Comandos avanzados de Git

Góngora Martínez Adán Antonio

Pull Request

Un pull request en git es un mecanismo para proporcionar cambios a un proyecto. Un usuario crea un “pull request” en un repositorio, el cual pide que los mantenedores del repositorio revisen y unan los cambios propuestos. Los mantenedores pueden revisar los cambios, discutirlos con el usuario, y en última instancia decidir si aceptar o rechazar el pull request. Este proceso ayuda a asegurar que los cambios en un proyecto sean revisados y probados antes de que sean incorporados en el código base.

Para implementar un pull request en git, se necesitan los siguientes pasos:

1. Bifurque el repositorio en el que desea realizar cambios. Esto creará una copia del repositorio bajo tu propia cuenta.
2. Clonar el repositorio bifurcado a tu maquina usando el comando ‘git clone <url-repositorio>’.
3. Crear un nuevo Branch para los cambios usando el comando ‘git Branch <nombre-branch>’.
4. Cambiar al nuevo Branch usando el comando ‘git checkout <nombre-branch>’.
5. Hacer los cambios que quieres al nuevo código, y entonces hacer el stage y el commit usando ‘git add’ y ‘git commit’.
6. Hacer un push a los cambios de tu repositorio bifurcado en GitHub usando el comando ‘git push origin <nombre-branch>’.
7. Ir al repositorio original en GitHub y crear un nuevo pull request. En el pull request, seleccionar el Branch que se acaba de subir (push) como el branch de recursos, entonces subir el pull request.
8. El mantenedor del repositorio revisará los cambios y decidirá si unir o no los cambios con el Branch principal.

Este es el flujo básico de un pull request. Puede ser diferente dependiendo de las herramientas específicas y procesos usados por el equipo u organización.

Fork

Un fork es una copia de un repositorio. Permite la experimentación libre ya que los cambios no afectan al proyecto original. Se pueden hacer los cambios al repositorio bifurcado (fork), y subirl la pull request al repositorio original con los cambios propuestos. Esto es común mente usado en el desarrollo de software open-source como una manera para que los desarrolladores propongan cambios al proyecto original.

Rebase

Es un comando que permite hacer cambios en una Branch y aplicarlos en otra Branch. Esto es útil para mantener las branches actualizadas y resolver conflictos cuando multiples personas están trabajando en la misma Branch. Rebase cambia el historial de commits de la Branch cuando se usa. Es una alternativa a unir las branches, ayuda a mantener el historial de commit lineal y limpio. Es un

comando poderoso, pero debe ser usado con cuidado, ya que puede llevar a la perdida de trabajo si no es usado correctamente.

Para implementar el rebase, primero se necesita de cambiar al Branch a la que quieres hacer el rebase sobre otra Branch. Entonces, puedes usar el comando 'git rebase' seguido del nombre de la Branch en la que quieres hacer el rebase. Por ejemplo, si estas en la Branch "feature" y quieres hacer un rebase a la Branch "master", tendrías que ejecutar el comando "git rebase master" mientras estas en la Branch "feature".

Se debería de tener cuidado ya que hacer un rebase puede causar conflictos, en ese caso, git detiene el proceso y pedirá que se solucionen los conflictos. Una vez se hayan resuelto los conflictos, se puede continuar el proceso con el comando "git rebase –".

Siempre es una buena idea hacer un respaldo del trabajo antes de hacer un rebase, en caso de que algo vaya mal.

Stach

Un stach, stack o stash, es una manera de guardar temporalmente los cambios que se le haya hecho a un directorio de trabajo, pero que no esté listo para hacer un commit. Esto permite cambiar a un Branch distinto, por ejemplo, sin perder los cambios. Después se puede hacer un "pop" al stash para aplicar los cambios de nuevo, o un "drop" al stash para descartar los cambios.

Clean

Un clean (limpiar) se refiere a un estado donde no hay cambios sin hacerles commit ni archivos sin seguimientos en un repositorio de trabajo. Cuando se ejecuta el comando "git clean", se remueven todos los archivos sin seguimiento y directorios de un árbol de trabajo. Este comando puede ser útil para mantener el directorio de trabajo libre de desorden y asegurando que solo estén presentes los archivos rastreados por git. Como sea, es importante ser cuidadoso cuando se usa un "git clean", ya que borra permanentemente los archivos y no puede ser des hecho.

Cherry-pick

Es un comando que permite seleccionar un commit especifico de una Branch y aplicarlos a otro Branch. Esto puede ser útil cuando quieres unir selectivamente los cambios de una Branch en otra, en lugar de unir la Branch entera. El comando Cherry-pick crea un nuevo commit en la Branch seleccionada con los mismos cambios que el commit seleccionado.