Resumen de Seccion:

En esta sección del curso aprenderás:

* Como regresar a etapas anteriores en:
* Working Directory
* Stage
* Commit

¿cómo arreglas errores? ¿cómo recuperar versiones buenas de código escritas previamente?

**Primeramente veamos cómo reparar algo que se encuentra en el working directory**

* El working directory se podría decir que se divide en dos partes, los archivos trackeados y los trackeados, de esta forma el comando git clean sólo limpia los archivos no trackeados

Por ejemplo voy a crear en la carpeta un archivo nuevo, archivo3.txt.

Si le doy el comando git status el resultado será:

A computer code with green and white text

Description automatically generated

Como era de esperarse, ya que este último archivo3.txt aún no ha sido pasado a la siguiente fase, la fase stage.

Ahora lo paso a la siguiente fase con git add archivo3.txt y doy git status de nuevo

A computer code with green text

Description automatically generated

Ya estará en la siguiente fase.

Ahora si hago un git commit los archivos pasarán a la otra fase.

A computer code with numbers and symbols

Description automatically generated

Posteriormente creo otro archivo4.txt y modifico el archivo3.txt y doy un git status:

A screenshot of a computer code

Description automatically generated

Es claro que el archivo4.txt es no trackeado, pero el archivo3.txt si lo es (significa que git ya tiene un registro de este archivo dado que ya se hizo un commit del archivo)

De nuevo, con git clean sólo se podrán eliminar los archivos no trackeados.

El comando git clean es un comando **peligroso**. Ya que en proyectos reales hay muchos archivos importantes que a propósito no se agregan a git.

Antes de usar git clean -f es mejor hacer antes git clean -n



Con esto, nos estaría primero preguntando qué archivos serán los que se removerán, en el caso que se hiciera un git clean -f.

Para este caso sólo sería el archivo4.txt, porque como ya se dijo ese comando sólo elimina los archivos no trackeados.

Así, si doy un git clean -f



Y si reviso la carpeta, ya no aparece el archivo4.txt

Si en este momento doy un git status aparece lo siguiente:

A screenshot of a computer error

Description automatically generated

Pero si ahora doy git checkout archivo3.txt me reestablece dicho archivo a antes del cambio que realicé.

Dada la confusión que puede generar todo esto desde la terminal, el autor recomienda hacerlo mejor desde la interfaz gráfica de visual studio code.

**Veamos como echar hacía atrás documentos en la fase 2, staging area**

El comando ahora sería git reset, y lo que hace es resetear o reiniciar el HEAD a un estado específico.

Doy un git status

A screenshot of a computer error

Description automatically generated

Vemos que todo está limpio

Ahora agrego un cambio en un archivo, por ejemplo en archivo.txt

Ahora lo subimos a la siguiente fase con git add archivo.txt

Ahora hacemos un git status

A screen shot of a computer code

Description automatically generated

Vemos que se ha pasado el archivo al staging

Ahora para dar el archivo una fase atrás hacemos:

git reset HEAD archivo.txt

y si de nuevo damos git status

A close up of a message

Description automatically generated

De nuevo vuelve a la etapa 1 (working directory). Recuerde la convención de colores, rojo en working directory y verde en staging área.

**Veamos como echar hacía atrás documentos en la fase 3, local repository (es decir revertir commit que hemos hecho)**

git revert

El anterior comando revierte todos los cambios que se hicieron en un commit en específico y agregar un nuevo commit en la historia de git con los cambios deshechos.

Veamos un ejemplo.

Voy agragando líneas al archivo.txt, seguido del git add archivo.txt y seguido git commit -m"agragando linea1" con el comentarios del número de línea.

Ahora veo el historial del commit

git log

A screenshot of a computer code

Description automatically generated

Ahora voy a hacer git revert y utilizamos el commit correspondiente a cuando agregamos al linea3 (flecha roja)