GIT CLEAN

git checkout / git clean / git revert / git reset / git rm

En esta sección, nos centraremos en un debate pormenorizado sobre el comando git clean. En cierto modo, Git clean es un comando para "deshacer". Git clean se puede considerar como complementario a otros comandos, por ejemplo, git reset y git checkout. Mientras que otros comandos actúan en archivos añadidos previamente al índice de seguimiento de Git, el comando git clean actúa en archivos sin seguimiento. Los archivos sin seguimiento son aquellos que se han creado en tu directorio de trabajo, pero aún no se han añadido al índice de seguimiento del repositorio a través del comando git add. Para demostrar mejor la diferencia entre archivos en seguimiento y sin seguimiento, consulta el siguiente ejemplo de línea de comandos:

$ mkdir git_clean_test $ cd git_clean_test/ $ git init . Initialized empty Git repository in /Users/kev/code/git_clean_test/.git/ $ echo "tracked" > ./tracked_file $ git add ./tracked_file $ echo "untracked" > ./untracked_file $ mkdir ./untracked_dir && touch ./untracked_dir/file $ git status On branch master Initial commit Changes to be committed: (use "git rm --cached ..." to unstage) new file: tracked_file Untracked files: (use "git add ..." to include in what will be committed) untracked_dir/ untracked_file

En el ejemplo, se crea un nuevo repositorio de Git en el directorio git_clean_test. El siguiente paso es crear un archivo tracked_file que se añade al índice de Git. Además, se crea un archivo untracked_file y un directorio untracked_dir. En el ejemplo se invoca el comando git status que muestra el resultado que indica el estado interno de Git de los cambios en seguimiento y sin seguimiento. Con el repositorio en este estado, podemos ejecutar el comando git clean para demostrar su uso previsto.

$ git clean fatal: clean.requireForce defaults to true and neither -i, -n, nor -f given; refusing to clean

En este punto, la ejecución del comando git clean predeterminado puede producir un error irrecuperable. En el ejemplo anterior puedes ver el aspecto que puede tener. De forma predeterminada, la configuración global de Git obliga a usar la opción "force" con el comando git clean para que sea efectivo. Se trata de un importante mecanismo de seguridad. Una vez que se ejecuta el comando git clean, no se puede deshacer. Cuando se ejecuta por completo, git clean realizará una eliminación permanente del sistema de archivos, algo parecido a lo que ocurre al ejecutar la utilidad "rm" de la línea de comandos. Asegúrate de que quieres eliminar los archivos sin seguimiento antes de ejecutar el comando.

OPCIONES Y USO HABITUALES

Una vez proporcionada la anterior explicación sobre los comportamientos y avisos predeterminados de git clean, el contenido que presentamos a continuación expone varios casos de uso de git clean y las opciones de línea de comandos correspondientes que son obligatorias para su funcionamiento.

-n

La opción -n realizará un simulacro de git clean. Podrás ver los archivos que se van a eliminar sin que se eliminen realmente. Se recomienda realizar siempre un simulacro de git clean antes de llevar a cabo esta acción. Podemos demostrar el uso de esta opción en el repositorio de demostración que hemos creado anteriormente.

$ git clean -n Would remove untracked_file

El resultado nos indica que untracked_file se eliminará al ejecutar el comando git clean. Ten en cuenta que untracked_dir no se notifica en este resultado. De manera predeterminada, el comando git clean no actuará de forma recursiva en los directorios. Se trata de otro mecanismo de seguridad para evitar la eliminación accidental de estos elementos de forma permanente.

-f or –force

La opción "force" inicia la eliminación real de los archivos sin seguimiento del directorio actual. Es obligatorio el uso de esta opción siempre que la opción de configuración clean.requireForce no se haya configurado como "false". Los archivos o carpetas sin seguimiento especificados como .gitignore no se eliminarán. Vamos a realizar ahora una demostración real del comando git clean en nuestro repositorio de ejemplo.

$ git clean -f Removing untracked_file

El comando dará como resultado los archivos que se han eliminado. Observa cómo en este ejemplo se ha eliminado untracked_file. Al ejecutar el comando git status en este punto o aplicar ls veremos que untracked_file se ha eliminado y no se encuentra en ninguna parte. El comando git clean -f actuará de forma predeterminada en todos los archivos sin seguimiento del directorio actual. Además, se puede usar un valor con la opción -f para eliminar un archivo concreto.

git clean -f -d include directories

La opción -d le indica a git clean que también quieres eliminar los directorios sin seguimiento, ya que este comando ignora los directorios de forma predeterminada. Podemos añadir la opción -d a nuestros ejemplos anteriores:

$ git clean -dn Would remove untracked_dir/ $ git clean -df Removing untracked_dir/

En este caso, hemos realizado un simulacro con la combinación dn que da como resultado que untracked_dir está listo para su eliminación. A continuación, realizamos una limpieza obligatoria y obtenemos como resultado la eliminación de ntracked_dir.

-x force removal of ignored files

Un patrón de lanzamiento de software habitual es tener un directorio de compilación o distribución que no esté confirmado en el índice de seguimiento de repositorios. El directorio de compilación contendrá artefactos de compilación efímeros que se generan a partir del código fuente confirmado. Este directorio de compilación se suele añadir al archivo .gitignore de los repositorios. Puede ser conveniente también limpiar este directorio con otros archivos sin seguimiento. La opción -x indica a git clean que incluya también los archivos ignorados. Al igual que ocurría con las anteriores invocaciones de git clean, se recomienda realizar un simulacro antes de la eliminación final. La opción -x actuará en los archivos ignorados, no solo en las compilaciones de proyectos específicos. Esta opción podría dar como resultado no deseado archivos de configuración ./.idea IDE.

git clean -xf

Al igual que sucede con la opción -d, -x puede usarse y combinarse con otras opciones. En este ejemplo, se muestra una combinación con -f que eliminará los archivos sin seguimiento del directorio actual, así como los archivos que Git suele ignorar.

MODO INTERACTIVO O COMANDO "GIT CLEAN" INTERACTIVO

Además de la ejecución de la línea de comandos ad hoc que hemos demostrado hasta el momento, git clean cuenta con un modo "interactivo" que se puede iniciar mediante el uso de la opción -i. Vamos a recordar la reposición de ejemplo de la introducción de este documento. En esa fase inicial, empezaremos con una sesión de limpieza interactiva.

$ git clean -di Would remove the following items: untracked_dir/ untracked_file *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now>

Hemos empezado la sesión interactiva con la opción -d que actuará también en nuestro untracked_dir. El modo interactivo mostrará el mensaje What now> que solicita un comando para aplicarlo a los archivos sin seguimiento. Los comandos en sí son bastante explicativos. Echaremos un vistazo a cada uno de ellos en orden aleatorio empezando por el comando 6: help. Al seleccionar el comando 6 explicaremos también los demás comandos:

What now> 6 clean - start cleaning filter by pattern - exclude items from deletion select by numbers - select items to be deleted by numbers ask each - confirm each deletion (like "rm -i") quit - stop cleaning help - this screen ? - help for prompt selection

5: quit

Comando sencillo para salir de la sesión interactiva.

1: clean

Comando para eliminar los elementos indicados. Si se ejecutara el comando 1: clean en este punto, se eliminaría untracked_dir/ untracked_file .

4: ask each

Procesará una iteración en cada uno de los archivos sin seguimiento y mostrará un mensaje de Y/N para la eliminación. Tendrá el siguiente aspecto:

*** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now> 4 Remove untracked_dir/ [y/N]? N Remove untracked_file [y/N]? N

2: filter by pattern

Mostrará un mensaje adicional que toma la entrada usada para filtrar la lista de archivos sin seguimiento.

Would remove the following items: untracked_dir/ untracked_file *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now> 2 untracked_dir/ untracked_file Input ignore patterns>> *_file untracked_dir/

En este punto, introducimos el patrón de caracteres comodín *_file que restringe la lista de archivos sin seguimiento únicamente a untracked_dir.

3: select by numbers

De forma parecida al comando 2, el comando 3 trabaja para perfeccionar la lista de nombres de archivos sin seguimiento. La sesión interactiva solicitará números que correspondan a un nombre de archivo sin seguimiento.

Se eliminarían los siguientes elementos: untracked_dir/ untracked_file *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now> 3 1: untracked_dir/ 2: untracked_file Select items to delete>> 2 1: untracked_dir/ * 2: untracked_file Select items to delete>> Would remove the following item: untracked_file *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help

RESUMEN

En resumen, git clean es un método adecuado para eliminar los archivos sin seguimiento en un directorio de trabajo del repositorio. Los archivos sin seguimiento son aquellos que se encuentran en el directorio del repositorio, pero que no se han añadido al índice del repositorio con git add. De manera general, el efecto de git clean se puede lograr mediante el uso de git status y de las herramientas de eliminación nativas de los sistemas. Git clean se puede usar junto con git reset para deshacer por completo las incorporaciones y confirmaciones en un repositorio.