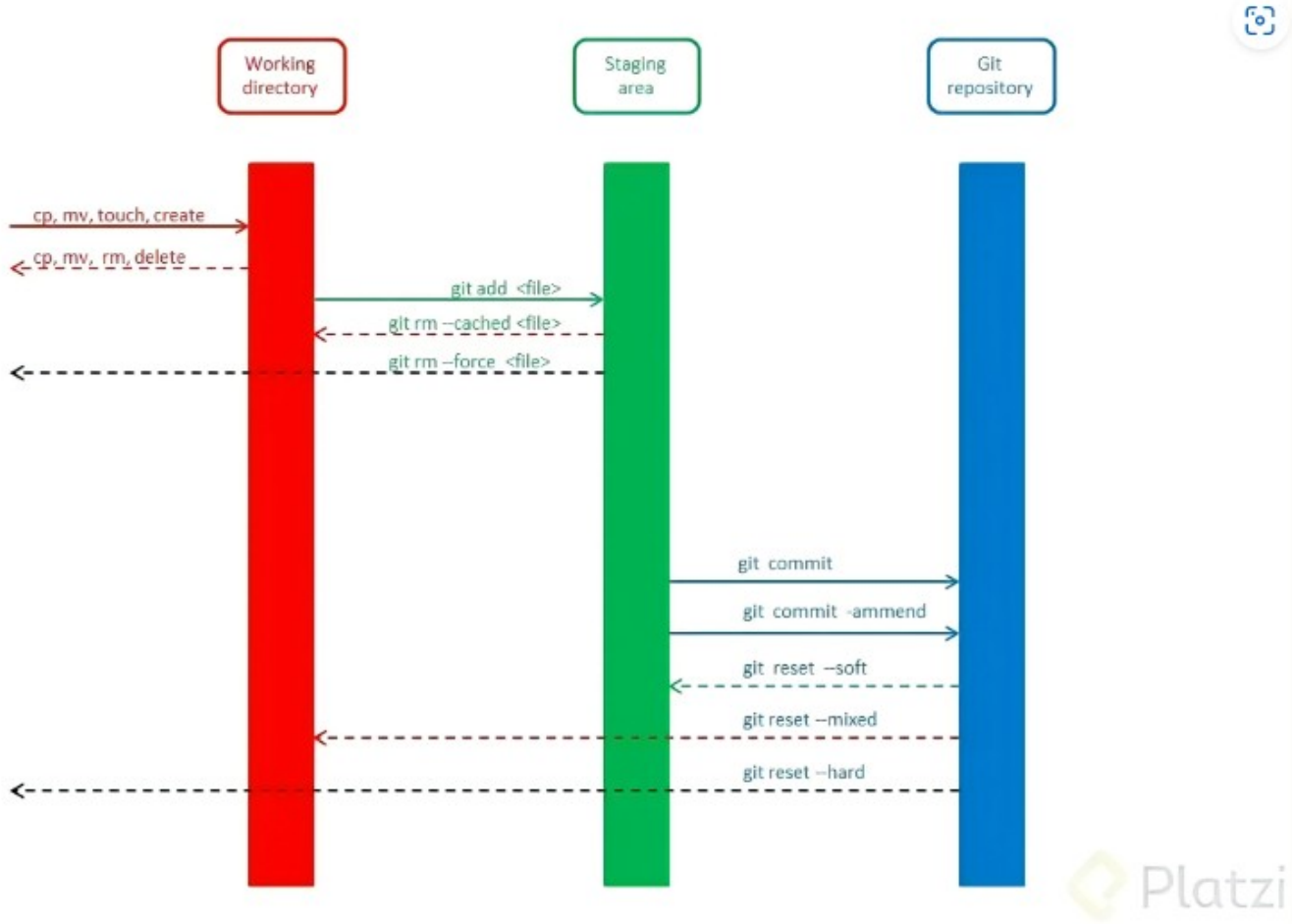
¿Qué es el staging?
El staging es el lugar donde se guardan temporalmente los cambios, para luego ser llevados definitivamente al repositorio. El repositorio es el lugar donde se guardan todos los registros de los cambios realizados a los archivos.
Para iniciar un repositorio, o sea, activar el sistema de control de versiones de Git en tu proyecto, solo debes ejecutar el comando git init.
¿Qué es el área de staging?
El área de staging se puede ver como un limbo donde nuestros archivos están por ser enviados al repositorio o ser regresados a la carpeta del proyecto.
¿Qué es git init?
git inites el comando que activa git en nuestro proyecto creando un espacio en memoria RAM llamado staging y una carpeta .git.
Este comando se encargará de dos cosas: primero, crear una carpeta .git, donde se guardará toda la base de datos con cambios atómicos de nuestro proyecto; segundo, crear un área que conocemos como **staging**, que guardará temporalmente nuestros archivos (cuando ejecutemos un comando especial para eso) y nos permitirá, más adelante, guardar estos cambios en el repositorio (también con un comando especial).

Cómo funciona el staging y el repositorio: ciclo básico de trabajo en git:
El flujo de trabajo básico en git es algo así:

1. Modificas una serie de archivos en tu directorio de trabajo.
2. Preparas los archivos, añadiéndolos a tu área de preparación (staging).
3. Confirmas los cambios (commit), lo que toma los archivos tal y como están en el área de preparación y almacena esa copia instantánea de manera permanente en tu directorio de git.

Veamos a detalle las 3 secciones principales que tiene un proyecto en git.

Working directory
El working directory es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de git y se colocan en el disco para que los puedas usar o modificar.
Staging area
Es un área que almacena información acerca de lo que va a ir en tu próxima confirmación. A veces se le denomina índice (index).
.git directory (repository)
En el repository se almacenan los metadatos y la base de datos de los objetos para tu proyecto. Es la parte más importante de git (carpeta .git) y es lo que se copia cuando clonas un repositorio desde otra computadora.



Ciclo de vida o estados de los archivos en git
Cuando trabajamos con git, nuestros archivos pueden vivir y moverse entre 4 diferentes estados (cuando trabajamos con repositorios remotos pueden ser más estados, pero lo estudiaremos más adelante):

Archivos tracked
Son los archivos que viven dentro de git, no tienen cambios pendientes y sus últimas actualizaciones han sido guardadas en el repositorio gracias a los comandos `git add` y `git commit`.
Archivos staged
Son archivos en staging. Viven dentro de git y hay registro de ellos porque han sido afectados por el comando `git add`, aunque no sus últimos cambios. Git ya sabe de la existencia de estos últimos cambios, pero todavía no han sido guardados definitivamente en el repositorio porque falta ejecutar el comando `git commit`.
Archivos unstaged
Entiéndelos como archivos "tracked pero unstaged". Son archivos que viven dentro de git pero no han sido afectados por el comando `git add` ni mucho menos por `git commit`. Git tiene un registro de estos archivos, pero está desactualizado, sus últimas versiones solo están guardadas en el disco duro.
Archivos untracked
Son archivos que NO viven dentro de git, solo en el disco duro. Nunca han sido afectados por `git add`, así que git no tiene registros de su existencia.
Recuerda que hay un caso muy raro donde los archivos tienen dos estados al mismo tiempo: staged y untracked. Esto pasa cuando guardas los cambios de un archivo en el área de staging (con el comando `git add`), pero antes de hacer commit para guardar los cambios en el repositorio haces nuevos cambios que todavía no han sido guardados en el área de staging.
Comandos para mover archivos entre los estados de Git
Estos son los comandos más importantes que debes conocer:
Git status
`git status` nos permite ver el estado de todos nuestros archivos y carpetas.
Git add
`git add` nos ayuda a mover archivos del untracked o unstaged al estado staged. Podemos usar `git nombre-del-archivo-o-carpeta` para añadir archivos y carpetas individuales o `git add -A` para mover todos los archivos de nuestro proyecto (tanto untrackeds como unstageds).
Git reset HEAD
Nos ayuda a sacar archivos del estado staged para devolverlos a su estado anterior. Si los archivos venían de unstaged, vuelven allí. Y lo mismo se venían de untracked.
Git commit
Nos ayuda a mover archivos de unstaged a tracked. Esta es una ocasión especial, los archivos han sido guardados o actualizados en el repositorio. Git nos pedirá que dejemos un mensaje para recordar los cambios que hicimos y podemos usar el argumento `m` para escribirlo (`git commit -m "mensaje"`).

Git rm

Este comando necesita alguno de los siguientes argumentos para poder ejecutarse correctamente:

- `git rm --cached`: mueve los archivos que le indiquemos al estado untracked.
- `git rm --force`: elimina los archivos de git y del disco duro. Git guarda el registro de la existencia de los archivos, por lo que podremos recuperarlos si es necesario (pero debemos usar comandos más avanzados).

Contribución creada con los aportes de: Diego Camacho, Jesús Sarabia y Angelo Yenque.