Git & GitHub

Advanced Commands (Stach, Clean, Cherry-pick)

STACH

Git tiene un área llamada "stash" donde puedes almacenar temporalmente una captura o instantánea de tus cambios sin enviarlos al repositorio. Está separada del directorio de trabajo (working directory), del área de preparación (staging area), o del repositorio de Git.
Esta funcionalidad es útil cuando has hecho cambios en una rama que no estás listo para realizarle commit, pero es necesario cambiarla a otra rama.

Guardar cambios en el Stash
Para guardar tus cambios en el stash, ejecuta el comando:

```
$ git stash save "mensaje opcional para ti"
```

Este comando guarda los cambios y revierte el directorio de trabajo a como se veía en tu último commit. Los cambios guardados están disponibles en cualquier rama de ese repositorio.
Hay que tener en cuenta que los cambios que se quieran guardar deben estar en los archivos rastreados. Si se ha creado un nuevo archivo y se intenta guardar los cambios, puede que se obtenga el error *No local changes to save (No hay cambios locales que guardar)*.

Ver cambios guardados en el Stash
Para ver los cambios guardados en el stash ejecutar el comando:

```
$ git stash list
```

Esto devolverá una lista de las capturas guardadas en el formato:

stash@{0}: RAMA-STASHED-CAMBIOS-SON-PARA: MESSAGE

La parte de `stash@{0}` es el **nombre del stash**, y el `número en las llaves ({ })` es el **índice (index) del stash**. Si se tienen múltiples conjuntos de cambios guardados en stash, cada uno tendrá un índice diferente.

Para ver un resumen de los cambios que se hicieron en el stash se utiliza el comando:

```
$ git stash show NOMBRE-DEL-STASH
```

Si quieres ver el típico diseño "diff" (con los + y - en las líneas con los cambios), puedes incluir la opción `-p` (para parche o patch) entre `show` y `NOMBRE-DEL-STASH`.

Recuperar cambios en Stash
Para recuperar los cambios del stash y aplicarlos a la rama actual, se tienen dos opciones:

1. Aplica los cambios y deja una copia en el stash:

```
$ git stash apply NOMBRE-DEL-STASH
```

2. Aplica los cambios y elimina los archivos del stash:

```
$ git stash pop NOMBRE-DEL-STASH
```

Borrar los cambios guardados en Stash
Para remover los cambios guardados en stash sin aplicarlos, ejecutar el comando siguiente:

```
$ git stash drop NOMBRE-DEL-STASH
```

Para limpiar todo del stash, ejecuta el comando:

```
$ git stash clear
```

CLEAN

El comando git clean se utiliza para eliminar archivos no deseados o limpiar tu directorio de trabajo. Esto podría incluir la eliminación de artefactos de construcción temporal o la fusión de archivos en conflicto. El comando clean actúa en archivos sin seguimiento, este tipo de archivos son aquellos que se encuentran en el directorio de trabajo, pero que aún no se han añadido al índice de seguimiento de repositorio con el comando add.

```
$ git clean
```

La ejecución del comando predeterminado puede producir un error. La configuración global de Git obliga a usar la opción `force` con el comando para que sea efectivo. Se trata de un importante mecanismo de seguridad ya que este comando no se puede deshacer.

Revisar que archivos no tienen seguimiento.

```
$ git clean -dry-run
```

Eliminar los archivos listados de no seguimiento. El `-f` significa fuerza o "realmente haz esto".

```
$ git clean -f
```

CHERRY-PICK

Es un comando en Git que selecciona y aplica commits específicos de una rama o branch a otra. Todo, sin tener que hacer un merge completo. Así, podemos copiar un commit específico y aplicarlo de forma aislada en la rama de destino, conservando su historial.
Este comando facilita la incorporación precisa de cambios, optimizando la colaboración y el mantenimiento en proyectos de desarrollo de software. Algunos casos de uso pueden ser:

1. Colaboración en equipo
Antes que nada, puede ser útil implementarlo cuando diferentes miembros del equipo trabajan en áreas similares del código, pues permite seleccionar y aplicar commits específicos a cada rama, facilitando el progreso individual.

2. Solución de bugs o errores
Cuando encuentras un error o bug, es importante solucionarlo y entregar la corrección a los usuarios lo más rápido posible. Con Git Cherry Pick, puedes aplicar rápidamente un commit de verificación en la rama principal, evitando que afecte a más usuarios.

3. Deshacer cambios y recuperar commits perdidos
A veces, una rama de funcionalidad puede ser obsoleta y no ser fusionada con la rama principal. O puede suceder que una solicitud de extracción (pull request) sea cerrada sin ser fusionada.

¿Cómo funciona Git Cherry Pick? Ejemplos
Imaginemos que tienes un repositorio con el siguiente estado de las ramas:

```
a - b - c - d  Rama Principal
     \\
      e - f - g  Rama de Características
```

El uso de Git Cherry Pick es bastante sencillo y se ejecuta de la siguiente manera:

1. Hay que asegurarse de estar en la rama principal empleando el comando:

```
$ git checkout rama-principal
```

2. Luego, ejecuta el siguiente comando:

```
$ git cherry-pick commitSha
```

Aquí, commitSha es una referencia al commit que deseas aplicar. Se puede encontrar la referencia del commit utilizando el comando git log. Supongamos que deseas usar el commit 'f' en la rama principal.
Una vez ejecutado el comando, el historial de Git se verá así:

```
a - b - c - d - f  Rama Principal
        \\
         e - f - g  Rama de Características
```

De esta forma, el commit 'f' se ha incorporado correctamente a la rama principal.
El uso de Git Cherry Pick debería aplicarse con mucho cuidado, ya que puede generar duplicación de commits y complicaciones en el historial de cambios.

Deshacer Cherry-Pick en caso de conflicto
Cuando has realizado un cherry-pick de un commit de otra rama la rama local, pero ocurren conflictos durante este proceso y se desea detenerlo y volver al estado anterior, En este caso, se puede emplear el siguiente comando:

```
$ git cherry-pick --abort
```

Esto significa que se pueden hacer las correcciones necesarias en tu rama local y volver a intentar el cherry-pick si así se desea.

REFERENCIAS:

Git Stash
https://www.freecodecamp.org/espanol/news/git-stash-explicado/
https://git-scm.com/book/es/v2/Herramientas-de-Git-Guardado-r%C3%A1pido-y-Limpieza
https://www.freecodecamp.org/espanol/news/como-usar-el-comando-git-stash/

Git Clean
https://git-scm.com/book/es/v2/Herramientas-de-Git-Guardado-r%C3%A1pido-y-Limpieza
https://platzi.com/clases/1557-git-github/19983-git-clean-limpiar-tu-proyecto-de-archivos-no-desea/
https://git-scm.com/book/es/v2/Ap%C3%A9ndice-C%3A-Comandos-de-Git-Seguimiento-B%C3%A1sico

Git Cherry-Pick
https://platzi.com/clases/1557-git-github/19982-git-cherry-pick-traer-commits-viejos-al-head-de-un/
https://git-scm.com/book/es/v2/Ap%C3%A9ndice-C%3A-Comandos-de-Git-Parcheo
https://git-scm.com/docs/git-cherry-pick
https://www.atlassian.com/es/git/tutorials/cherry-pick