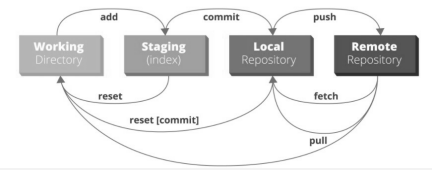
# Chuleta de Git en Español Cheat Sheet

by José Carlos (jcarlosweb) via cheatography.com/131894/cs/26603/

## Estados




## Publicar

`git push origin main`

`origin`: Repositorio remoto, por defecto origin

`main`: Rama principal, por defecto en github (antes era master)

`git push` Envía los cambios automáticamente al origin y rama actual

`git push -f` Para cuando quieres forzar la subida y trabajas solo o también cuando quieres joder al equipo

`git push -u origin main`: Tambien se puede usar para indicar el repositorio remoto.

`-u`: Es un flag de `--set-upstream`, se refiere al repositorio remoto principal al que harás pull y push, esta opción se utiliza una sola vez y se puede configurar en el .gitconfig

## Ramas (Branch)

`git branch`: Lista toda las ramas

`git branch -d <branch-name>` : Elimina una rama

`git checkout <branch-name>` : Cambia de rama

`git checkout -b <branch-name>` : Cambia de rama y si no existe la crea

## Unir o fusionar ramas

`git merge <rama>`: Fusiona la rama indicada con la actual

`git merge --no-ff <rama>` : Fusiona la rama sin avance rápido, es decir creando commit y dejando historial en el head

`git merge --squash <rama>` : Unir todos los commit en uno solo de la rama y fusionar.

`git rebase <rama>`

`git rebase -i HEAD~3` : Los últimos 3 commits

`-i`: Modo interactivo pudiendo poner squash en todos menos el primero para unir todos los commits

Unir commits sin rebase: https://stackoverflow.com/a/5201642/2046442

`git reset --soft HEAD~3 && git commit`

`git commit --edit -m"$(git log --format=%B --reverse HEAD..HEAD@{1})"`

`git reset --rebase`: Deshacer rebase

`git reset --merge`: Deshacer merge

**Git rebase** básicamente lo que hace es recopilar uno a uno los cambios confirmados en una rama, y reaplicarlos sobre otra. Utilizar rebase nos puede ayudar a evitar conflictos siempre que se aplique sobre commits que están en local y no han sido subidos a ningún repositorio remoto.

es una manera de unir los cambios que hagamos en un branch dedicado a algún feature o un hotfix hacia nuestro branch principal pero manteniendo cierto orden en nuestra línea de tiempo, teniendo todo dentro de un sólo timeline.

## .gitconfig ~/.gitconfig

`git config --global user.name "-jcarlosweb"`

`git config --global user.email "nomeacuerdo@gmail.com"`

`git config --global core.f-ilemode false`

`git config --global pull.rebase true`

`git config --global rebase.au-toStash true`

`git config --global init.defa-ultBranch "main"`

## Iniciar repositorio

`git init`

`git clone https://github.com/us-ername/repository-name.git`

## Añadir cambios

`git add .` añade todo

`git commit -m "Mi primer commit"`

`git commit -am "Mi segundo commit para archivos modificad-os"`

`-am` es evitar ahorrarte el paso de añadir y hacer todo de una vez, pero solo vale para archivos modificados, no para nuevos.

`git commit --amend -m "Este es el mensaje correcto"`: Modificar un commit existente sin haber hecho push

## Deshacer cambios en Local

`git reset <commit-id-or-HEAD~1>`

`--mixed`: Por defecto. Elimina los commit, no conserva los cambios en el stage area y mantiene los cambios en el working tree

`--soft`: Elimina los commit, conserva los cambios en el stage area y mantiene los cambios en el working tree

By **José Carlos** (jcarlosweb)
cheatography.com/jcarlosweb/

Published 13th February, 2021.
Last updated 17th February, 2021.

## Deshacer cambios en Local (cont)

`--hard`: Elimina los commit, no conserva los cambios en el stage area y deshace los cambios en el working tree

```
git checkout -- <nombre-archivo>
```

`git checkout <commit>` Utiliza git checkout para desplazarte por el historial de confirmaciones y consultarlo.

## Deshacer cambios en remoto

```
git revert <commit-id>
```

```
git revert @
```

`@`: Último commit

## Actualizar desde remoto

`git remove -v`: Listar todos los remotos configurados

`git remote show <remote>`: Mostrar información sobre el remoto

`git fetch <remote>`: Descargar todo los cambios, pero no lo integra en el HEAD

`git pull <remote> <branch>`: Descarga y directamente lo integra/merge en el HEAD. Es decir fetch + merge

`git pull --rebase`: La opción "--rebase" nos permite, de forma ordenada, uno a uno ir aplicando nuestros commits sobre el código que estaba en github

`git fetch && git rebase --autostash`: Guardar automáticamente los cambios antes de realizar la operación de rebase. Antes de ese lanzamiento, rebase simplemente se negaría a ejecutarse. Equivalente a: `git stash & git pull --rebase & git stash pop`

```
git rebase --abort
```

## Actualizar desde remoto (cont)

```
git rebase --continue
```

El concepto de **HEAD** es muy simple: se refiere al commit en el que está tu repositorio posicionado en cada momento. Por regla general HEAD suele coincidir con el último commit de la rama en la que estés, ya que habitualmente estás trabajando en lo último. Pero si te mueves hacia cualquier otro commit anterior entonces el HEAD estará más atrás.

https://www.campusmvp.es/recursos/post/-git-los-conceptos-de-master-origin-y-head.aspx

## Estado, inspeccionar y comparar

| Estado | Inspeccionar | Comparar |
|---|---|---|
| `git status` | `git log --oneline` | `git diff` |
| | `git blame <archivo>` | |
| | `git show` | |
| | `git reflog` | |

## .Gitignore

`/docker/`: Ignora solo el directorio docker

`docker/`: Ignora todos los directorios docker

`*.log`: Ignora todos los archivos con la extensión log

`/var/cache/*`: Ignorar todos los archivos de la carpeta cache

`!var/cache/.gitkeep`: Mantener el archivo. gitkeep

**Reiniciar .gitignore**

```
git rm -r --cached . && git add . && git commit -m "update .gitignore"
```

## Utilidades

**Cherry Pick**: Traerte uno varios commit de una rama a otra sin necesidad de hacer un merge

```
git cherry-pick <commit-ID>
```

**Bisect**: Para encontrar bugs

```
git bisect start <begin> <end>
```

```
git bisect reset
```

```
git bisect run "test -f mojón"
```

```
git bisect run test.sh
```

**Dejar de monitorizar un archivo**

```
git update-index --assume-unchanged nocheck.php
```

**Volver a monotorizar**

```
git update-index --no-assume-unchanged nocheck.php
```

**Stash**: coge los cambios sin confirmar (tanto los que están preparados como los que no), los guarda aparte para usarlos más adelante y, acto seguido, los deshace en el código en el que estás trabajando

`git stash`:Lo almacena

`git stash pop`: Lo recupera

`git stash apply`: Lo recupera, pero no lo elimina del stash

**Alias**

https://stackoverflow.com/a/2553799/-2046442

**Git Extras**

https://github.com/tj/git-extras/blob/master/-Commands.md

**TLDR**

https://github.com/raylee/tldr-sh-client

https://explainshell.com/explain?cmd=git+c-ommit+-a+-m

https://www.clicksolution.es/#home

By **José Carlos** (jcarlosweb)
cheatography.com/jcarlosweb/

Published 13th February, 2021.
Last updated 17th February, 2021.

| Git Gui |
| --- |
| Source Tree |
| PHPStorm Git |
| VS Code Git |

By **José Carlos** (jcarlosweb)
cheatography.com/jcarlosweb/

Published 13th February, 2021.
Last updated 17th February, 2021.
Page 3 of 3.