Repositorio Git

02/03/2015

scastiello

Contenido

[Introducción 3](#_Toc413747428)

[Resumen de comandos 4](#_Toc413747429)

[Actualización código 5](#_Toc413747430)

[Subir código 6](#_Toc413747431)

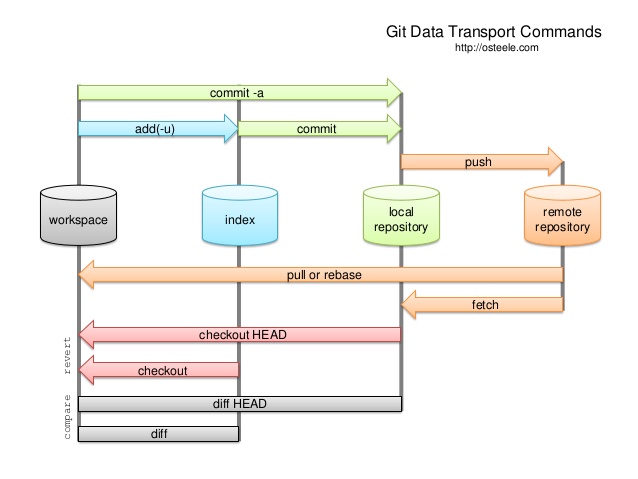
[Generar un tag 8](#_Toc413747432)

[Deshacer ultimo commit 9](#_Toc413747433)

# Introducción

En este documento, se va a detallar el proceso a seguir en distintas gestiones que se realizan sobre un repositorio git.

* Actualizar código
* Subir código
* Guardar código
* Fusionar código



Se recomienda la visualización de la siguiente guía online, que explica brevemente la filosofía y el uso de un repositorio git: <http://rogerdudler.github.io/git-guide/index.es.html>

Para obtener más información acerca de ciertos comandos de git, podéis acceder a las siguientes páginas:

* <http://git-scm.com/doc>
* <http://www.edy.es/dev/docs/git-guia-rapida/>

# Resumen de comandos

A continuación se listan una serie de comandos, que se han ido recopilando y son los más utilizados:

|  |  |
| --- | --- |
| Comando | Descripción |
| git status | Nos muestra el estado actual de nuestra copia de trabajo |
| git log | Muestra el historial de commits |
| git stash | Nos guarda nuestros cambios actuales |
| git stash pop | Recupera los cambios guardados |
| git pull origin master | Actualiza nuestra copia con los cambios del repositorio remoto |
| git fetch –tags | Actualiza los tags de nuestra copia con los tags del repositorio remoto |
| git push origin master | Sube al repositorio remoto todos los commits locales |
| git push --tags --force | Sube al repositorio remoto todos los tags locales |
| git tag -a <tag\_name> | Agrega un tag, solo si no existe |
| git tag -f <tag\_name> | Agrega un tag, si ya existe lo sobrescribe |
| git diff > <diff\_name> | Hace una copia de los cambios actuales sobre un fichero |
| git apply <diff\_name> | Aplica una un fichero con cambios |
| git add <file\_name> | Marca como modificado un fichero, para poder commitearlo |
| git rm <file\_name> | Marca como borrado un fichero, para poder commitearlo |
| git reset <file\_name> | Desmarca como agregado un fichero, para no comitearlo(no revierte los cambios). |
| git checkout <file\_name> | Deshace los cambios locales que no están comiteados. |

# Actualización código

El proceso general a seguir en caso de que necesitemos realizar una actualización de nuestra copia de trabajo, es el siguiente:

El primer paso a seguir, y solo si tenemos cambios locales, es guardar estos cambios locales, podemos detectar si tenemos cambios locales o no a través del comando.

git status

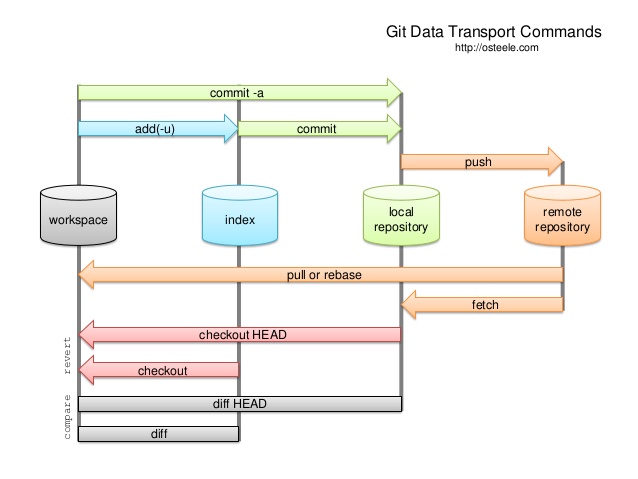
En caso de que tengamos **cambios locales**, debemos de guardarlos mediante el comando:

git stash

Una vez que tenemos los cambios locales guardados, podemos proceder con la actualización de la copia.

git pull origin master (password: “git”)

git fetch --tags (password: “git”)



Una vez nos hemos actualizado, debemos volver a aplicar los cambios anteriormente guardados, mediante el comando:

git stash pop

Si todo ha ido bien, en la salida de este último comando no aparecerá ninguna línea marcada en rojo, esto quiere decir que no han surgido conflictos en nuestra actualización y podemos continuar trabajando.

En caso de tener conflictos en nuestra actualización, debemos resolverlos. Tenemos que dirigirnos a los ficheros marcados en rojo y revisar las líneas de código con el siguiente formato:

<<<<<<< Updated upstream

Cambios del repositorio remoto

=======

Cambios locales

>>>>>>> Stashed changes

Una vez resuelto el conflicto sobre el fichero, debemos marcarlo como resuelto.

git add <file\_name> ó git reset <file\_name>

# Subir código

Antes de hacer un “commit”, **siempre es necesario realizar una actualización del código**, para ello debemos seguir los pasos del apartado anterior.

Una vez estamos actualizados y tenemos nuestros cambios listos para subir debemos seguir los siguientes pasos.

Marcar los ficheros que queremos que se suban en el “commit”, aquí tenemos varias posibilidades. Podemos marcar todos los ficheros nuevos y modificados con el comando:

git add -A

También podemos únicamente seleccionar los ficheros que están modificados con el comando:

git add -u

Y por último, es posible marcar los ficheros uno a uno con el comando:

git add <file\_name>

En caso de que también se eliminaran ficheros, deben ser marcados de otra manera:

git rm <file\_name>

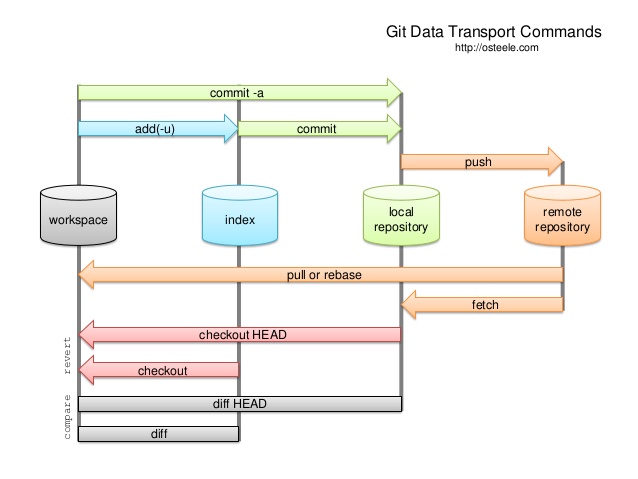
En caso de equivocación, es posible desmarcar estos ficheros, para que cuando se efectúe el “commit” no sean actualizados:

git reset <file\_name>

Una vez que tenemos todos los ficheros marcados y preparados para subir, debemos realizar el “commit”.

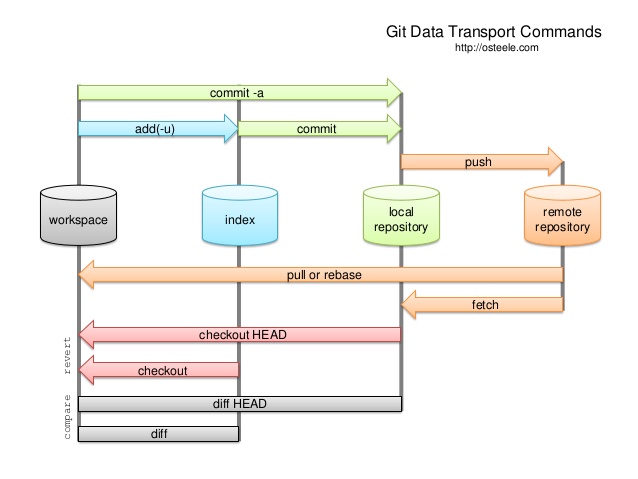
git commit

Al ejecutar este comando, se abre el programa “vim” con la plantilla por defecto, la cual debemos rellenar de acuerdo a los cambios de los ficheros que se van a actualizar. Al guardar y salir del “vim” el “commit” se realiza automáticamente.



Con el comando anterior el cambio realizado es únicamente visible por la persona que lo ha realizado, por lo que es necesario un paso más para que el resto de usuarios tengan visible el cambio.

git push origin master



Por último para comprobar que el proceso se ha realizado correctamente, podemos visualizar nuestro “commit” mediante el historial:

git log

# Generar un tag

El repositorio permite realizar una serie de marcas o “tags” para localizar fácilmente versiones del código, entregas o actualizaciones.

Podemos consultar las marcas existentes en nuestro repositorio. El siguiente comando, muestra un listado de las mismas:

git tag

Para crear una de estas marcas, basta con ejecutar el siguiente comando:

git tag –a <tag\_name>

Una vez ejecutado el comando el repositorio queda marcado en el “commit” actual. En caso de que nos hayamos equivocado, es posible rehacer el “tag”, mediante el siguiente comando:

git tag –f <tag\_name>

Por último, una vez que tenemos el “tag” correctamente generado, es necesario que lo notifiquemos al repositorio remoto:

git push --force –tags

De esta manera el resto de usuarios podrán actualizar estos “tags” mediante otro comando:

git fetch --tags (password: “git”)

# Deshacer último commit

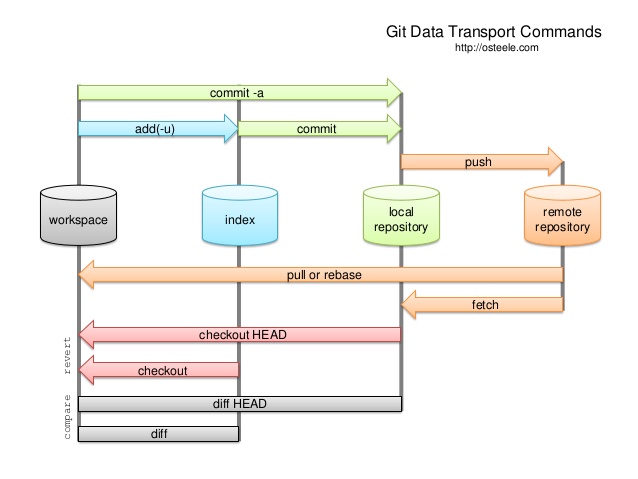
Es habitual realizar un “commit” y darse cuenta después de que no se ha actualizado la copia antes de realizarlo. En este proceso se indica cómo deshacer un “commit” del que no se ha hecho un “push”.

Debemos ejecutar el siguiente comando para deshacer el último “commit”:

git reset --soft HEAD~1

Una vez se ha revertido, es necesario actualizarse la copia consultando los apartados anteriores.

Una vez actualizados, podemos rehacer el “commit”, consultar el apartado correspondiente.



En caso de que el commit se encuentre en el repositorio remoto (se ha hecho un push), es necesario ejecutar el siguiente comando, teniendo en cuenta que si alguien se ha actualizado con dicho commit no sería válido:

git push origin HEAD –force

