Github

(Pull request, Fork, Rebase, Stach, Clean & Cherry-pick)

Pull request

El pull request también conocido como "solicitud de extracción" es una herramienta que permite la colaboración y el control de calidad en proyectos de software colaborativos, facilitando la revisión y la integración de cambios realizados por diferentes desarrolladores. Un pull request permite que un colaborador envíe una sugerencia de cambios al propietario del repositorio para que sean revisados y, si se aceptan, estos cambios son fusionados en la rama principal o preferida del repositorio..

Para crear un pull request, el colaborador hace una copia del repositorio en su propia cuenta de GitHub, crea una nueva rama y realiza los cambios necesarios. Luego, el colaborador envía un pull request desde su rama hacia la rama principal del repositorio original. En este proceso, el colaborador puede añadir comentarios, describir los cambios realizados y explicar su propósito.

El propietario del repositorio original puede entonces revisar los cambios propuestos, hacer comentarios y solicitar cambios adicionales si es necesario. Si se aceptan los cambios, el propietario del repositorio puede fusionar los cambios en la rama principal del repositorio. Los pull requests son una forma efectiva de colaborar en proyectos de código abierto y facilitan la contribución de nuevos colaboradores al proyecto.

Fork

Un fork en GitHub es una operación usual en el sistema de Git que se encarga de la creación de una copia de un repositorio de código en la cuenta del usuario que realiza el fork. se crea una copia del repositorio original en su propia cuenta de GitHub, lo que le permite trabajar en el código sin afectar al repositorio original. El nuevo repositorio copiado será igual al repositorio desde el que se realiza el fork en Git. A pesar de esto, cuando se cree la copia, cada repositorio se ubicará en espacios diferentes y tendrán la posibilidad de evolucionar de forma diferente

Los forks son una forma común de colaboración en GitHub. Al hacer un fork, un desarrollador puede trabajar en una copia del código existente, hacer cambios y mejoras, y luego enviar un pull request al repositorio original para que los cambios sean incorporados. Una mejor forma de entender este concepto de colaboración y su propósito es ver a los forks y a los pull request como los branches y el merge, es el mismo concepto de crear una copia de código, modificarlo y después mandar una petición de unir el código modificado con el original.

Rebase

Este comando así también como el comando git merge, tiene la función de hacer fusión de código, es decir, de integrar cambios de rama a otra. A esta función también se le conoce como fusión por cambio de base y la diferencia con el git merge es que al usar git merge los cambios realizados en una rama secundaria (remota)se agregan a la rama principal (main) en forma de un nuevo commit. Esto puede ocasionar que el historial de la rama principal se vuelva confuso y difícil de seguir si hay muchos commits.

Es por ello que se opta por usar el git rebase, que aplica los cambios realizados en una rama sobre la otra rama de manera lineal, como si los cambios hubieran sido realizados directamente sobre la rama objetivo, es decir, este comando toma una serie de commits realizados en un Branch y los reproduce sobre la rama principal en donde se quiera ejecutar el comando, creando asi una secuencia de commits que parecen haber sido realizados directamente en la rama objetivo. El motivo principal del uso de este comando es para mantener un historial de commits del proyecto limpio y lineal

Comando para aplica el git rebase

```
$ git rebase <base>
```

este comando se utiliza para aplicar los cambios de la rama especificada a la rama actual

```
git rebase -i <commit>
```

Este comando permite editar la historia de confirmaciones de la rama actual de una manera más interactiva. El usuario puede reordenar, modificar o eliminar confirmaciones antes de aplicarlas a otra rama.

Stash

Git stash es un comando de Git que permite a los desarrolladores almacenar temporalmente cambios locales que no están listos para ser confirmados. El hecho de guardar los cambios en stashes resulta práctico si se tiene que cambiar rápidamente de rama y hacer otra actividad, de esta manera se puede salvar código sin tener la necesidad de hacer commit al mismo.

El comando toma los cambios sin confirmar (tanto los que están preparados como los que no) los guarda y acto seguido, los deshace en el código en el que se está trabajando y vuelve al último commit confirmado.

Comando para realizar un stash

```
$ git stash
```

Comando para volver a aplicar los cambios de un stash

```
$ git stash pop
```

Al hacer pop del stash, se eliminan los cambios de este y se vuelven a aplicar en el código en el que estás trabajando.

Otra opción es volver a aplicar los cambios en el código en el que estás trabajando y conservarlos en tu stash mediante el comando git stash apply:

```
$ git stash apply
```

Clean

Git clean es un comando de Git que se utiliza para eliminar archivos no rastreados o sin seguimiento del directorio de trabajo. Los archivos no rastreados son aquellos que no están bajo control de versión de Git, no se han añadido al índice de seguimiento del repositorio con git add y que no están incluidos en el archivo .gitignore. Es importante saber que el comando git clean elimina permanentemente los archivos no rastreados del directorio de trabajo y que no se pueden recuperar fácilmente una vez eliminados.

Comando que sirve para mostrar los archivos que serán eliminados sin eliminarlos realmente

```
$ git clean -n
```

Comando que elimina los archivos no rastreados del directorio de trabajo sin pedir confirmación.

```
$ git clean -f
```

Comando que elimina los directorios no rastreados del directorio de trabajo.

```
$ git clean -d
```

Cherry-pick

Git cherry-pick es un comando de Git que se utiliza para aplicar una confirmación (commit) específica de una rama a otra rama. Esto puede ser útil si se desea incorporar un cambio específico de una rama a otra rama sin fusionar todas las confirmaciones de la rama original, por ejemplo, supongamos que una confirmación se aplica accidentalmente en la rama equivocada. Puedes cambiar a la rama correcta y ejecutar cherry-pick en la confirmación para aplicarla a donde debería estar.

Este comando permite que las confirmaciones arbitrarias de Git se elijan por referencia y se añadan al actual HEAD de trabajo. La ejecución de cherry-pick es el acto de elegir una confirmación de una rama y aplicarla a otra. git cherry-pick puede ser útil para deshacer cambios.

Para usar el comando **`$ git cherry-pick`**, primero se debe identificar el hash de la confirmación que se desea aplicar. Luego, se debe ejecutar el comando

`$ git cherry-pick <hash>` en la rama de destino para aplicar los cambios.

Es importante tener en cuenta que el uso de este comando puede generar conflictos de fusión si la confirmación que se desea aplicar depende de otras confirmaciones que no están presentes en la rama de destino. Además, si se aplica una confirmación que ya existe en la rama de destino, esto puede generar duplicados en la historia de confirmaciones. Por lo tanto, se recomienda usarlo con precaución