La mejor forma de entender estos mecanismos es creando un conjunto de cambios en un repositorio y siguiéndolos a través de los tres árboles. Averigüémoslo.

$ mkdir git\_reset\_test

$ cd git\_reset\_test/

$ git init .

Initialized empty Git repository in /git\_reset\_test/.git/

$ touch reset\_lifecycle\_file

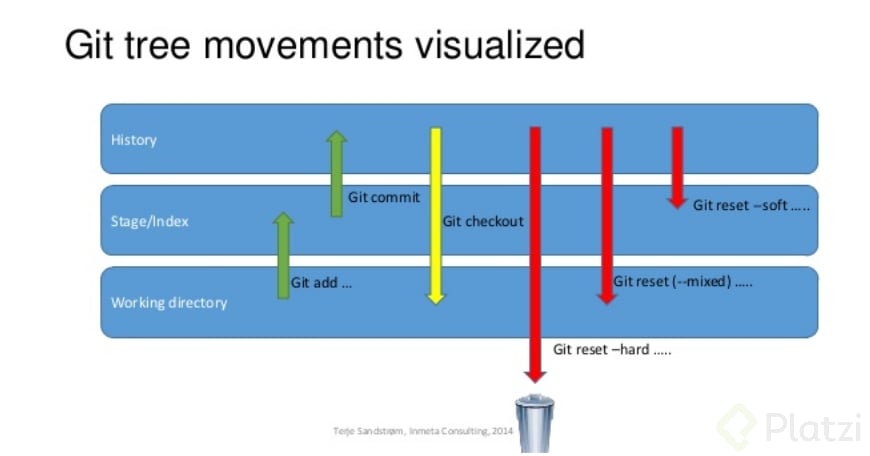
$ git add reset\_lifecycle\_file

$ git commit -m"initial commit"

[main (root-commit) d386d86] initial commit

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 reset\_lifecycle\_file



¿Cómo funciona Git Reset en tu flujo de trabajo?

git reset permite moverte entre diferentes commits para deshacer o rehacer cambios. Git guarda todos lo nuevo del repositorio como commits, que son instantáneas del estado del código en un momento dado y existen variaciones de este comando.

Variaciones de Git Reset

* git reset --soft: Borra el historial y los registros de Git de commits anteriores, pero guarda los cambios en Staging para aplicar las últimas actualizaciones a un nuevo commit.
* git reset --hard: Deshace todo, absolutamente todo. Toda la información de los commits y del área de staging se elimina del historial.
* git reset --mixed: Borra todo, exactamente todo. Toda la información de los commits y del área de staging se elimina del historial.
* git reset HEAD: El comando git reset saca archivos del área de staging sin borrarlos ni realizar otras acciones. Esto impide que los últimos cambios en estos archivos se envíen al último commit. Podemos incluirlos de nuevo en staging con git add si cambiamos de opinión.

Ten en cuenta que, si deshaces commits en un repositorio compartido en GitHub, estarás cambiando su historia y esto puede causar problemas de sincronización con otros colaboradores.

¿Qué es git reset HEAD?

git reset HEAD es un comando que te permite revertir los cambios que ya habías preparado para subir, y moverlos de vuelta a tu proyecto. Con este comando puedes cancelar los cambios que ya habías agregado, para que puedas revisarlos, modificarlos o deshacerlos antes de confirmarlos con un commit.

Git rm

Por otro lado, git rm es un comando que nos ayuda a eliminar archivos de Git sin eliminar su historial del sistema de versiones. Para recuperar el archivo eliminado, necesitamos retroceder en la historia del proyecto, recuperar el último commit y obtener la última confirmación antes de la eliminación del archivo.

Es importante tener en cuenta que git rm no puede usarse sin evaluarlo antes. Debemos usar uno de los flags siguientes para indicarle a Git cómo eliminar los archivos que ya no necesitamos en la última versión del proyecto.

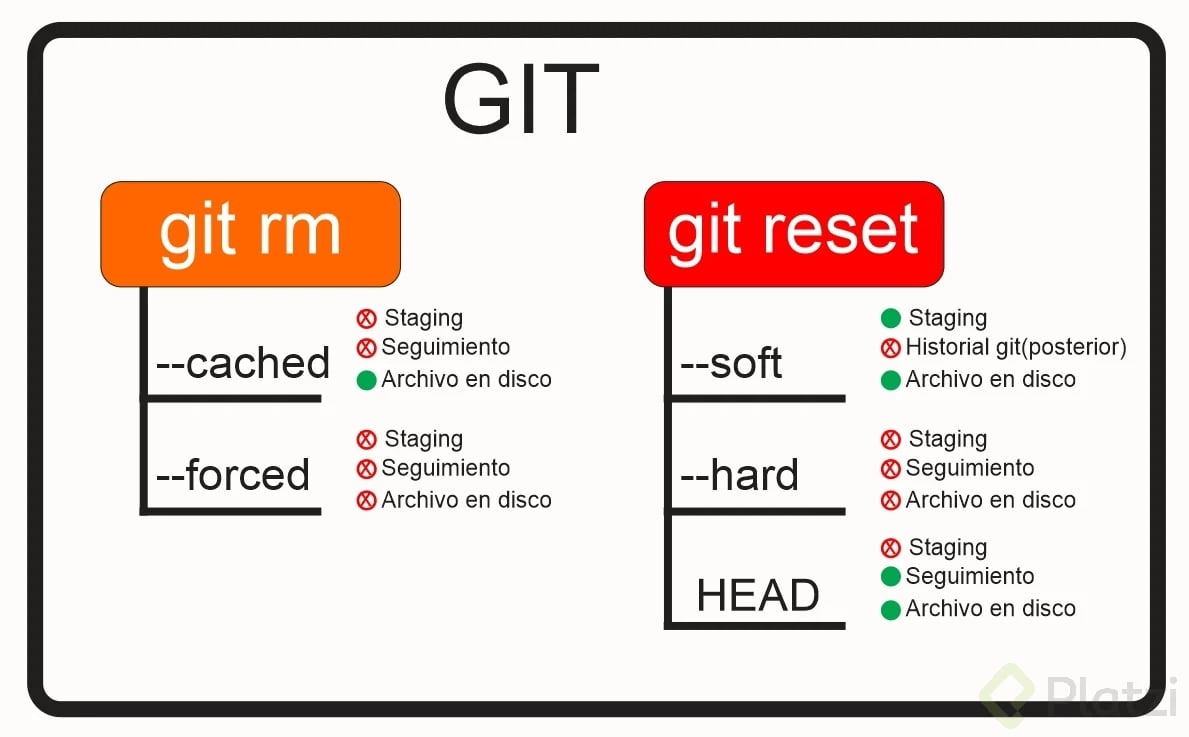
Variaciones de Git rm

* git rm --cached: Elimina archivos del repositorio local y del área de staging, pero los mantiene en el disco duro. Deja de trackear el historial de cambios de estos archivos, por lo que quedan en estado untracked.
* git rm --force: Elimina los archivos de Git y del disco duro. Git guarda todo, por lo que podemos recuperar archivos eliminados si es necesario (empleando comandos avanzados).

¡Al usar git rm lo que haremos será eliminar este archivo completamente de git!

¿Cuál es la diferencia entre git rm y git reset Head?

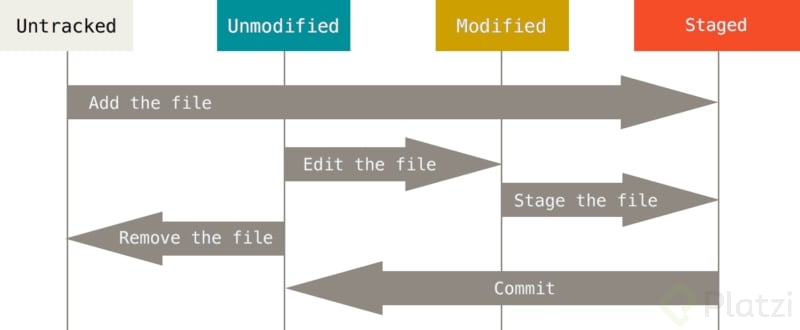
La diferencia principal entre git rm y git reset HEAD radica en que git rm elimina archivos del repositorio y de la historia del proyecto, mientras que git reset saca los cambios del área de preparación y los mueve del espacio de trabajo, sin afectar la historia del repositorio.



Es importante tener en cuenta el efecto que cada comando tiene en el proyecto y usarlos según tus necesidades y objetivos específicos.

¿Cuándo utilizar git reset en lugar de git revert?

Para reescribir la historia del repositorio y eliminar confirmaciones anteriores, se utiliza git reset. Para deshacer cambios de confirmaciones anteriores de forma segura sin modificar la historia del repositorio, se emplea git revert.



Resumen

Para evitar problemas en el trabajo, es valioso entender las implicaciones y riesgos de cada comando y elegir el enfoque adecuado según las necesidades y el flujo de trabajo del proyecto.

Con git rm eliminamos un archivo de Git, pero mantenemos su historial de cambios. Si no queremos borrar un archivo, sino dejarlo como está y actualizarlo después, no debemos usar este comando en este commit.

Empleando git reset HEAD, movemos los cambios de Staging a Unstaged, pero mantenemos el archivo en el repositorio con los últimos cambios en los que hicimos commit. Así, no perdemos nada relevante.

Siguientes pasos

Bueno, todos los cambios están en el área de Staging, incluido el archivo con los cambios que no están listos. Esto significa que debemos sacar ese archivo de Staging para poder hacer commit de todos los demás.