El comando git stash te permite almacenar temporalmente (o guardar en un stash), los cambios que hayas efectuado en el código en el que estás trabajando para que puedas trabajar en otra cosa y, más tarde, regresar y volver a aplicar los cambios.

Guardar los cambios en stashes resulta práctico si tienes que cambiar rápidamente de contexto y ponerte con otra cosa, pero estás en medio de un cambio en el código y no tienes todo listo para confirmar los cambios.

Git stash lo puedes usar sin necesidad de crear una nueva rama o hacer un commit. Además, no pierdes tus cambios.

* **git stash:** guarda los cambios temporalmente en memoria cuando no quieres hacer un commit aun
* **git stash save “mensaje”:** guarda un stach con mensaje
* **git stash list:** muestra la lista de cambios temporales
* **git stash pop:** trae de vuelta los cambios que teníamos guardados en el ultimo stash
* **git stash apply stash@{n}:** trae el stash que necesites con indicar su número dentro de las llaves
* **git stash drop:** borra el ultimo stash
* **git stash clear:** borra todos los stash

**Cómo funciona el comando git stash**

Para agregar los cambios al stash se utiliza el comando:

git stash

Podemos poner un mensaje en el stash, para así diferenciarlos en “git stash list” por si tenemos varios elementos en el stash. Esto con:

git stash save "mensaje identificador del elemento del stashed"

**Ejemplo de uso de git stash**

El stashed nos permite cambiar de [rama o branch](https://platzi.com/clases/1557-git-github/19947-que-es-un-branch-rama-y-como-funciona-un-merge-en-/), hacer cambios, trabajar en otras cosas y, más adelante, retomar el trabajo con los archivos que teníamos en staging, pero que podemos recuperar, ya que los guardamos en el stash.

Por ejemplo:

$ git status

On branch master

Changes to be committed:

 new file: style.css

Changes not staged for commit:

 modified: index.html

$ git stash

Saved working directoryandindex state WIP onmaster: 5002d47 our new homepage

HEADisnowat5002d47 our new homepage

$ git status

On branch master

nothingtocommit, working tree clean

**Cómo ver los stash en git**

**Utiliza git stash pop.** Este muestra los cambios guardados en el stash, también podemos mostrar los cambios de un stash determinado usando su índice que nos muestra el git stash.

El stashed se comporta como una stack de datos comportándose de manera tipo [LIFO](https://es.wikipedia.org/wiki/LIFO) (del inglés *Last In, First Out*, «último en entrar, primero en salir»), así podemos acceder al método pop.

El método **pop** recuperará y sacará de la lista el **último estado del stashed** y lo insertará en el **staging area**, por lo que es importante saber en qué *branch* te encuentras para poder recuperarlo, ya que el stash será **agnóstico a la rama o estado en el que te encuentres**. Siempre recuperará los cambios que hiciste en el lugar que lo llamas.

Para recuperar los últimos cambios desde el stash a tu staging area utiliza el comando:

git stash pop

Para aplicar los cambios de un stash específico y eliminarlo del stash:

git stash pop stash@{<num\_stash>}

Para retomar los cambios de una posición específica del stash puedes utilizar el comando:

git stash apply stash@{<num\_stash>}

Donde el <num\_stash> lo obtienes desde el git stash list

**Cómo ver el listado de elementos en el stash**

Para ver la lista de cambios guardados en stash y así poder recuperarlos o hacer algo con ellos podemos utilizar el comando:

git stash list

Retomar los cambios de una posición específica del Stash || Aplica los cambios de un stash específico.

**Cómo crear una rama con el stash**

Para crear una rama y aplicar el stash más reciente podemos utilizar el comando:

git stash branch <nombre\_de\_la\_rama>

Si deseas crear una rama y aplicar un stash específico (obtenido desde git stash list) puedes utilizar el comando:

git stash branch nombre\_de\_rama stash@{<num\_stash>}

Al utilizar estos comandos **crearás una rama** con el nombre <nombre\_de\_la\_rama>, te pasarás a ella y tendrás el **stash especificado** en tu **staging area**.

**Cómo eliminar elementos del stash**

Para eliminar los cambios más recientes dentro del stash (el elemento 0), podemos utilizar el comando:

git stash drop

Pero si, en cambio, conoces el índice del stash que quieres borrar (mediante git stash list) puedes utilizar el comando:

git stash drop stash@{<num\_stash>}

Donde el <num\_stash> es el índice del cambio guardado.

Si, en cambio, deseas eliminar todos los elementos del stash, puedes utilizar:

git stash clear

**Consideraciones:**

* El cambio más reciente (al crear un stash) **SIEMPRE** recibe el valor 0 y los que estaban antes aumentan su valor.
* Al crear un stash tomará los archivos que han sido modificados y eliminados. Para que tome un archivo creado es necesario agregarlo al staging area con git add [nombre\_archivo] con la intención de que git tenga un seguimiento de ese archivo. O también utilizando el comando git stash -u (que guardará en el stash los archivos que no estén en el staging).
* Al aplicar un stash este no se elimina, es buena práctica eliminarlo.