



366 pts ▾



Menú

>  [Curso Profesional de Git y GitHub](#)

Artículo **Git reset vs. Git rm**

9  **John Freddy Vega** ⏲ 16 de Septiembre de 2019

10

11 [Méjico: @juandc](#)

12 **git reset y git rm** son comandos con utilidades muy diferentes, pero aún así se confunden muy fácilmente.

13

git rm

14

15 **Este comando nos ayuda a eliminar archivos de Git sin eliminar su historial del sistema de versiones.** Esto quiere decir que si necesitamos recuperar el archivo 16 solo debemos “viajar en el tiempo” y recuperar el último commit antes de borrar el archivo en cuestión.

17

18 Recuerda que `git rm` no puede usarse así nomás. Debemos usar uno de los flags para indicarle a Git cómo eliminar los archivos que ya no necesitamos en la última versión del proyecto:

- `git rm --cached`: Elimina los archivos del área de Staging y del próximo commit pero los mantiene en nuestro disco duro.
- `git rm --force`: Elimina los archivos de Git y del disco duro. Git siempre guarda todo, por lo que podemos acceder al registro de la existencia de los archivos, de modo que podremos recuperarlos si es necesario (pero debemos usar comandos más avanzados).



Flujo de trabajo básico con u...

Este comando nos ayuda a volver en el tiempo. Pero no como `git checkout` que nos deja ir, mirar, pasear y volver. Con `git reset` volvemos al pasado sin la posibilidad de volver al futuro. Borramos la historia y la debemos sobreescribir. No hay vuelta atrás.

Este comando es **muy peligroso** y debemos usarlo solo en caso de emergencia. Recuerda que debemos usar alguna de estas dos opciones:

Hay dos formas de usar `git reset`: con el argumento `--hard`, borrando toda la información que tengamos en el área de staging (y perdiendo todo para siempre). O, un poco más seguro, con el argumento `--soft`, que mantiene allí los archivos del área de staging para que podamos aplicar nuestros últimos cambios pero desde un commit anterior.

- `git reset --soft`: Borramos todo el historial y los registros de Git pero guardamos los cambios que tengamos en Staging, así podemos aplicar las últimas actualizaciones a un nuevo commit.
- `git reset --hard`: Borra todo. Todo todito, absolutamente todo. Toda la información de los commits y del área de staging se borra del historial.

¡Pero todavía falta algo!

- `git reset HEAD`: Este es el comando para sacar archivos del área de Staging. No para borrarlos ni nada de eso, solo para que los últimos cambios de estos archivos no se envíen al último commit, a menos que cambiemos de opinión y los incluyamos de nuevo en staging con `git add`, por supuesto.

¿Por qué esto es importante?

Imagina el siguiente caso:



Flujo de trabajo básico con u...

Hacemos cambios en los archivos de un proyecto para una nueva actualización. Todos los archivos con cambios se mueven al área de staging con el comando `git add`. Pero te das cuenta de que uno de esos archivos no está listo todavía. Actualizaste el archivo pero ese cambio no debe ir en el próximo commit por ahora.

¿Qué podemos hacer?

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.

¡Al usar `git rm` lo que haremos será eliminar este archivo completamente de git! Todavía tendremos el historial de cambios de este archivo, con la eliminación del archivo como su última actualización. Recuerda que en este caso no buscábamos eliminar un archivo, solo dejarlo como estaba y actualizarlo después, no en este commit.

En cambio, si usamos `git reset HEAD`, lo único que haremos será mover estos cambios de Staging a Unstaged. Seguiremos teniendo los últimos cambios del archivo, el repositorio mantendrá el archivo (no con sus últimos cambios pero sí con los últimos en los que hicimos commit) y no habremos perdido nada.

Conclusión: Lo mejor que puedes hacer para salvar tu puesto y evitar un incendio en tu trabajo es conocer muy bien la diferencia y los riesgos de todos los comandos de Git.



Escribe aquí tu comentario

+ 2 R



David Solorzano Estudiante · hace 11 horas



Flujo de trabajo básico con u...



 Erika Gabriela Villanueva Perez de Leon Estudiante • hace 13 horas

Espero no tener que usarlos, pero es bueno saberlo, ya que anteriormente prefería hacer un nuevo repositorio desde 0 😅 jeje...



 Stiven Prado Estudiante • hace 15 horas

gran explicacion y buen consejo, casi siempre hacer un delete o una reversion sin las debidas precauciones genera muchos inconvenientes y en la vida real hasta el trabajo se pone un juego por dos lineas de codigo



 Fredy Mendoza Vargas Estudiante • ayer

Buen dato



 ÁNGEL URIEL VELASCO MEJÍA Estudiante • anteayer

Con mucho dolor recuerdo el where en un delete from que olvidé en la prepa. Se me hace difícil imaginarme en mi carrera profesional sin haber cometido un error similar con un proyecto realmente importante, pero haré todo lo posible por que eso no pase.



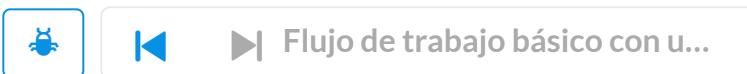
 Gonzalo Santiago Naya Estudiante • anteayer

Alguien podría aclararme en este caso antes de leer la aclaración de freddy yo habría usado git rm --cached ya que solo lo estaríamos quitando del staging para el commit, cual es el error ahí? ya entendí que debo usar el reset head pero sigo con dudas.



 Simon Correa Henao Estudiante • anteayer

Excelente lectura. Me encantó el ejemplo practico del final, además de dejar todo muy claro y de





Antonio Demarco Bonino Estudiante · hace 3 días

Esta es de esas clases para seguir leyendo siempre.



0



Paty Patricia Estudiante · hace 3 días

Buen aporte. La** Conclusión**, ha hecho en mi un choque de realidad, y hace que lo aprenda más rápido. Gracias



0



Rafael Miranda Almendaris Estudiante · hace 3 días

Se necesita saber plenamente el uso de los comando, una linea de comando mal colocada nos costara el puesto en el trabajo.



0



Cristian Gastelbondo Estudiante · hace 4 días

Importante tener claro la funcionalidad de los comandos y el contexto en la cual la vamos a aplicar, esto con el fin de evitar dificultades.



0



Giovanni Osorio Estudiante · hace 4 días

quiere decir que si uso git reset HEAD saco TODOS los archivos de staging, vuelvo al ultimo commit.

PERO los archivos que saque de staging siguen estando en el disco duro.

Y volviendo al ejemplo del texto, entonces tengo que añadir cada archivo de nuevo con git add EXCEPTO EL QUE NO ESTA LISTO y despues si hacer el commit

Eso fue lo que entendi

No se si estoy equivocado? someone who back me up here?



0



Sergio Naranjo Morales · hace 4 días

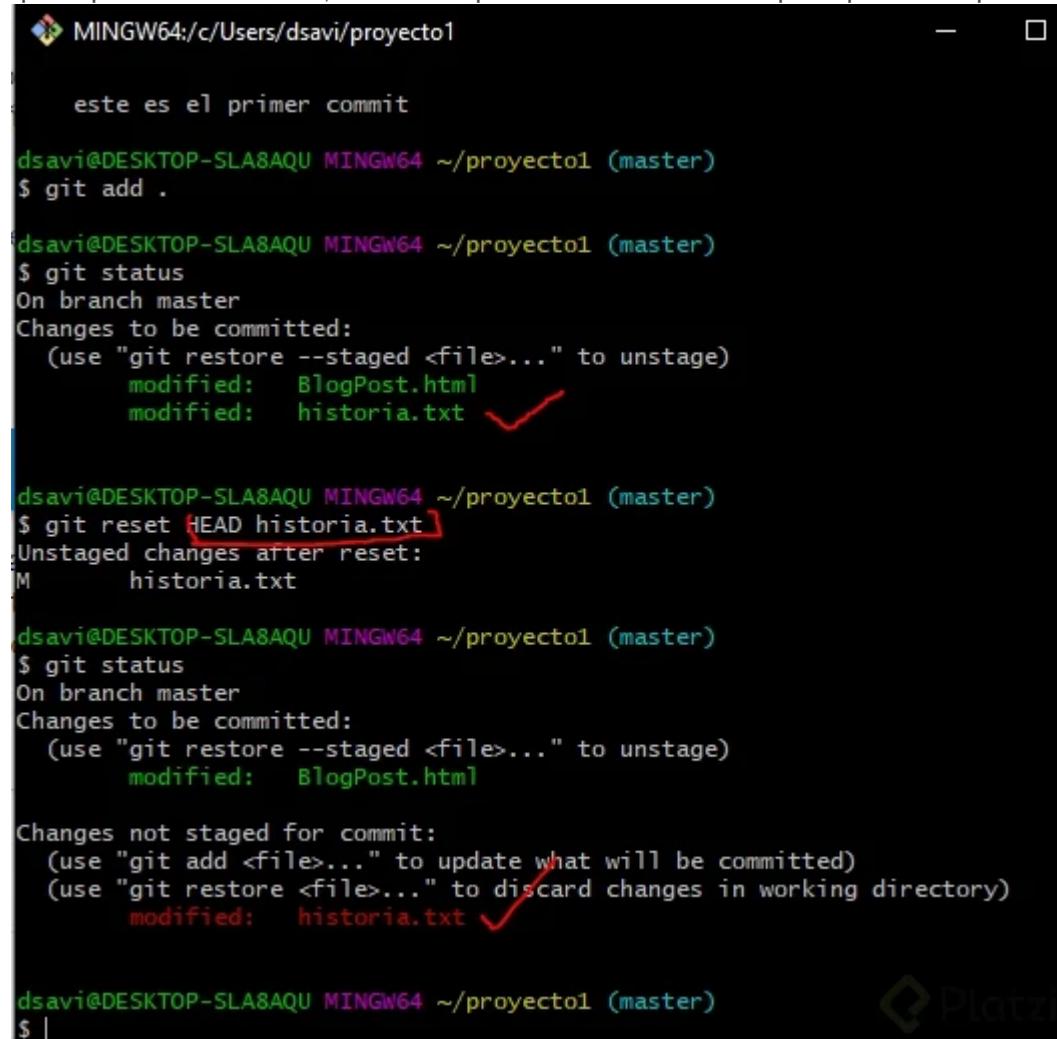
Si, yo entendí lo mismo, con el head se saca todo y se toca agregar add y los archivos específicos menos el que no está listo



Flujo de trabajo básico con u...

David Steven Avila Vela · hace 3 días

No, con el git reset HEAD nombreArchivo ... solo va a sacar del staging el archivo q no esta listo para mandar al commit, el resto de archivos van a quedar en la ram, mientras agregas con git add, nuevamente el archivo que sacaste con el git reset HEAD. no se si me hago entender... Te recomiendo que lo pruebes en consola, te envio un pantallazo de mi consola para q se vea lo que trato de explicar.



```
MINGW64:/c/Users/dsavi/proyecto1
este es el primer commit

dsavi@DESKTOP-SLA8AQU MINGW64 ~/proyecto1 (master)
$ git add .

dsavi@DESKTOP-SLA8AQU MINGW64 ~/proyecto1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   BlogPost.html
    modified:   historia.txt ✓

dsavi@DESKTOP-SLA8AQU MINGW64 ~/proyecto1 (master)
$ git reset HEAD historia.txt
Unstaged changes after reset:
M       historia.txt

dsavi@DESKTOP-SLA8AQU MINGW64 ~/proyecto1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   BlogPost.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   historia.txt ✓

dsavi@DESKTOP-SLA8AQU MINGW64 ~/proyecto1 (master)
$ |
```

7

[Ver más respuestas](#)



sebastian felipe herrera sanchez Estudiante · hace 4 días

No entiendo muy bien la diferencia entre:

git restore --staged <file_name>

git reset HEAD <file_name>



0



Flujo de trabajo básico con u...

¿Con git reset HEAD todos los archivos salen del área de staging, pero hay alguna forma de sacar solo un archivo de esa área?

2 

 Adriana Selena Tito Ilasaca · hace 4 días

```
git resetHEAD <fileName>
```

Si quieres sacar el archivo history.txt de staging

```
git resetHEAD history.txt
```

6

 Adriana Selena Tito Ilasaca · hace 4 días

O como nos recomienda git cuando adicionamos archivos a staging

```
Changes to be committed:  
(use "git rm --cached <file>..." to unstage)  
  new file: historia.txt
```

1

[Ver más respuestas](#)



Fernando Juarez Baca Estudiante · hace 5 días

Una explicación muy útil, al leerlo piensas que es más sencillo pero cuando lo pones en práctica muchas veces es cuando realmente entiendes la función y riesgo de cada comando.

¡A seguir practicando! 😊

0 

 Victor Andres Castillo Zambrano Estudiante · hace 5 días

muy buena la informacion

1 



Flujo de trabajo básico con u...

0

M_Romero Estudiante · hace 5 días

¿Qué son los “registros”? ¿Es lo mismo que las “versiones”?

0

Orlando Jose Altamiranda Piñango Estudiante · hace 6 días

Está explicado de manera muy sencilla, aún así no lo entendía.
La mejor manera de entenderlo fue probar y repetir cada comando con archivos varias veces.
Sinceramente, es complejo en términos de utilidad, pero simple en términos de funciones y me remito fielmente a lo que dice la conclusión: "**Lo mejor para evitar incendios es conocer muy bien la diferencia, utilidad y riesgos de cada comando**"

No es algo que se logre a las primeras, así que avanzo, pues solo es ejecutar mucha practica.
Gran articulo, muchas gracias 😊

2

René Lara Estudiante · hace 6 días

Guao, muy buena explicación

1

Guillermo Camacho Estudiante · hace 6 días

Muchas gracias.

0

Freivys Antonio Paredes Briceño Estudiante · hace 6 días

Excelente tema, importante conocer los riesgos de utilizar git rm y git reset. ↗

0

Nino jesus espinoza garcia Estudiante · hace 7 días

que gran explicacion



Flujo de trabajo básico con u...

Muy bien explicado gracias!

0

Johan Huaman Castañeda Estudiante • hace 8 días

Interesante!

2

José Álvarez Estudiante • hace 8 días

Teniendo git reset head no entiendo en que caso me serviria usar git rm --soft
Cual seria su aplicacion?

2

Luis Homez • hace 7 días

Hola, creo que el comando al que te refieres es git reset --soft según la prueba que hice este solo borro los registros de git hasta la versión que le indique en el reset, pero el archivo se conserva con los últimos cambios que hice y que no he agregado, los cuales aun puedo agregar al área de staging y luego hacer el commit, supongo que es una opción para limpiar el historial de commits. Por otra parte git reset HEAD sirve para sacar todos los archivos que habíamos agregado al área de staging luego de haber dado git add . que es el paso anterior al commit.

2

Luis Homez • hace 7 días

Agregando a lo anterior que si noto similitud entre git reset head y git rm --cached, solo que con head no hay que indicar el nombre del archivo que deseamos sacar del área de staging, porque saca todo lo que este esta área mientras que con git rm "nombre_archivo" --cached si debemos indicar el nombre del archivo, es decir, solo podemos sacar un archivo del área de staging a la vez.

2

[Ver más respuestas](#)

David Antonio Suárez Arévalo Estudiante • hace 9 días

¿ Que comando permite recuperar archivos eliminados con el comando git rm --force ?

1



Flujo de trabajo básico con u...

puedes recuperar alguna versión de ese archivo borrado siempre y cuando ya exista por lo menos un commit de el.
primero ejecuta

```
git log
```

copia el ID del commit que tenia el archivo y luego ya sea que uses

- para viajar en el tiempo con la posibilidad de retorno

```
git checkout id_commit archivo.txt
```

- para viajar al pasado sin la posibilidad de retorno

```
git reset id_commit --hard
```

2



Diego Buesaquillo Estudiante • hace 9 días

Muy importante la aclaración!



1



Brandon Argel Verdeja Dominguez Estudiante • hace 10 días

Hola, en git recomienda usar git restore --staged "file" para quitar a un archivo del Staging Area
Y me llegó una duda, cual es la diferencia entre \$git restore --staged "file" y \$git reset HEAD
A lo que veo hacen lo mismo, pero por la sintaxis pienso que el git reset HEAD resetea el HEAD
(version actual) con lo que también se eliminan los archivos de Staging Area Y \$git restore --staged
"file" nomás quita a dicho archivo de Staging Area...

¿Si es así?



6



René Lara • hace 6 días

Interesante pregunta, pendiente de la respuesta

0



Flujo de trabajo básico con u...

Muy buena la aclaración

1

Mario Enrique Ascencio Garcia Estudiante • hace 11 días

Gracias, excelente informacion.

0

Luis Daniel Hernández Guevara Estudiante • hace 11 días

La aportación me ha sido de gran utilidad , en el inicio puede ser confuso pero en este apartado queda claro . Gracias!

2

Cristian Camilo Hernández Ramírez Estudiante • hace 15 días

Buenas, tengo una pregunta ¿Alguien sabe la diferencia entre `$git rm --cached` y `$git restore --stage`?

Veo que el primero me pasa de **Stage a Untracked** y el segundo me pasa de **Stage a Unstaged** pero no le veo ninguna utilidad, Gracias.

5

Cristian Camilo Hernández Ramírez • hace 15 días

Veo que al parecer `$git restore --stage` es lo mismo que `$git reset HEAD`, si es así la duda está resuelta.

0

Sergio Garnica • hace 12 días

Por lo que he visto al practicar con la terminal y fijándome en los insights o comentarios que la misma me da, pude notar que ambos funcionan para lo siguiente:

Hacen que el archivo que ya ha pasado a **staged** con `git add`, pase a **unstaged** para que lo puedas modificar y después volverlo a añadir y hacer un commit. Sin embargo la única diferencia que pude notar al hacer el ejercicio es el siguiente:

`$git rm --cached`

Funciona para los archivos que **aún no se hayan añadido al repositorio** para que pasen de 'staged' a 'unstaged'

`$git restore --staged`

Funciona para los archivos que **ya se han añadido al repositorio pero han sido modificados**. Esto quiere decir que si modificas tu archivo y lo añades con `git add` pero al final te arrepientes y lo quieres



Flujo de trabajo básico con u...

observar al tener la misma duda.
Espero te ayude a resolver tu duda tambien
Saludos!

2

 Brandon Argel Verdeja Dominguez • hace 10 días

A lo que ví si usas el `$git rm --cached` queda **Untracked** el archivo como tu dices, y en ese estado probé y no te deja seguir trabajando con ese archivo hasta que lo agregues de nuevo con `git add`, entonces supongo que puedes "**Untrackear**" un archivo con el que de mientras no quieras trabajar para no equivocarte y modificarlo por error

O tal vez algo relacionado con `$pull` ya que estes vinculado con el repositorio de GitHub, pero todavía no llego ahí, puede que solo lo averigüemos con la práctica jajajaja

Saludos y si encuentro algo regresaré aquí 😊👍

2

 Cristian Camilo Hernández Ramírez • hace 9 días

Sergio Muchas gracias por tu comentario, ya corroboré todo lo que me dices y solo tengo un apunte. Efectivamente `$git rm --cached` se usa para archivos en **staging** que aún no se han añadido al repositorio pero siempre te pasan de '**staged**' a '**unstaged**'. Anexo ejemplo:

```
$ touch archivo.txt//Creas el archivo
$ git add archivo.txt//lo pones en staging
$ git rm --cached archivo.txt//Haces el rm --cached
rm 'archivo.txt'
$ git status // El archivo quedó 'Untracked'
On branch tests
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    archivo.txt

nothing added to commit but untracked files present (use "git add" to
track)``
```

2

 Cristian Camilo Hernández Ramírez • hace 9 días

Escribí mal jajaja, te pasa de '**Staged**' a '**Untracked**'



Flujo de trabajo básico con u...



Luis Miguel Daraviña Henao Estudiante • hace 18 días

es posible volver a una versión pasada, después ejecutar git reset --hard



1



Omar Medina • hace 17 días

Según el post, se borra todo para siempre, entiendo que es porque también se borra lo de staying y los commits hechos, por lo que no hay un ancla de donde agarrarse para recuperar los archivos.

2



cesaralmeidareyes • hace 17 días

La última versión, la actual, será la versión a la que has vuelto. Así que si tienes versiones anteriores sí es posible volver a una versión pasada.

0



Juan Miguel Garcia Carballo • hace 13 días

capaz con git restore "nombreDelArchivo.Extension", sin las comillas, puedes, pero realmente ni idea.
No lo probe todavía

1

[Ver más respuestas](#)



Andres Felipe Cogollo Hincapie Estudiante • hace 19 días

Información complementaria que aclara más la utilidad de los comandos



2



Sebastian Páez Estudiante • hace 19 días

Excelente recurso, estaba un poco confundido con la funcionalidad de ambos comandos pero ya todo está aclarado, Gracias



0



Levis Manuel Luna Arrieta Estudiante • hace 20 días



Flujo de trabajo básico con u...

[2](#) [Share](#) Víctor Hugo Torres Fierro • hace 19 días

Hola Levis,
mientras estas en tu carpeta de home (si no estas usa cd home) una vez ahí, escribe:
`rm -rf .git`
Después dale enter. Git debería cesar de hacer el tracking de tu carpeta home. No te preocunes, nada se borra excepto los archivos ocultos que git agregó para poder realizar el init. Saludos.

3

 Levis Manuel Luna Arrieta • hace 19 días

Hola VTorres, muchas gracias, ahí me funcionó.

0

[Ver más respuestas](#) Christian Latorre Estudiante • hace 22 días

viendo el curso a 1.75x, despues se hace la practica, tenia abandonado el curso por que no entendia nada, despues de 6 meses lo retomo y ya se de que es lo que habla, esperemos ver que no ha dicho como se conecta el repo a la cuenta de git personal desde la consola

[3](#) [Share](#) Ciro Edgardo Camey Valdez • hace 15 días

Yo estoy esperando esa parte, conectar el respo y que ya no sea local

0

 Alejandro Sotelo Salazar Estudiante • hace 23 días

Entonces cual es la diferencia entre `git rm --cached` y `git reset HEAD`?

[3](#) [Share](#) Hugo Rojas Martinez • hace 23 días

Oie, pero que buena pregunta, tengo entendido por lo dicho hasta ahora en el curso que hacen lo mismo (ambos comandos), sin embargo, si se quiere eliminar un archivo/commit del área de staging es

   Flujo de trabajo básico con u...

ya que los comandos que implican

```
git rm
```

esta mas enfocado a una eliminación directa de archivos que a “sacar archivos del área de Staging” Entonces evitas el riesgo de eliminar algún archivo de manera física por un error cometido habria que usar

```
git rm --cached
```

Seria mas bien una especie de debate a partir de cual es mas adecuado como buena practica, así lo entiendo yo.

2



Alejandro Sotelo Salazar • hace 22 días

Platicando con un amigo me comentó que usar reset tampoco es buena práctica, porque de hecho git te dice cómo hacer unstaged:

```
git restore --staged <file>
```

3



eltrovador09 • hace 18 días

Es cierto, git te recomienda git restore --staged <file>, pero entonces me surge una duda, ¿es malo usar los comando git reset y git rm para hacer unstaged?

0



Brandon Argel Verdeja Dominguez • hace 10 días

¡Hola @Alejandro!

La diferencia entre usar \$ git rm --cached y \$ git reset head para eliminar un archivo de Staging: Con \$ git rm --cached deja los archivos sin ser rastreados (untracked), los elimina completamente de git.

Con \$ git reset head el archivo sigue siendo trackeado, solo se eliminan de Staging los ultimos cambios.

-Platzi

2

[Ver más respuestas](#)



Aleiandro Cuello Maure Estudiante • hace 23 días



Flujo de trabajo básico con u...

Pero en el comando

```
git reset HEAD
```

se usó para borrar los archivos que están ubicados en la ram (área de staging). Entonces, ¿HEAD no solo apunta al ultimo commit si no que tambien a los archivos que tenemos en el area de staging?

3



Edison Manrique · hace 20 días

Es porque tambien es un puntero a la rama actual, digamos que en Git sólo podemos tener abierta una rama a la vez, por ello se originó este término.

2



Jared Bustamante Quiñonez Estudiante · hace 23 días

Existe algun comando de git que pueda borrar del disco duro los Untracked files ?
Me ha pasado que una pagina genere archivos (por algun proceso interno) y necesito hacer pull pero no puedo porque existen cambios en el proyecto... entonces lo que tengo que hacer es un git reset HEAD para que ignore todos los cambios, pero los archivos siguen allí y cada vez que hago pull tengo que hacer lo mismo. No se si me explique ... en mi cabecita suena bien jejejej

0



Adrian Mansilla Salas · hace 20 días

Lo primero que tienes que hacer es poner los archivos que no quieras que se suban al repositorio en el .gitignore, entonces no se trapearan mas, de ahí ya puedes jugar con los borrados.
Saludos.

0



Humberto Bernal Mendivil Estudiante · hace 26 días

¿Estos comandos hacen lo mismo?

```
git resetHEAD  
git checkout -- .
```



Flujo de trabajo básico con u...



Cristobal Nyram • hace 24 días

No amigo . no hacen lo mismo...

Por que con :

```
git reset HEAD nombreDelArchivoQueQuieresSacarDeStanged.txt
```

Lo que haces es sacar de stangin lo cambios que aun no quieres mandar al commit como se muestra en el siguiente ejemplo, supongamos que hiciste

```
git add .
```

Y has mandado a stanged los siguientes archivos

0



Javier Romero • hace 23 días

No, ten en cuenta que si haces git reset HEAD es como cancelar la ultima operacion de git add que hayas hecho (por eso pasa de staging a unstaged) todo esto sin hacer commit aún, pero git chekout lo que hace es volver a una version del archivo que ya has hecho commit,

2

[Ver más respuestas](#)



CesarGuzman Estudiante • hace 26 días

Creo que estos conceptos merecen videos para hacer ejercicios sobre todo para corregir errores, yo continé en el curso y me tire el repositorio por que hice cambios donde no era, está bien que aprendiendo se puede borrar y volver a empezar pero en la vida real no se puede hacer eso, y estos son los comandos que te salvan la vida.



2



Xiomara Elizabeth Gutierrez Henriquez Estudiante • hace 28 días

Excelente resumen! Importantísimo conocer muy bien la función de cada comando
Muchas gracias!



1



Kevin Espinosa Caballero Estudiante • hace 28 días

Interesante, los he utilizado en pruebas sin comprender al 100% sus funcionalidades.



Flujo de trabajo básico con u...

 Jhon Anderson Alonso Hernandez Estudiante • hace 29 días

Super interesante el manejo del reset, toca explorarlo un poco mas

 2 

 Stalin Badillo Estudiante • hace 29 días

Muy útil la información

 0 

 Jhon Alexander Mondragon Trinidad Estudiante • el mes pasado

soy el único que lo lee con la voz de freddy jajaja

 3 

 Jonathan Blanco Hernandez • hace 29 días

jajaja no eres el unico...

3

 Patricio Zavala • hace 29 días

sobre todo la parte del " es muy peligroso" y la del "todo, todito, todo" jajajajaja

2

 Jhon Alexander Mondragon Trinidad • hace 29 días

jajajajajaja

1

 Jesus Daniel Martinez Bautista • hace 29 días

jaja mas cuando dice todo todito

2

 Eduardo Angel Mex Herrera • hace 27 días



Flujo de trabajo básico con u...

[Ver más respuestas](#)

Andres Alvarez Becerra Estudiante • el mes pasado

Sigo sin entender

```
git resed HEAD <archivo.extencion>
```

y

```
git rm --cached <archivo.extencion>
```

Por qué en ambos casos el archivo queda **Untracked**. No entiendo a que se refieren con que el segundo comando elimina el archivo de git. Agradezco la explicación



0



dvsalcedo • el mes pasado

Las diferencias son:

- git reset HEAD lo sacas del área de Staging (Mantienes los cambios).
- git rm --cached le remueve el seguimiento (Es un archivo nuevo para git)

5



Andres Alvarez Becerra Estudiante • el mes pasado

Hola. ¿Qué es un flag?



0



dvsalcedo • el mes pasado

Es un parametro extra que agregas al comando que estas escribiendo.

2



Jose Antonio Sanchez • el mes pasado

En el caso de "git rm --cached" lo último (-cached) sería el flag



Flujo de trabajo básico con u...



Christopher Brian Guzmán Martínez Estudiante • el mes pasado

Muy buen aporte, complementé mis notas 😊



1



Carlos Nassif Trejo Garcia Estudiante • el mes pasado

Reset: Volver a una versión anterior

- Hard: Todo vuelve al estado anterior (borra todo del staging)
 - Soft: El staging sigue ahí, disponible para el próximo commit
 - HEAD: Sacar archivos del área de staging, se puede revertir con un git add .
- Rm:** Quitar un archivo, pero mantenerlo en el control de versiones
- Cached: Del staging y del próximo commit, pero los mantiene en el disco duro
 - Force: Del git y del disco duro. Sin embargo, se puede viajar en el tiempo y recuperarlo



11



Andrés Felipe Lopez gomez • hace 22 días

Excelente aporte

1



Juan Carlos Bonilla González • hace 16 días

Muy claro, buen aporte

0

[Ver más respuestas](#)



Jaime Mauricio Garrido Gomez Estudiante • el mes pasado

Excelente resumen, hasta leerlo fui que me di cuenta que estaba navegando en un checkout en un par de commit's previos y además de salirme "números raros" al final de mi rama, no podía ver muchos de los cambios que había hecho y era por la línea de experimentación que medio el checkout, hasta que vi que con git switch - podía volver a mi rama master.

En fin, creo que concluye muy bien el artículo, "Lo mejor que puedes hacer para salvar tu puesto y evitar un incendio en tu trabajo es conocer muy bien la diferencia y los riesgos de todos los comandos de Git." 😊



1



Flujo de trabajo básico con u...

0



Iván Arcos Estudiante • el mes pasado

Excelente resumen, la verdad tenía dudas y esto me las resolvió. Gracias.



2



Jeremy Quintero Rocha Estudiante • el mes pasado

Excelente resumen, creo que todo es más de práctica con casos reales.



0



Gabriel Ichcanziho Pérez Landa Estudiante • el mes pasado

Tengo una duda:

Supongamos que mi proyecto solo tiene un archivo txt.

Cuál sería la diferencia entre utilizar: git reset HEAD y git rm --cached archivo.txt

O a caso en ese caso son equivalentes? Gracias 😊



2



Edison Manrique • hace 20 días

Para nada, con:

```
$ git rm --cached file.txt
```

Lo estás sacando del monitoreo de Git, es decir, ya Git no sabrá qué es ese archivo ni qué con su vida, entrará en **untracked area**. Por otro lado, el

```
$ git resetHEAD
```

Sólo saca de Stage el cambio (no lo borra, sólo funcionara para sacarlo, en caso de que haya hecho falta algo). Entonces, sigue siendo conocido por Git, pues no se vuelve un archivo untracked como con rm, en lugar de eso, pasa a la **unstaged area**.

2



Flujo de trabajo básico con u...

Voy a hacer algunas pruebas para estar seguro pero creo que el texto es esclarecedor.

0

María Fernanda Barrero Rincón Estudiante · [el mes pasado](#)

Tengo una pregunta.

Yo he utilizado el comando `git reset .` (nótese el punto al final) justamente para sacar **todos** mis archivos del `staging` area y queden en mi working directory.

Por el momento, me ha funcionado bastante bien, no me ha borrado cambios ni archivos.

¿Estaría correcto usarlo de esa manera? Además, ¿el HEAD lo que hace es movernos **todos** los cambios del `staging` area o un archivo en específico? ¿Debo especificar mi archivo? Por ejemplo:
`git reset hola.txt`

1

Jeith S Carrillo A · [el mes pasado](#)

con el git reset asi lo que haces es volver a Head. en ese caso no hay problema si lo que quieres es quitar los cambios de staging

2

Bryan Ricardo Armas Loyaga Estudiante · [el mes pasado](#)

Gran aporte, con estas diferencias queda mas claro el panorama, y tener en cuenta la forma adecuada de usar los comandos.

0

Verónica Alcaraz Estudiante · [el mes pasado](#)

Excelente artículo, me aclaró muchas dudas, sin embargo, consulté y para quienes aún no les queda claro, esta definición puede dar muchas luces: `git rm --cached` elimina el file del índice. `git reset HEAD` restablece la versión de índice del file a su estado en la confirmación HEAD . Entonces, la diferencia es que el primero elimina el file, mientras que el segundo lo revierte a la última versión comprometida.

0

Pedro Daniel Rojas Corona Estudiante · [el mes pasado](#)

Tenía mis dudas, pero con este texto va se me aclaró más la mente.



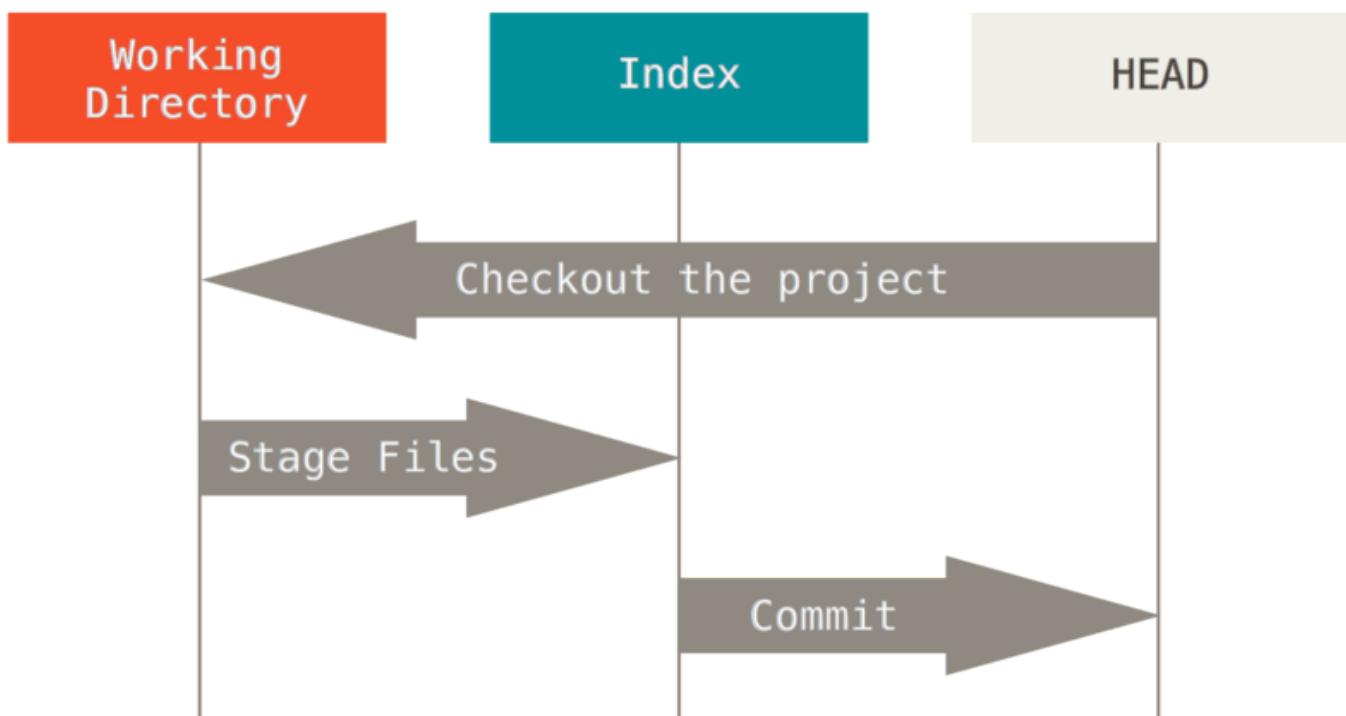
Flujo de trabajo básico con u...



Jhon Fabio Cardona Martinez Estudiante • el mes pasado

Considero que el articulo esta interesante y lo entiendo bien.

Me hubiese gustado que adjuntara capturas de pantalla o imágenes describiendo el articulo, para hacerlo mas atractivo.



2



David Santamaria Jimenez Estudiante • el mes pasado

Está confuso, pero creo le entendí



0



Gustavo Andres Niño Forero Estudiante • el mes pasado

Hasta ahora me a gustado la metodología de platzi, todo es concreto y practico



3



Ricardo Navarro Estudiante • el mes pasado

Hola a todos !!

una pregunta esta parte del curo que es la clase número 13, no hay video?



Flujo de trabajo básico con u...

me sale solamente explicación por texto.

The screenshot shows a course page from 'Curso Profesional de Git y GitHub'. The article title is 'Git reset vs. Git rm' by John Freddy Vega, published on September 16, 2019. The text discusses the difference between 'git rm' and 'git reset'. It explains that 'git rm' removes files from the Git history, while 'git reset' only removes them from the working directory. A note at the bottom says 'Recuerda que git rm no puede usarse así nomás. Debemos usar uno de los flags para indicarle' (Remember that git rm cannot be used like this. We must use one of the flags to indicate it). There are navigation icons for back, forward, and search.

Curso Profesional de Git y GitHub

Artículo Git reset vs. Git rm

John Freddy Vega 16 de Septiembre de 2019

Texto: @juandc

Git reset y git rm son comandos con utilidades muy diferentes, pero aún así se confunden muy fácilmente.

git rm

Este comando nos ayuda a eliminar archivos de Git sin eliminar su historial del sistema de versiones. Esto quiere decir que si necesitamos recuperar el archivo solo debemos "viajar en el tiempo" y recuperar el último commit antes de borrar el archivo en cuestión.

Recuerda que `git rm` no puede usarse así nomás. Debemos usar uno de los flags para indicarle

A comment from user Yotziri Paloma Pérez Ríos, posted a month ago, saying: 'Únicamente es el articulo, tómalo para hacer retro-alimentación :)' (Only the article, take it to make feedback :)).

Yotziri Paloma Pérez Ríos • el mes pasado

Únicamente es el articulo, tómalo para hacer retro-alimentación :')

2

A comment from user Ricardo Navarro, posted a month ago, saying: 'Yeah, muchas gracias 😊 !' (Yeah, many thanks 😊 !).

Ricardo Navarro • el mes pasado

Yeah, muchas gracias 😊 !

0

A comment from user David Santamaria Jimenez, posted a month ago, saying: 'Es parte del curso, aunque no tenga vídeo' (It's part of the course, even though it doesn't have a video).

David Santamaria Jimenez • el mes pasado

Es parte del curso, aunque no tenga vídeo

2

A comment from user katsumoto, posted a month ago.

katsumoto • el mes pasado



Flujo de trabajo básico con u...



Jaime Mauricio Garrido Gomez · el mes pasado

Correctirijillo,

Hay resúmenes o materiales en texto entre clases y clases (videos) para fortalecer, tomar nota o simplemente activar otras zonas del cerebro 😊

2

[Ver más respuestas](#)



Martín Eduardo Larrea Armijos Estudiante · [el mes pasado](#)

Yo usaria el alias para cambiar de git reset a git usameconciudadado. :v



5



Rosel Miguel Castillo Romero Estudiante · [el mes pasado](#)

Muy preciso la explicación



0



Andres Navarro Estudiante · [el mes pasado](#)

Llevo años usando git y me doy cuenta que no se nada de git jajaja



2



Gustavo Alonso Pacheco Mex Estudiante · [el mes pasado](#)

El curso sumado a la retroalimentación de la comunidad no se puede igualar, gracias a todos!



1



Leonardo Chuello Estudiante · [el mes pasado](#)

Excelente explicación



0



Juan Pablo Sora Ospina Estudiante · [el mes pasado](#)

Buenas noches tengo el siguiente problema tengo mi repositorio actualizado quiero borrar una



Flujo de trabajo básico con u...

```
Miguel@zeus MINGW64 /c/xampp2/htdocs/gia-dev/app/PHPWord (dev_phpWord)
$ git status
On branch dev_phpWord
nothing to commit, working tree clean

Miguel@zeus MINGW64 /c/xampp2/htdocs/gia-dev/app/PHPWord (dev_phpWord)
$ ls
composer.json  README.md

Miguel@zeus MINGW64 /c/xampp2/htdocs/gia-dev/app/PHPWord (dev_phpWord)
$ git rm --force composer.json
fatal: pathspec 'composer.json' did not match any files

Miguel@zeus MINGW64 /c/xampp2/htdocs/gia-dev/app/PHPWord (dev_phpWord)
$ git rm --force README.md
fatal: pathspec 'README.md' did not match any files

Miguel@zeus MINGW64 /c/xampp2/htdocs/gia-dev/app/PHPWord (dev_phpWord)
$ git rm --force README.md
fatal: pathspec 'README.md' did not match any files

Miguel@zeus MINGW64 /c/xampp2/htdocs/gia-dev/app/PHPWord (dev_phpWord)


Flujo de trabajo básico con u...

Si despues del `$ git rm --cached` vemos el status, podemos ver que efectivamente el archivo fue eliminado, y que no esta siendo rastreado.

```
david@LAPTOP-02Q7BFKO MINGW64 ~/Documents/Cursos/Git y GitHub/proyecto1 (master)
$ git rm --cached historia.txt
rm 'historia.txt'

david@LAPTOP-02Q7BFKO MINGW64 ~/Documents/Cursos/Git y GitHub/proyecto1 (master)
$ git status
On branch master
Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 deleted: historia.txt

Untracked files:
 (use "git add <file>..." to include in what will be committed)
 historia.txt
```

Con `$ git reset head` el archivo sigue trackeado, pero los ultimos cambios no han sido añadidos a Staging. Si vemos el status despues de ejecutar el

`$ git reset head` vemos que el archivo sigue siendo trackeado, no ha sido eliminado de Git, pero los ultimos cambios no han sido agregados a Staging.

```
david@LAPTOP-02Q7BFKO MINGW64 ~/Documents/Cursos/Git y GitHub/proyecto1 (master)
$ git reset head
Unstaged changes after reset:
M historia.txt

david@LAPTOP-02Q7BFKO MINGW64 ~/Documents/Cursos/Git y GitHub/proyecto1 (master)
$ git status
On branch master
Changes not staged for commit:
 (use "git add <file>..." to update what will be committed)
 (use "git restore <file>..." to discard changes in working directory)
 modified: historia.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

En conclusion `$ git rm --cached` es más agresivo que `$ git reset head`, porque `$ git rm --cached` elimina el archivo de Git, no solamente lo que habia en Staging, sino todo el archivo. Deja de ser trackeado.

Con `$ git reset head` el archivo sigue siendo trackeado, solo se eliminan de Staging los ultimos cambios.

11

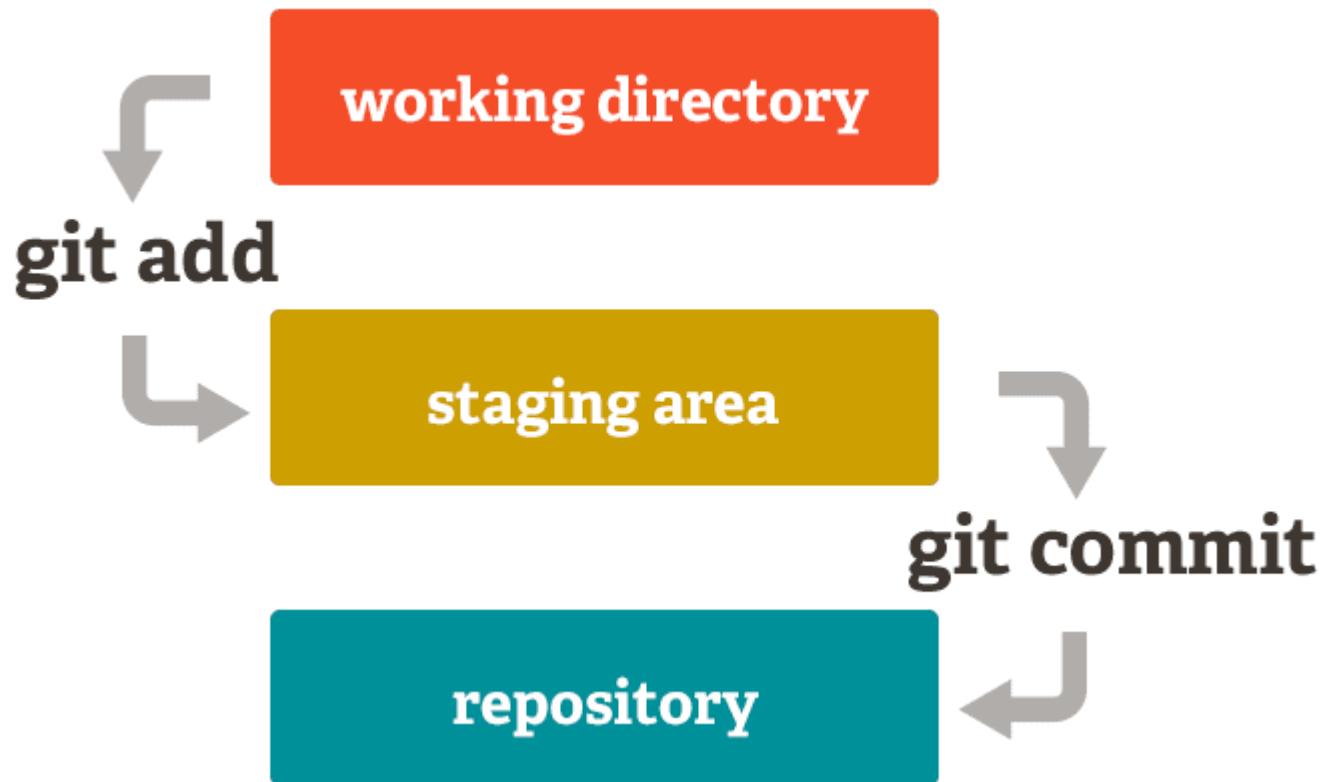


Nicolas Leal Estudiante · hace 2 meses



Flujo de trabajo básico con u...

Por si alguien quedo con duda de que es el Staging area aqui les dejo esta imagen quita dudas:



17   

Carlos Mario Rueda Suarez Estudiante · hace 2 meses

Alguien me podría hacer el favor de darme un ejemplo en el que utilice los siguientes comandos y por que?

- 1)rm-cached,
- 2)rm-forced
- 3)reset-soft
- 4)reset-hard
- 5)checkout

La verdad ya he visto varias veces los videos y las lecturas y sigo sin entender cuando y por qué utilizarlos, tampoco tengo muy claro como funcionan 😕

1   

Jair Olmos · hace 2 meses

tengo entendido Fredy nos explico en un video anterior que el staging es un lugar que se prepara en la RAM.

Partiendo de esto entendemos mejor que git rm --cached elimina la informacion que esta en el staging (RAM) y deja vivo lo que esta en el disco duro pudiendo asi recuperar la informacion en caso de requerirlo, a diferencia de git rm --forced que elimina la informacion del staging (RAM) y tambien la del disco duro

Flujo de trabajo básico con u...

recuperar los cambios)

git checkout te hace ir al instante requerido sin necesidad de borrar el historial

4



Cristian Jose Curup Muyus Estudiante • hace 2 meses

Entonces si elimino un archivo con \$ git rm --force, ¿podría recuperarlo con un \$git reset --hard siempre y cuando el archivo exista en el commit al que estoy intentando regresar?



0



Juan José Rodríguez Balza • el mes pasado

Es correcto, con tener la ID del commit puedes recuperar el archivo aun cuando ha sido borrado con git reset --hard . Si no se tiene la ID del commit lo puedes encontrar usando el comando git reflog.

2



Noé Lara Estudiante • hace 2 meses

Pero que pasa si solo queremos sacar 1 archivo de staging, ocupariamos

```
git rm --cached "Nombre del Archivo.extension"
```



3



Luis Kennedy Saavedra Fuentes • hace 2 meses

Para sacar solo un archivo de staging:

```
git resetHEAD <file.extension>
```

El archivo sacado del staging sigue en el repositorio con los cambios del último commit.

7



lidiar86 • hace 2 meses

Este lo eliminaría rm es para eliminar



Flujo de trabajo básico con u...

[Ver más respuestas](#)

Jessica Huancollo Chambi Estudiante • hace 2 meses

Uhm, consulta:

El comando \$git reset HEAD es igual al comando \$git rm --cached entonces?

Ya que solo lo saca del staging area pero, no lo borra.



3



jorge luis jaime sánchez • hace 2 meses

la diferencia de \$git reset HEAD es que devuelve el archivo al ultimo commit y lo sigue rastreando en cambio el \$git rm --cached ademas de sacarlo del staging lo elimina del repositorio y ya no le dará seguimiento pero el archivo seguirá en tu carpeta.

5



Carlos Alberto Basilio Torres • hace 2 meses

Por lo que entiendo

`git rm --cached "archivo"`

borras el staging de un archivo pero con

`git rest HEAD`

borras todo tu staging

2



Luis Kennedy Saavedra Fuentes • hace 2 meses

Con este comando El archivo es sacado del staging pero sigue en el repositorio con los cambios del último commit.

`$ git resetHEAD <file.extension>`

Con este comando El archivo es sacado del staging y también del repositorio.



Flujo de trabajo básico con u...

Ejemplo con: git rm --cached

```
LKS Engineer@DESKTOP-U6E9MPM MINGW64 ~/Documents/PLATZI/Git-Github/proyecto1
(master)
$ git rm --cached prueba.txt
rm 'prueba.txt'

LKS Engineer@DESKTOP-U6E9MPM MINGW64 ~/Documents/PLATZI/Git-Github/proyecto1
(master)
$ git status
On branch master
Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 deleted: prueba.txt

Untracked files:
 (use "git add <file>..." to include in what will be committed)
 prueba.txt
```

Ejemplo con: git reset HEAD

```
Admin@DESKTOP-U6E9MPM MINGW64 ~/Documents/PLATZI/Git-Github/proyecto1
(master)
$ git resetHEAD prueba.txt
Unstaged changes after reset:
M prueba.txt

Admin@DESKTOP-U6E9MPM MINGW64 ~/Documents/PLATZI/Git-Github/proyecto1
(master)
$ git status
On branch master
Changes not staged for commit:
 (use "git add <file>..." to update what will be committed)
 (use "git restore <file>..." to discard changes in working directory)
 modified: prueba.txt
```

---

3



Nicolas Leal • hace 2 meses

\*\*Mira \*\* resetHEADen este caso es para sacar un archivo del Staging area

`git resetHEAD <nombreDelArchivo.laextensión>```



Flujo de trabajo básico con u...

 lidiar86

• hace 2 meses

Es fácil de recordar si recuerdas los comandos de terminal que normalmente usas en Linux, se usa rm para remover (eliminar) "rm" significa remover:

sudo rm /Documents/archivo.txt: este lo elimina de tu carpeta y no hay mas registro. Es lo mismo en Git, lo elimina de staging y del proyecto. Esto es para que se borre tanto de tu proyecto como de Git ya que si solo lo borras de tu proyecto, seguira en Git.

Mientras que reset es un reseteo del HEAD, quitas todo lo que este en head o sea en staging, pero si agregas el nombre del archivo le dices que resetee del HEAD solo ese archivo:

git reset HEAD "archivo.extencion":quita solo el archivo

git resetHEAD: quita todo lo que esta en staging

3



Jessica Huancollo Chambi • hace 2 meses

Ok, ya lo tengo más claro , gracias 😊

1

[Ver más respuestas](#)

Yazuline Viveros Bedoya Estudiante • hace 2 meses

Gracias por la aclaración.



2



John Ruiz Estudiante • hace 2 meses

Es decir, es como tener un BACKUP todo el tiempo de los que hacemos. LA verdad que está genial el curso,



0



JORGE ROJAS Estudiante • hace 2 meses

Lesuento que practicando y practicando los comandos de rm y reset me pasó justo lo que freddy advirtió y es que se me borraron varios commits y no he podido recuperarlos. Mi history ya tenia como 7 commits y solo me quedaron 2, el primero y el último. Lo que me extraña es que yo estaba haciendo las pruebas solo con el primero y nose que pasó. En fin me queda de aprendizaje y saber que esto es realmente de mucho cuidado. Espero freddy explique mas adelante algún comando especial para recuperar los archivos borrados de forma agresiva por error.



4



Flujo de trabajo básico con u...

me parece que se puede recuperar los commits, para eso se tiene q tener un backup de los commit para q con el identificador puedes volver a un commit borrado. Eso es lo que me acuerdo de alguna clases q llevé con Leonidas Esteban.

1



JORGE ROJAS • hace 2 meses

Gracias jessica por el dato aunque realmente no sé como tener el indicador puesto que nunca los anoté en otro lugar, pero como he visto Freddy dice que Git no borra nada, entonces imagino que explicará como recuperar un commit así

0



Noé Lara • hace 2 meses

Vale aclarar que con:

```
git reset +ID--hard
```

Se te borra todo de Git, pero el directorio actual está intacto una, perfectamente puedes volver a hacer commit, pero si, los commits y historial se VA.

0



Alejandro Vidal • hace 2 meses

me pasó lo mismo.

0

[Ver más respuestas](#)

Andrés Mauricio Pérez Peña Estudiante • hace 2 meses

Excelente aclaración, muchas gracias



0



Irving Lopez Estudiante • hace 2 meses

como dice @freddier en su video siguiente, es muy importante que **practiquen** los comandos, y para que les quede claro este artículo **NO basta con leerlo, deben de practicar los comandos**, como el mismo dice no importa que echen a perder esta carpeta, total es una carpeta de prueba, peor sera si



Flujo de trabajo básico con u...



Agustín Villegas Quintero Estudiante · hace 2 meses

<\$git reset HEAD>

Cambia el archivo a un estado anterior al de aplicar el comando git add (unstaged)

2



jonathan2138 · hace 2 meses

piensa que este comando saca tu archivo del staging (deja de estar bajo “supervision”) y puedes volver a hacer tus cambios que creas necesarios y ya con esto tu archivo estara listo para empezar a hacerle supervision con el comando de git add .

0



Diego Andres Jimenez Quintero Estudiante · hace 2 meses

estoy aprendiendo git para poder seguir con mi carrera de desarollo web

3



Sebastián Andrade Estudiante · hace 2 meses

Que es el staging?

0



JOSE PINTO VEGA · hace 2 meses

el es staging es el paso intermedio entre el directorio y el repositorio, al cual accedemos con: git add y luego para pasarlo al repositorio git commit -m “mensaje de tus cambios en el documento para luego reconocer la versiones”

2



Sebastián Andrade Estudiante · hace 2 meses

Quedaría genial un git reset y un git checkout en la vida real xD

2



Flujo de trabajo básico con u...

Es cierto, soy nuevo en git y practicando nuevamente el ejercicio de la clase volver en el tiempo, hice la prueba del "git reset (ID 1er commit) --hard " y efectivamente! Hazta eliminó la carpeta css y el html, volví al momento antes de crear todos estos nuevos archivos, es decir solo con el historia.txt como pretendía.



 jonathan2138 · hace 2 meses

Es bastante peligroso si tenemos trabajo adelantado, de mi parte comparto que por curioso elimine todos los archivos que tenia en una carpeta (estaba jugando cuando estabamos en la clase de rm) y cuando Freddy explico esto del git reset, todos los archivos (que ya habia dado por perdidos) aparecieron como por arte de magia...

0

 Jorge Fidel Zubieta Choque Estudiante · hace 2 meses

Yo estoy confundido! alguien me puede explicar diferencia entre  
1 - git reset --soft vs git rm --cached?  
2 - git reset --hard vs git rm --force?



 Edwin Antonio Jimenez Palma · hace 2 meses

1. *git reset --soft* es para regresar a una version anterior, pero mantiene en staging los cambios que puedes tener del archivo. En cambio el *git rm --cached* elimina los archivos que tengas en staging, pero los mantiene en el disco duro de tu equipo, es decir conserva la ultima versión a la que le hiciste commit.
2. *git reset --hard* es para regresar a una version anterior sin mantener ningún cambio en staging, no quedará nada mas que la version a la que decides regresar y el registro de los commits anteriores a esa version. En cambio el *git rm --force* es para eliminar archivos de git pero tambien los elimina del disco duro de tu equipo.

Espero haber sido útil.

4

 Jorge Alberto Alba Vega · hace 2 meses

1 La diferencia es que el rm --cached no te borra los registros y el historial, solo saca los cambios de ese archivo de stage. mientras que el soft, te borra el historial y el directorio, y te deja el archivo en el Stage.

0



Flujo de trabajo básico con u...

hola, tengo una duda ¿solo debo poner git rm --cached y listo? por que veo que me pide mas opciones.

0



Johan Herrera • hace 2 meses

Hola Diego!

El comando `git rm --cached` se usa para eliminar los archivos que se encuentren en staging pero no los elimina del disco duro.

es decir, cuándo haces un `(git add texto.txt)` estás enviando el archivo de texto a staging.

Entendiendo eso ponemos el ejemplo de que te hayas equivocado y no hubieses querido hacer ese staging, es ahí cuando entra el comando `(git rm --cached)` para eliminar lo que este en el staging

EJM: Aquí hice un `git status` y se muestra que lo tengo en staging

```
● ● ● johanherrera@MBP-de-Johan:~/desktop/proyecto1
johanherrera@MBP-de-Johan proyecto1 % git status
On branch master
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)

modified: historia.txt
johanherrera@MBP-de-Johan proyecto1 %
```



Si quisiera deshacer esto tendría que usar el comando y el nombre del texto.

`git rm historia.txt --cached`

```
● ● ● johanherrera@MBP-de-Johan:~/desktop/proyecto1
johanherrera@MBP-de-Johan proyecto1 % git rm historia.txt --cached
rm 'historia.txt'
johanherrera@MBP-de-Johan proyecto1 %
```



Ahora hagamos un `git status` de nuevo para ver qué sucede.

[Screen Shot 2020-11-12 at 1.17.23 PM.png](#)

0



Miguel Abache Estudiante • hace 2 meses

Muy buena explicacion , gracias

0



María Cristina Tamayo Ossa Estudiante • hace 2 meses

WOW, excelente explicación.

Es mejor prevenir que lamentar.

1



Flujo de trabajo básico con u...

Este resumen me habría servido muchísimo cuando inicié a utilizar git, ahora con esta info que conozco, puedo utilizar mejor esta herramienta



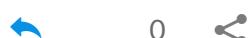
 Martín Martínez Estudiante · hace 2 meses

Ahora entiendo lo que decía Freddy en la primera clase, git tiene un poder INMENSO!



 Naldo Duran Estudiante · hace 2 meses

n



 Mauricio Alberto Maldonado Narváez Estudiante · hace 2 meses

quedo muy claro



 Sandra Milena Rojas Herrán Estudiante · hace 2 meses

Excelente aporte que salva muchos puestos de trabajo!!! Gracias.



 Carlos Jiménez Estudiante · hace 2 meses

Gracias, importante observación!



 Pablo Pizarro R. Estudiante · hace 2 meses

Excelente



Flujo de trabajo básico con u...