Advanced commands (Stach, Clean, Cherry pick, etc.)

1. git stash:

Este comando se utiliza para guardar temporalmente los cambios en el directorio de trabajo que aún no se han confirmado en un commit, permitiendo así cambiar de rama o realizar otras operaciones sin comprometer los cambios actuales.

- *Uso:*

  git stash [save "mensaje"]

- *Ejemplo:*

  git stash save "Cambios temporales antes de cambiar de rama"

2. git clean:

“git clean” se utiliza para eliminar archivos no rastreados en el directorio de trabajo. Los archivos que aún no han sido agregados al índice (staged) serán eliminados de manera permanente.

- *Uso:*

  git clean [opciones]

- *Ejemplo:*

  git clean -f -d ------> # Elimina archivos y directorios no rastreados

3. git cherry-pick:

Este comando se utiliza para aplicar un commit específico de una rama a otra. Permite seleccionar un commit individual y aplicarlo en la rama actual.

- *Uso:*

  git cherry-pick <hash-del-commit>

- *Ejemplo:*

  git cherry-pick abc123 ------> # Aplica el commit con hash "abc123" en la rama actual

4. git revert:

Crea un nuevo commit que revierte los cambios introducidos por un commit específico. No modifica la historia existente, pero añade un nuevo commit para deshacer los cambios.

- Uso:

- git revert <hash-del-commit>

- Ejemplo:

- git revert abc123 ------> # Crea un nuevo commit que deshace los cambios introducidos por el commit con hash "abc123"

5. git reset:

Permite retroceder cambios en el historial. Puede ser utilizado de diferentes maneras, como --soft para retroceder commits manteniendo los cambios en el área de preparación, --mixed para también deshacer los cambios en el área de preparación, o --hard para deshacer cambios en el directorio de trabajo, el área de preparación y el historial.

- Uso:

- git reset [--soft | --mixed | --hard] <hash-del-commit>

- Ejemplo:

- git reset --soft abc123 ------> # Retrocede al commit con hash "abc123" manteniendo los cambios en el área de preparación

6. git bisect:

Ayuda a encontrar el commit que introdujo un bug, realizando una búsqueda binaria en el historial.

- Uso:

- git bisect start
- git bisect good <commit-bueno>
- git bisect bad <commit-malo>

- Ejemplo:

- git bisect start
- git bisect good v1.0 # Indica un commit bueno
- git bisect bad v2.0 # Indica un commit malo
- git bisect run prueba-script.sh ------> # Automatiza la búsqueda del commit problemático ejecutando un script de prueba

7. git reflog:

Muestra un registro detallado de todas las operaciones realizadas en el repositorio, incluyendo cambios en ramas, reseteos, etc. Es útil para recuperar accidentalmente eliminaciones o cambios no deseados.

- Uso:

- git reflog

- Ejemplo:

- git reflog ------> # Muestra el registro detallado de operaciones realizadas

8. git cherry:

Compara dos ramas y muestra los commits que existen en una rama pero no en la otra. Útil para identificar commits que aún no se han aplicado en otra rama.

- Uso:

- git cherry <rama-origen> <rama-destino>

- Ejemplo:

- git cherry feature-branch master ------> # Compara la rama "feature-branch" con "master" y muestra los commits que no están en "master"