con los cambios del repositorio original para evitar conflictos.

Si deseas contribuir a un proyecto de código abierto, primero realiza un fork del repositorio principal en tu cuenta de GitHub. Haz clic en el botón "Fork" en la esquina superior derecha. Esto creará una copia del repositorio en tu cuenta y clona tu fork a tu máquina local usando **git clone**.

Rebase

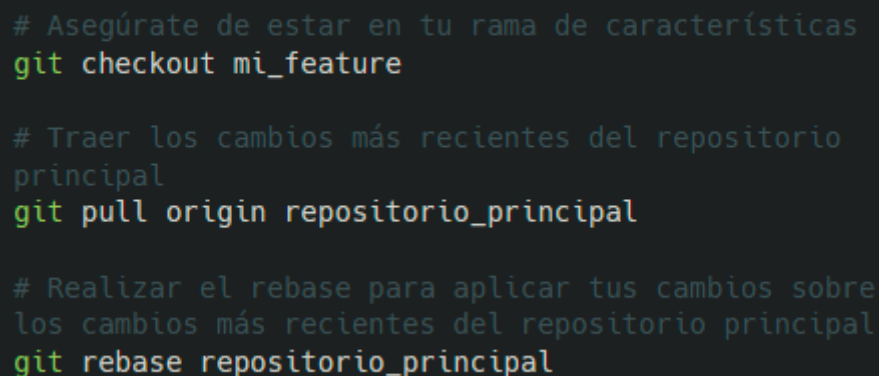
El comando *Rebase* se utiliza para reorganizar la historia de confirmaciones.

-Reescritura de historial: Permite reorganizar y combinar confirmaciones, lo que puede hacer que la historia del proyecto sea más limpia y coherente.

-Evitar conflictos: Puede ayudar a evitar conflictos durante la fusión (merge) de ramas al traer cambios del repositorio principal a tu rama antes de realizar tus propios cambios.

-Uso con precaución: Debe usarse con precaución, ya que puede causar problemas si se aplica a confirmaciones que ya se han compartido con otros colaboradores.

Imagina que tienes una rama de características que se ha separado del repositorio principal durante algún tiempo, y deseas traer los cambios más recientes del repositorio principal a tu rama. El rebase te ayuda a hacerlo de la siguiente manera:

A screenshot of a terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays a series of Git commands with their corresponding comments in Spanish. The commands are: 'git checkout mi_feature', 'git pull origin repositorio_principal', and 'git rebase repositorio_principal'. Each command is preceded by a comment line starting with '#'.

```
# Asegúrate de estar en tu rama de características
git checkout mi_feature

# Traer los cambios más recientes del repositorio principal
git pull origin repositorio_principal

# Realizar el rebase para aplicar tus cambios sobre los cambios más recientes del repositorio principal
git rebase repositorio_principal
```

Ejemplo rebase

Stash

El comando *Stash* se usa para guardar temporalmente los cambios no confirmados en una rama sin confirmarlos ni desecharlos.

-Guardado de cambios temporales: Es útil cuando necesitas cambiar de rama o realizar alguna acción que requiera un directorio de trabajo limpio sin perder tus cambios en progreso.

-Aplicación y eliminación de Stash: Puedes aplicar o eliminar los cambios almacenados en el Stash cuando sea conveniente.

-Múltiples Stashes: Puedes tener múltiples Stashes y nombrarlos para identificarlos más fácilmente.

Supongamos que estás trabajando en una rama con cambios no confirmados y necesitas cambiar a otra rama sin perder esos cambios. Puedes usar **stash**:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It displays four lines of Git commands, each preceded by a comment in Spanish. The commands are: 'git stash save "Cambios temporales"', 'git checkout otra_rama', 'git stash apply', and 'git stash drop'.

```
# Guardar los cambios en el stash
git stash save "Cambios temporales"

# Cambiar a otra rama
git checkout otra_rama

# Aplicar los cambios desde el stash
git stash apply

# Eliminar los cambios del stash
git stash drop
```

Ejemplo stash

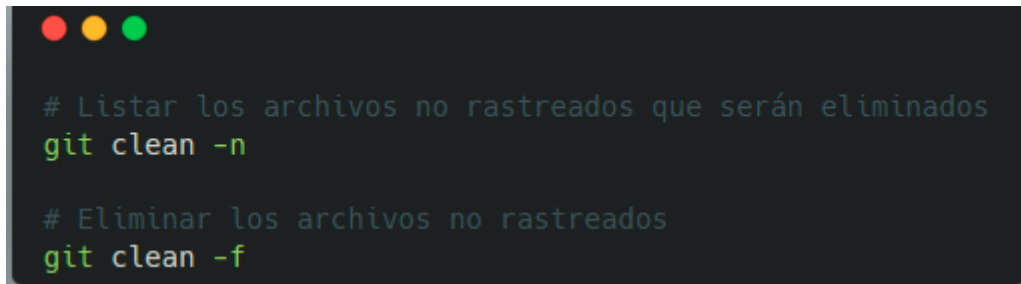
Clean

El comando *Clean* se utiliza para eliminar archivos y directorios no rastreados en el repositorio.

-Limpieza del espacio de trabajo: Ayuda a mantener el espacio de trabajo limpio eliminando archivos generados automáticamente o no deseados.

-Precaución: Debe usarse con precaución, ya que eliminará permanentemente los archivos no rastreados. Se recomienda revisar los cambios antes de ejecutar el comando.

Por ejemplo, para eliminar todos los archivos no rastreados:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains two lines of text: a comment in light blue and a command in green.

```
# Listar los archivos no rastreados que serán eliminados  
git clean -n  
  
# Eliminar los archivos no rastreados  
git clean -f
```

Ejemplo clean

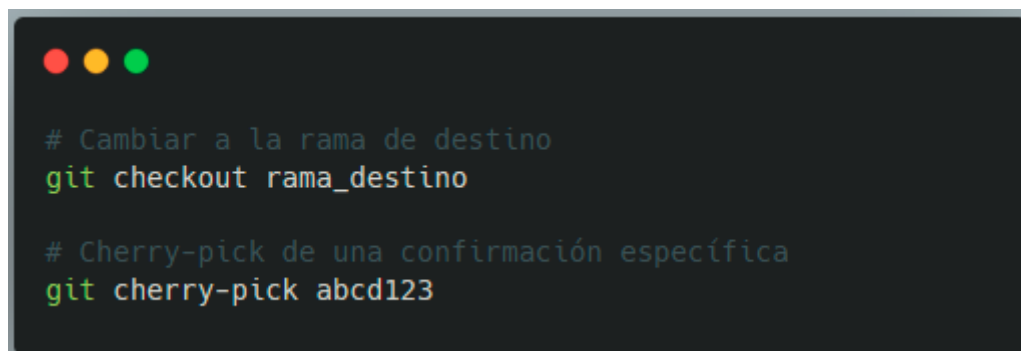
Cherry-pick

El comando *Cherry-pick* se utiliza para aplicar una confirmación específica de una rama a otra.

-Selección de confirmaciones: Permite seleccionar confirmaciones individuales y aplicarlas a otra rama sin la necesidad de fusionar ramas completas.

-Resolución de conflictos: Puede ser necesario resolver conflictos si las confirmaciones que intentas aplicar entran en conflicto con los cambios existentes en la rama de destino.

Por ejemplo, para aplicar la confirmación con el identificador `abcd123` de la rama `rama_fuente` a la rama actual:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains two lines of text: a comment in light blue and a command in green.

```
# Cambiar a la rama de destino  
git checkout rama_destino  
  
# Cherry-pick de una confirmación específica  
git cherry-pick abcd123
```

Ejemplo cherry-pick

Estos comandos avanzados de Git te permiten realizar operaciones más específicas y personalizadas en tu flujo de trabajo de control de versiones, facilitando la colaboración, la gestión del historial de confirmaciones y la limpieza de tu espacio de trabajo. Cada uno tiene su propia utilidad y debe utilizarse con cuidado para evitar problemas inesperados en el repositorio.