Git y GitHub

Resumen de Git y de sus principales comandos

Juan Quemada, DIT - UPM

Resumen e índice de comandos Git más habituales

Glosario Git I

Glosario de doc Git en
https://git-scm.com/docs/gitglossary

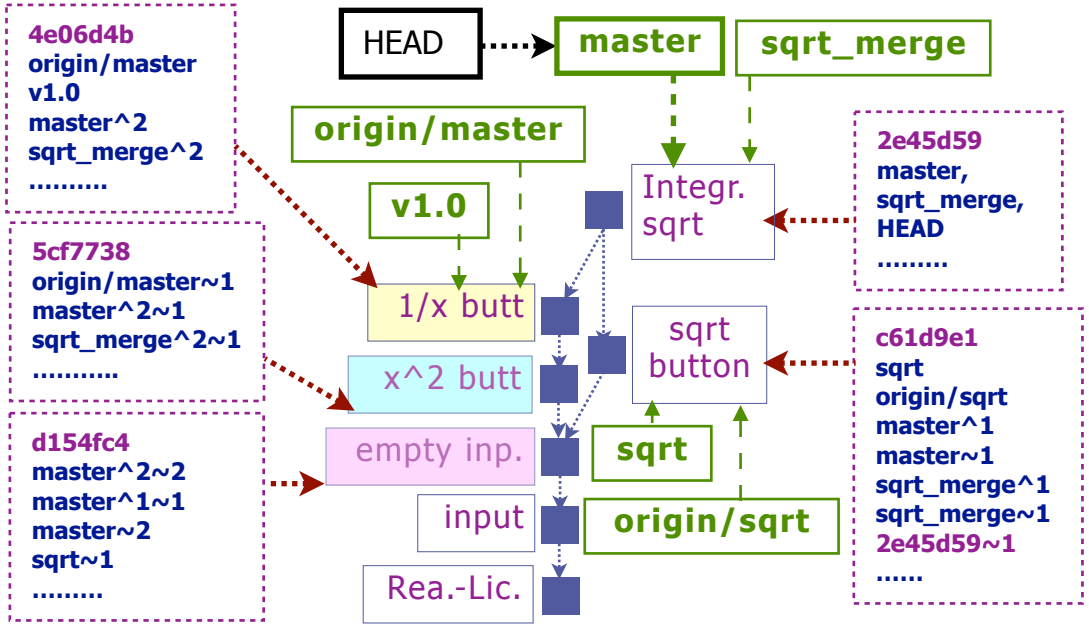
- **ancestro (de un commit):** commit perteneciente a la historia de ese commit que se referencia como: commit~1, commit~2, ...
- **árbol de trabajo (working tree):** conjunto de ficheros y directorios guardados en un commit o contenidos en el directorio de trabajo.
- **auto-merge:** operación de integración de ramas (merge) sin conflictos, donde git crea el commit automáticamente.
- **clone:** acción de clonar un repositorio remoto a uno local.
- **commit:** árbol de trabajo guardado en un repositorio git, con la instantánea de un proyecto.
- **checkout:** acción de restaurar, en el directorio de trabajo, el contenido de un commit guardado en un repositorio git, actualizando también el índice (o registro) y HEAD.
- **directorio de trabajo:** directorio que contiene el árbol de trabajo actual.
- **fast-forward:** operación de integración de ramas (merge) que reutiliza otra integración igual realizada anteriormente.
- **fetch:** acción de incluir (visibilizar) una rama de un repositorio remoto en uno local.
- **grafo de commits:** conjunto de commits y relaciones de precedencia de un repositorio.
- **índice, área de cambios, registro o cache (en desuso):** registro del estado de los ficheros en el directorio de trabajo a incluir en el próximo commit.
- **limpio/sucio:** directorio de trabajo sin/con cambios respecto a commit anterior (HEAD).
- **master:** convención para nombrar la rama principal de un repo creada con primer commit.
- **nombre, hash o identificador SHA-1 (de un objeto git):** identificador global único de un objeto git obtenido con el algoritmo SHA-1, que identifica el objeto en cualquier cualquier repositorio git y garantiza su integridad.
- **objeto (git):** la unidad de almacenamiento de Git creada al guardar el árbol de trabajo en un commit. Se identifica con el identificador SHA-1 y no puede modificarse una vez creado.

Glosario Git II

Glosario de doc Git en
https://git-scm.com/docs/gitglossary

- **origin:** convención para nombrar al repositorio origen de un repositorio local, normalmente el repositorio remoto del que se ha clonado.
- **padre de un commit:** commit del que se ha generado directamente este commit. Por ejemplo, se referencia como: commit^1, commit^2, ...
- **pull:** acción de integrar en una rama de un repositorio local los nuevos commits de esa rama en un repositorio remoto.
- **push:** acción de actualizar una rama remota con los nuevos commits de una rama local.
- **rama:** Secuencia de commits, ordenada por fechas, que soporta un desarrollo. Se implementa como una referencia en el repositorio.
- **rama local:** rama cuyos commits están copiados en el repositorio local y donde es posible realizar desarrollos y cerrar nuevos commits.
- **rama remota:** rama de un repositorio remoto visible en un repositorio local.
- **rama tracking:** rama local asociada a una remota.
- **referencia o ref:** nombres usados para implementar ramas, HEAD y otros elementos de Git. Suele empezar por refs/ (p.e. refs/heads/master, ..).
- **refspec:** se usa en fetch y push para describir la relación entre la referencia remota y local.
- **remote:** nombre corto de un repositorio remoto asociado a su URL de acceso.
- **repositorio o repo:** colección de referencias y base de datos de objetos git de un proyecto.
- **repositorio bare:** repositorio sin directorio de trabajo, utilizado normalmente para crear repositorios remotos accesibles a través de Internet.
- **repositorio local:** repositorio con directorio de trabajo, donde se trabaja en un proyecto.
- **repositorio remoto:** repositorio en un servidor accesible a través de Internet identificado con un URL. Se usan para compartir código, para backup, ..

Grafo de commits de un repositorio y sus referencias



✦Grafo de commits

- **Relaciona** cada **commit** de un repositorio con los **commits utilizados para generarlo**
  - Un **commit** con una flecha saliente, solo tiene un padre porque se generó **modificando el padre**
  - Un **commit** con dos o mas flechas salientes tiene varios padres, se generó **integrando ramas**

✦Las formas mas habituales de identificar los commits del grafo son:

- **HEAD:** puntero o referencia al commit actual en el directorio de trabajo
- **Rama**: puntero al último commit de una rama de desarrollo
  - **Rama local**: rama copiada en el repositorio local (**master**, **sqrt**, ..)
  - **Rama remota**: rama de un repositorio remote (**origin/master**, **origin/sqrt**, **remote/origin/sqrt**, ..)
- **Identificador** corto o largo: identificador único de un commit (**d154fc4, 973751d2**, ,…)
- **Ancestro**: **commit~n** -> sigue la línea del 1er padre, p.e. master~2, sqrt~1, ..
- **Padre**: **commit^n** -> El primer **padre** (c^1) es también el primer ancestro (c~1)
- **Tag o etiqueta**: **v1.0**, **v0.1**, …. -> Se añaden con: **git tag -a v1.0**

Mas info sobre referencias a commits: https://www.atlassian.com/git/tutorials/refs-and-the-reflog, https://git-scm.com/docs/gitrevisions

.gitignore

✦**git ls-file --other --ignored --exclude-standard**

- lista todos los ficheros de este proyecto ignorados por Git

```
# .gitignore es un fichero que informa a Git de los ficheros que no debe gestionar.
#  - git status no los presentará como ficheros untracked.
#  - git add . no los añadira al índice (staging area).

# .gitignore se crea en el directorio o subdirs de trabajo y afecta todo el arbol asociado.

# Su contenido: líneas con patrones de nombres.
#  - Puede usarse los comodines * y ?
#  - Patrones terminados en / indican directorios
#  - Un patron que empiece con ! indica negación
#  - Se ignoran líneas en blanco y que comiencen con #
#  - [abc] indica cualquiera de los caracteres entre corchetes
#  - [a-z] indica cualquier carácter ASCII (rango desde a hasta z)

# Ejemplo
private.txt     # excluir los ficheros con nombre "private.txt"
*.class         # excluir los ficheros acabados en ".class"
*.[oa]          # excluir ficheros acabados en ".o" y ".a"
!lib.a          # no excluir el fichero "lib.a"
*~              # excluir ficheros acabados en "~"
testing/        # excluir directorio "testing"
```

Reglas de sintaxis de comandos

Literal

✦ **literal: git, clone, checkout, remove, --,….**

- palabra o símbolo separado por blanco o nueva línea que debe incluirse literalmente

Elemento genérico

✦ **< elemento >**

- Representa un elemento (singular) o varios (plural) del tipo indicado por el nombre
  - **<options>** representa una o varias opciones, por ejemplo **-q**, **-v**, **--all**, **--oneline**, …
  - **<commit>** representa un commit, por ejemplo **d154fc4, master, master~2, ..**

Elemento opcional

✦ **[elemento]**

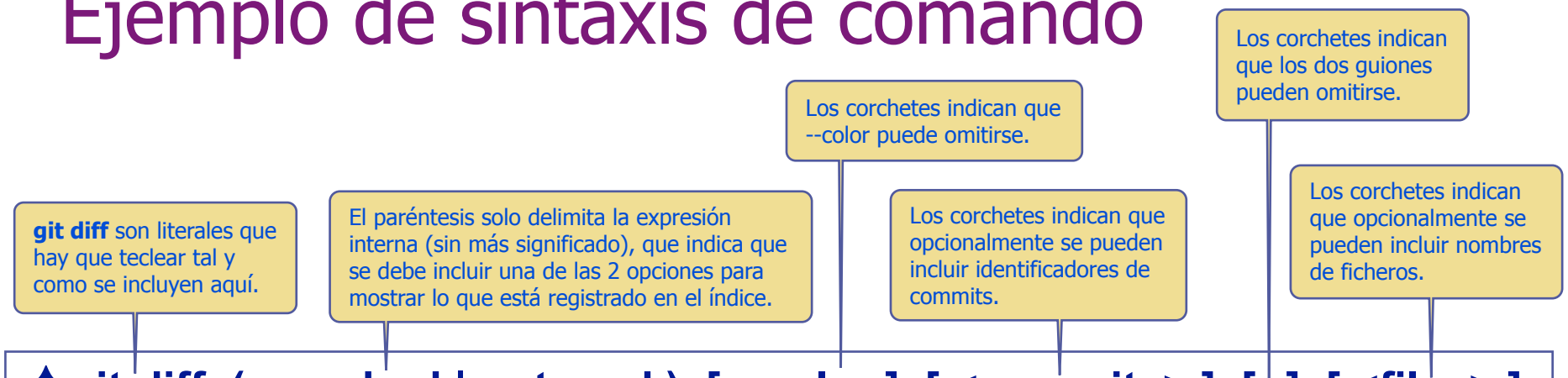
- Los corchetes delimitan uno o varios elementos opcionales que podrán incluirse o no
  - **[<options>] [<commit>]** o **[-q]** indica que estos elementos son opcionales
  - **[[--] <files>]** los corchetes pueden anidarse indicando que todo ([[--] <files>]) o la primera parte ([--]) son opcionales

Elementos alternativos

✦ **elemento1 | elemento2**

- La barra vertical separa alternativas, que irán delimitadas por corchetes o paréntesis
  - **[add | rename | remove | add-url]** indica opcionalidad, es decir puede incluirse uno de estos literales o ninguno
  - **(-b | -B)** paréntesis solo delimita, es decir se usan cuando debe incluirse necesariamente una de las opciones

Ejemplo de sintaxis de comando



✦git diff (--cached | --staged) [--color] [<commits>] [--] [<files>]

- Muestra diferencias de código en los ficheros registrados en el indice (staged) respecto a la última versión guardada (commit)
- Algunas opciones de interés
  - **--staged, --cached** mostrar diferencias con ficheros registrados (staged)
  - **--color** mostrar diferencias en colores diferentes (opción por defecto)
- Ejemplos: mostrar cambios entre commits
  - **git diff --staged** mostrar diferencias en todos los fiches registrados (staged)
  - **git diff --cached file.js** mostrar diferencias en el el fichero file.js respecto al último commit

Doc. en https://git-scm.com/docs

Recetario de comandos habituales

Crear un repositorio local vacío

- **git init** ## Inicia directorio actual de trabajo como repositorio Git
- **git init proy1** ## Crea directorio proy1 y lo inicia como repositorio Git

Directorio de trabajo y commits

- **git add .** ## Añade todos los ficheros del directorio de trabajo al índice
- **git add f1, f2, f3** ## Añade los ficheros fi, f2 y f3 al índice

- **git reset .** ## Extrae todos los ficheros del índice (al dir. de trabajo)
- **git reset f1, f2, f3** ## Extrae los ficheros fi, f2 y f3 del índice

- **git status [-s]** ## Muestra estado de los ficheros en el directorio de trabajo

- **git commit -m "texto"** ## Guarda nuevo commit con título "texto"
- **git commit** ## Guarda nuevo commit y abre editor para crear título
- **git commit --amend [-m "texto"]**
  ## Modifica último commit con registros en índice !**OJO** cambia el id de commit¡

Doc. en https://git-scm.com/docs

Diferencias entre commits o con el directorio de trabajo

- **git diff**
  - Muestra los cambios de los ficheros **modifed** del directorio de trabajo respecto al commit anterior
- **git diff --unified=2000**
  - Muestra cambios de ficheros **modifed** con un contexto de hasta 2000 líneas (todo el fichero)
- **git diff [--] f1, f2, ..**
  - Muestra cambios en los ficheros **f1, f2, .. modifed** del dir. de trabajo respecto a commit anterior
- **git diff (--cached | --staged) [[--] f1, f2, ..]**
  - Muestra cambios en **<files> staged** respecto a la versión del commit anterior
- **git diff 97d75 master**
  - Muestra todos los cambios entre el commit 97d75 y la rama master
- **git diff sqrt master [--] fich1**
  - Muestra los cambios en fich1 entre las ramas sqrt y master
- **git diff -b 97d75 7f9y67**
  - Muestra los cambios entre commits 97d75 y 7f9y67 sin considerar espacios en blanco
- **git diff -b --unified=2000 97d75 7f9y67**
  - Muestra cambios de 97d75 y 7f9y67 sin considerar blancos con un contexto hasta de 2000 líneas
- **git diff -b --numstat 97d75 7f9y67**
  - Muestra estadísticas de cambios entre 97d75 y 7f9y67 sin considerar espacios en blanco
- **git diff --name-status 97d75 7f9y67**
  - Muestra nombres de ficheros con cambios entre commits 97d75 y 7f9y67 y su estado

Doc. en https://git-scm.com/docs

Cambios en el directorio de trabajo

- **git mv old_file new_file**
  - Cambia nombre del fichero old_file a new_file en el directorio de trabajo (y en el índice)
- **git rm file1, file2, ..**
  - Borra file1, file2, .. del directorio de trabajo y registra el borrado en el índice.
- **git rm (--cached | --staged) file1, file2, ..**
  - Borra file1, file2, .. del índice, los ficheros pasan de **staged** a **untracked.**
- **git checkout file1, file2, ..**
  - Elimina cambios en file1, file2, .. **modified** que pasan a **unmodified** (**OJO!** Cambios se pierden)
- **git checkout .**
  - Elimina cambios de los ficheros **modified,** que pasan a **unmodified** (**OJO!** Cambios se pierden).

Mostrar elementos de commits

- **git show master**
  - Muestra los elementos resumidos del último commit de la rama master.
- **git show 7f9y67**
  - Muestra los elementos resumidos del commit 7f9y67.
- **git show 7f9y67:package.json**
  - Muestra el contenido del fichero package.json del commit 7f9y67.
- **git show HEAD~2:package.json**
  - Muestra el contenido del fichero package.json del segundo ancestro de HEAD.

Doc. en https://git-scm.com/docs

Listar ramas

- **git branch [-v] [-vv] [-a] [-r]**
  - Lista las ramas locales y remotas de un repositorio

Crear ramas locales

- **git branch [-t | --track] <branch> <commit>**
  - Crear una rama de nombre <branch> que comienze en <commit>
- **git checkout [-f | ..] (-b | -B) <branch> [<commit>]**
  - Crear (o reiniciar) <branch> en <commit> y restaurarla en el directorio de trabajo

Borrar ramas

- **git branch (-d | -D) <branch>**
  - Borra la rama <branch> si se ha integrado ya (-d) o aunque no se haya integrado (-D)

Rehacer la historia de commits de una rama

- **git reset [<commit>]**
  - Deshace commits posteriores a **<commit>** dejando cambios en ficheros que pasan a **modified**
- **git reset --hard [<commit>]**
  - Elimina commits posteriores a **<commit>** (**OJO!** commits eliminados y sus cambios se pierden)

Cambiar la base de una rama

- **git rebase sqrt**
  - Cambiar commit de comienzo de una rama activa (HEAD) a sqrt rehaciendo todos sus commits
- **git rebase --abort** ## deshace rebase interrumpido (por conflictos de integración)
- **git rebase --continue** ## continua rebase interrumpido (por conflictos de integración)

Doc. en https://git-scm.com/docs

Mostrar historia

- **git log master**
  - Muestra la historia de commits de la rama master en formato largo.
- **git log -10 --oneline master**
  - Muestra los 10 últimos commits de la rama master en formato de una linea.
- **git log --oneline --follow package.json master**
  - Muestra los commits en formato corto, con cambios en package.json, de la rama master.
- **git log -10 --oneline --graph master**
  - Muestra los últimos 10 commits de la rama master incluyendo su grafo de integración de ramas.
- **git log --oneline --graph --all**
  - Muestra todo el grafo local con todas sus ramas e integraciones.

Restaurar commits en el directorio de trabajo

- **git checkout sqrt**
  - Restaura el último commit de la rama master en el directorio de trabajo.
- **git checkout -f 45gt6u**
  - Restaura commit 45gt6u en el directorio de trabajo, aunque haya cambios. **¡OJO Se pierden!**

Integrar ramas locales

- **git merge -m "texto" master**
  - Integra rama master en rama activa con "texto" como título del commit de integración generado
- **git merge --abort** ## deshace un merge interrumpido por conflictos de integración
- **git merge --continue** ## continua un merge interrumpido por conflictos de integración

Doc. en https://git-scm.com/docs

stash

- **git stash [<name>]**
  - Guarda en la pila de stashed las modificaciones del directorio de trabajo y el índice. Le da el nombre **name** si se incluye.
- **git stash list**
  - Lista el contenido de la pila de stashed
- **git stash apply [<name>] [<options>]**
  - Aplica los cambios del último stash guardado, o los del stash llamado **name**, a los ficheros del area de trabajo, y no actualiza el índice, excepto si se usa la opción --**index**, y no elimina el stash aplicado de la pila.
- **git stash drop [<name>]**
  - Elimina el último stash de la pila (o el indicado por **name**),
- **git stash pop [<name>]**
  - Aplica el último stash (o el indicado por **name**) y lo elimina de la pila.

Gestionar definiciones de repositorios remotos

- **git remote [add | rename | remove | set-url | ..] [<options>] ……**
- **git remote add backup https://github.com/ging/cal**
  - crea remote backup asociado al repositorio https://github.com/ging/cal
- **git remote rename backup backup1** ## cambia nombre de remote backup por backup1
- **git remote remove backup** ## borra remote backup
- **git remote set-url backup https://github.com/ging/cal1** ## Cambia url de remote backup

Doc. en https://git-scm.com/docs

Sincronizar repositorios

- **git clone https://github.com/jquemada/cal**
  - clona repo remoto (https://github.com/jquemada/cal) en un repo local de nombre cal
- **git clone https://github.com/jquemada/cal mi_cal**
  - clona repo remoto (https://github.com/jquemada/cal) en un repo local de nombre mi_cal
- **git push**
  - Sube a origin (remote por defecto) commits nuevos de master (rama por def.), si son compatibles
  - Equivale a: **git push origin master**
- **git push [-f] [--all] [<options>] [<remote> [[+]<rbranch>[:<lbranch>]]]**
  - Sube rama (**lbranch**) del repo local a rama (**rbranch**) de un remoto (**remote**).
  - **-f** o **+** sobre-escribe la rama remota. **--all** sube todas las ramas.
- **git pull**
  - Trae de origin (remote por defecto) commits nuevos de master (rama por def.), si son compatibles
  - Equivale a: **git pull origin master**
- **git pull [<options>] [<remote> [[+]<rbranch>[:<lbranch>]]]**
  - Integra rama (**rbranch**) de un remoto (**remote**) en rama local (**lbranch**).
  - **-f** o **+** sobre-escribe la rama local. **--all** integra todas las ramas.
- **git fetch [-f] [--all] [<options>] [<remote> [[+]<rbranch>[:<lbranch>]]]**
  - Trae una rama (**rbranch**) de un remoto (**remote**) a rama local (**lbranch**).
  - **-f** o **+** sobre-escribe rama local. **--all** trae todas las ramas.

Doc. en https://git-scm.com/docs

Guía resumida de comandos Git

add

✦git add [<options>] [<files> | .]

- Añade <files> al índice (para inclusión en nuevo commit)
  - **Ojo!** Ficheros **modificados** pero **no añadidos al índice con add** no irán en la versión
- Algunas opciones de interés
  - **-u** añade todos los ficheros **modified**, pero no los **untracked**
  - **-v** modo verboso (--verbose)
  - **-n** no añade ficheros, solo muestra si existen o serán ignorados (--dry-run)
  - **-i** preguntar en cada caso (--interactive)
  - **-e** abrir editor por defecto (normalmente vi) con diffs respecto a índice para re-editar cambios (--edit)
- Ejemplos:
  - **git add -u** añade todos los ficheros modificados al índice
  - **git add .** añade todos los ficheros del directorio de trabajo (.) al índice
  - **git add -i .** intenta añadir todos los ficheros al índice, preguntando antes
  - **git add file_1 file_2** añade file_1 y file_2 al índice

Doc. en https://git-scm.com/docs

branch

✦git branch [<options>] [<branch>] [<commit>]

- Listar, crear o borrar ramas
- Algunas opciones de interés
  - **-d, -D** borrar la rama, si está integrada (-d) o aunque no se haya integrado (-D)
  - **-f** forzar operación indicada aunque se pierdan cambios (—force)
  - **--track** crear rama tracking cuando va acompañado de -b
  - **-a, -r** listar ramas: todas (-a) o solo tracking remotas (-r)
  - **-v, -vv** modo verboso, muestra último commit (-v) y ramas tracking (-vv)
  - **--merged, --no-merged** muestra solo ramas integradas o no integradas
- Ejemplos: listar ramas
  - **git branch** listar las ramas locales del repositorio
  - **git branch -a -v** listar las ramas locales y remotas del repositorio en formato extendido
  - **git branch --merged** listar las ramas locales del repositorio ya integradas en otras
  - **git branch -vv** listar las ramas **tracking** con la rama remota asociada y su estado
- Ejemplos: crear, borrar o reconfigurar ramas locales
  - **git branch sqrt** crear rama sqrt en commit actual (HEAD)
  - **git branch sqrt d154fc4** crear rama sqrt en commit d154fc4
  - **git branch -d sqrt** borra la rama sqrt del repositorio
  - **git branch -f sqrt** reinicia rama sqrt borrando cambios y commits
  - **git branch --track r1/square** transforma la rama local **square**, ya existente, en tracking de **r1/square**
  - **git branch --track s2 r1/s1** transforma la rama local s2, ya existente, en tracking de r1/s1

Doc. en https://git-scm.com/docs

checkout I (restaurar commit o fichero)

✦git checkout [<options>] [<commit>]

- Restaurar ficheros, ramas o commits en el directorio de trabajo
  - Actualiza puntero **HEAD** al commit restaurado en el directorio de trabajo
- Algunas opciones de interés
  - **-q** no genera estadísticas de salida (—quiet)
  - **-f** forzar checkout aunque se pierdan cambios (--force)
- Ejemplos: cambio de rama
  - **git checkout master** restaura en el directorio de trabajo el último commit de master
  - **git checkout 4e06d4b** restaura en el directorio de trabajo el commit **4e06d4b**
    - **OJO! Peligro** modo detached HEAD, el commit activo no es una rama y los cambios a este commit no podrán guardarse a no ser que se cree una rama nueva

✦git checkout [<files> | .]

- Restaurar ficheros **modified** del directorio de trabajo al commit anterior
  - **!OJO** los cambios se perderán y no podrán volver a recuperarse
- Ejemplos: restaurar ficheros modified
  - **git checkout file1.js** elimina cambios en el fichero **modified file.js**
  - **git checkout .** elimina cambios en todos los ficheros **modified** del directorio de trabajo

Doc. en https://git-scm.com/docs

checkout II (restaurar y crear rama)

✦git checkout [<options>] (-b | -B) <branch> [<commit>]

- Crear rama y restaurar ficheros, ramas o commits de la rama
  - Actualiza puntero **HEAD** al commit que restaura en el directorio de trabajo
- Algunas opciones de interés
  - **-q** no genera estadísticas de salida (--quiet)
  - **-f** forzar checkout aunque se pierdan cambios (--force)
  - **-m** fuerza un merge con <commit> y con cambios en el área de trabajo
  - **-b <new_branch>** crea rama de nombre <new_branch> en el commit restaurado (HEAD)
  - **-B <new_branch>** crea rama de nombre <new_branch> en el commit restaurado (HEAD) y si existe la reinicia (borra los commits de la rama) hasta <commit>
- Ejemplos: creación de una rama en un commit determinado
  - **git checkout -b sqrt 4e06d4b** crea rama **sqrt** en 4e06d4b y la restaura en directorio de trabajo
  - **git checkout -B sqrt 4e06d4b** restaura rama **sqrt** en **4e06d4b y** borra sus commits si existen

✦git checkout [<branch>]

- Copia una rama remota en una local tracking y restaura la rama local (solo si no existe)
  - Actualiza puntero **HEAD** al commit restaurado en el directorio de trabajo
- Ejemplos: de copia de rama remota
  - **git checkout square** crea y restaura la rama tracking square asociada a **<remote>/square**

Doc. en https://git-scm.com/docs

clone

✦**git clone [<options>] <URL> [<directory>]**

- Copiar un repositorio identificado por <URL> a un directorio local:
  - Define el repositorio remoto como repositorio remote origin
  - Copia la rama master como rama tracking local de origin/master
  - Define además todas las ramas de origin como ramas remotas
- Algunas opciones de interés
  - **-v** modo verboso (--verbose)
  - **-q** no genera estadísticas de salida (--quiet)
  - **--no-hardlinks** fuerza una copia física de todos los ficheros cuando se copia repositorio local
- Ejemplos: clonar ficheros
  - **git clone https://github.com/CORE-UPM/cal_5com** clona en directorio local **cal_5com**
  - **git clone https://github.com/CORE-UPM/cal_5com cal** clona en directorio local **cal**

Doc. en https://git-scm.com/docs

commit

✦**git commit [<options>] [-m <msg>]**

- Guarda lo registrado en el índice como un nuevo commit en la rama actual
- Algunas opciones de interés
  - **-m <msg>** incluir el mensaje <msg> que identifica la versión
  - **-q** no genera estadísticas de salida (quiet)
  - -**a** añade al índice todos los ficheros modificados antes de generar commit
    - **OJO!** No añade ficheros untracked
  - **--amend** rehace el commit anterior con lo indicado en el índice
    - **OJO!** Corrige errores. Cambia identificador de versión y será incompatible con anterior
- Ejemplos: crear nuevo commit
  - **git commit -m "hola"** crea commit en rama actual con cambios registrados y mensaje "hola"
  - **git commit -a -m "hola"** añade todos los ficheros modificados al índice y crea commit
- Ejemplos: modificar commits existentes (**OJO!** Cambia el id del commit)
  - **git commit --amend** rehace último commit en rama actual con cambios registrados en índice
  - **git commit --amend -m "hola que tal"** rehace último commit con cambios registrados y en la rama activa con los cambios registrados en el índice y con mensaje "hola"

Doc. en https://git-scm.com/docs

diff I (diferencias entre commits)

✦**git diff [<opciones>] [<commit>] [<commit>] [[--] <files>]**

- Muestra diferencias de código entre commits, ramas, ficheros, etc
- Algunas opciones de interés
  - **--color** mostrar diferencias en colores diferentes (opción por defecto)
  - **--unified=2000** mostrar un contexto de hasta 2000 líneas (todo el fichero)
  - **-b** mostrar diferencias sin considerar espacios en blanco
  - **--name-status** mostrar solo nombres de ficheros con cambios y su estado
  - **--name-only** mostrar solo nombres de ficheros con cambios
  - **--stat** mostrar estadísticas de cambios
  - **--numstat** mostrar estadísticas, incluyendo número de lineas borradas e insertadas
- Ejemplos: mostrar cambios entre commits
  - **git diff -b HEAD d154fc4** diferencias entre HEAD y d154fc4 sin considerar blancos
  - **git diff master master~2** diferencias entre último commit de master y dos anteriores
  - **git diff sqrt master** diferencias entre las ramas master y sqrt
  - **git diff --stat sqrt master** estadísticas de cambios entre las ramas master y sqrt
  - **git diff sqrt master file.js** diferencias en file.js entre las ramas master y sqrt
  - **git diff --unified=2000 sqrt master** diferencias entre master y sqrt con todo el contexto
  - **git diff --name-only sqrt master** ficheros con diferencias entre master y sqrt

Doc. en https://git-scm.com/docs

diff II (diferencias en el directorio de trabajo)

✦git diff [[--] <files>]

- Diferencias en ficheros (<files>) **modifed** respecto al commit anterior
- Las opciones anteriores se pueden usar aquí también
  - **--color, --unified=2000, -b, --name-status, --name-only, --stat, --numstat, ...**
- Ejemplos: mostrar cambios de ficheros **modified**
  - **git diff** mostrar diffs de todos ficheros modified respecto al commit anterior
  - **git diff file.js** mostrar diffs de fichero modified file.js respecto al commit anterior

✦git diff (--cached | --staged) [[--] <files>]

- Diferencias en ficheros (<files>) **staged** respecto al commit anterior
- Las opciones anteriores se pueden usar aquí también
  - **--color, --unified=2000, -b, --name-status, --name-only, --stat, --numstat, ...**
- Ejemplos: mostrar cambios de ficheros staged
  - **git diff --cached** mostrar diffs de todos ficheros staged respecto al commit anterior
  - **git diff --cached file.js** mostrar diffs de fichero staged file.js respecto al commit anterior

Doc. en https://git-scm.com/docs

fetch

✦ git fetch [<options>] [<repository> [[+]<local_branch>[:<remote_branch>]]]

- actualizar en el repositorio local las ramas indicadas de un repositorio remoto
  - [<remote> [[+]<rbranch>[:<lbranch>]] actualiza la rama local (lbranch) con la remota (rbranch)
    - **+** fuerza actualización aunque haya commits incompatibles (no-ff) en la rama local
- Algunas opciones de interés
  - **--all** actualizar todas las ramas de todos los repositorios remotos definidos
  - **-f** o **+** forzar actualización si hay commits incompatibles (--force) **OJO!** puede perder commits
  - **-p** eliminar ramas que no ya existan en el remoto (--prune)
  - **-v, -q** modos verboso (--verbose) y sin realimentación (--quiet)
- Ejemplos: actualización de ramas de repositorios remotos definidos
  - **git fetch cal** crea las ramas remotas del remote cal_branches o actualiza su estado si existen
  - **git fetch** actualiza el estado de todas las ramas tracking
  - **git fetch --all** crea o actualiza el estado de todas las ramas de todos los remotes definidos
  - **git fetch -p origin** la opción -p (--prune) actualiza las ramas de origin eliminando las que ya no existen
- Ejemplos: actualización de ramas con refspecs: [[+]<rbranch>[:<lbranch>]]]
  - **git fetch origin square:sqrt**
    - Crea o actualiza la rama local sqrt con los commits de la remota origin/square
  - **git fetch origin pull/1/head:s1**
    - Crea o actualiza la rama local s1 con el pull_request 1 del repositorio remoto origin en GitHub
  - **git fetch cal_branches +s1:s1**
    - Crea o actualiza la rama local s1 con la remota cal_branches/s1 aunque sean incompatibles
  - **git fetch https://github.com/jquemada/cal square:square**
    - Crea o actualiza la rama local square con la rem. square de https://github.com/jquemada/cal

Doc. en https://git-scm.com/docs

init

✦git init [<options>] [<repository>]

- Crea <repository> y lo inicia como repositorio Git (crea directorio **.git**)
  - A partir de este momento ya se pueden invocar todos comandos Git en el directorio
- Algunas opciones de interés
  - **-q** no genera estadísticas de salida (--quiet)
  - --bare Crea repositorio bare, sin directorio de trabajo
- Ejemplos:
  - **git init** Inicia el directorio donde se invoca como repositorio Git
    - El repositorio debe ser el directorio de trabajo del terminal de comandos
  - **git init proy1** Crea o inicializa el repositorio en el directorio **proy1**
    - Crea el directorio proy1 (mkdir proy1), si no existe, e invoca en él el comando **git init**

Doc. en https://git-scm.com/docs

log I (historia de un commit)

✦**git log [<options>] [<commit>] [--follow <file>]**

- Muestra la historia de commits que lleva al <commit> indicado
- Algunas opciones de interés
  - **--oneline**: muestra metadatos resumidos en una linea solo con id_corto y mensaje
  - **-n**: muestra n últimos commits
  - **-p**: muestra commits con diferencias (diff) con versión anterior
  - **--stat**: muestra también estadísticas del commit
  - **--since=2.weeks o --until=2.weeks**: muestra commits después o hasta hace 2 semanas
  - **--follow <file>** muestra la historia de un fichero particular, incluyendo los cambios de nombre
- Ejemplos: historia de commits
  - **git log** muestra todos los commits hasta HEAD (HEAD es la opción por defecto y se omite)
  - **git log -3 HEAD** muestra los 3 últimos commits hasta HEAD (HEAD se deja aquí).
  - **git log** --**since=2.weeks --until=1.week** muestra commits de las penúltima semana.
  - **git log --oneline** muestra commits resumidos en una linea (incluye id_corto y mensaje).
  - **git log master** historia de commits de la rama master (desde cualquier posición).
  - **git log --oneline -2 d154fc4**: muestra 2 últimos commit resumidos hasta d154fc4**.**
  - **git log -3 master~2**: historia de 3 commits desde segundo commit anterior a master.
- Ejemplos: historia de ficheros
  - **git log --follow calculator.htm**: historia de versiones del fichero calculator.htm desde HEAD.
  - **git log sqrt --follow .gitignore**: historia del fichero .gitignore en la rama sqrt.

Doc. en https://git-scm.com/docs

log II (grafo de commits)

✦**git log [<options>] [<commit>] [--follow <files>]**

- Muestra la historia de commits que lleva al <commit> indicado
- Algunas opciones de interés
  - **--oneline**: muestra metadatos resumidos en una linea solo con id_corto y mensaje
  - **-n**: muestra n últimos commits
  - **--all** muestra la historia de todas las ramas (debe omitirse <commit>)
  - **--graph** muestra grafo de ramas integradas en la rama actual
- Ejemplos: historia desde commits
  - **git log --oneline --graph** muestra el grafo de integración de la rama actual (formato corto)
  - **git log --oneline --graph sqrt** muestra el grafo de integración de la rama sqrt (formato corto)
  - **git log --oneline --graph --all** muestra el grafo completo de commits (formato corto)
  - **git log --oneline --all -2** muestra los 2 últimos commits del repositorio (formato corto)
  - **git log --oneline --all --graph -3** muestra grafo de commits con profundidad 3 (f. c.)

Doc. en https://git-scm.com/docs

merge

✦git merge [<options>] [-m <msg>] <commit>

- Integra el commit indicado en HEAD y genera un commit de integración con “msg”
  - Si la integración no tiene conflictos, genera automáticamente un commit por auto-merge
  - Si HEAD es un ancestro del commit a integrar pasa al commit a integrar con ff (fast-forward)
  - Si hay conflictos, git los incluye como modificaciones de los ficheros afectados
    - Una vez resueltos, **git --continue** (o **git commit …**) genera el commit de integración
- Algunas opciones de interés
  - **-m <msg>** fijar el mensaje del commit de integración
  - **--abort** aborta la operación de merge y reconstruye el estado anterior
  - **--continue** continua la operación de merge y genera commit de integración
  - **-v, -q** modos verboso (--verbose) y sin realimentación (--quiet)
- Ejemplos:
  - **git merge -m “msg” master** integra master en HEAD y genera commit de integr. con “msg”
  - **git merge master** integrar la rama master con el desarrollo o rama actual (HEAD)
    - Pedira editar el mensaje con el editor por defecto para el commit de integración
  - **git merge 4e06d4b** integrar el commit 4e06d4b con el desarrollo o rama actual (HEAD)
  - **git merge --abort** aborta la operación de merge en curso, trata de reconstruir estado anterior
  - **git merge --continue** continúa la integración para generar commit de integración (después de resolver conflictos y registrarlos en el índice)

Documentación completa: https://git-scm.com/docs.
Merging vs. Rebasing: https://www.atlassian.com/git/tutorials/merging-vs-rebasing
Merge: https://git-scm.com/docs/git-merge
Rebase: https://git-scm.com/docs/git-rebase

mv

✦**git mv [<options>] <origin> <destination>**

- Mover o renombrar un fichero, directorio o enlace eliminando del índice
  - **OJO!** El comando **mv** de **UNIX** crea un fichero untracked, pero el original sigue en el índice
- Algunas opciones de interés
  - **-f** forzar mover o renombrar aunque fichero destino exista (--force)
  - -**v** modo verboso (--verbose)
- Ejemplos: cambios de ficheros (eliminando el original del índice)
  - **git mv file1.js file2.js** cambia el nombre de file1.js a file2.js
  - **git mv file1.js dir1/file2.js** mueve file1.js al directorio dir1 con nombre file2.js
  - **git mv -f file1.js dir1** mueve file1.js al directorio dir1, aunque ya exista dir1/file1.js

Doc. en https://git-scm.com/docs

pull

✦**git pull [<options>] [<repository> [[+]<local_branch>[:<remote_branch>]]]**

- traer la rama remota indicada e integrarla con una rama del repositorio local
  - Equivale normalmente a hacer primero **git fetch ..** y a continuación **git merge ..**
- Algunas opciones de interés
  - **--all** integrar todas las ramas en el repositorio local
  - **-f** forzar actualización si hay commits incompatibles (--force) **OJO!** Los commits eliminados se pierden
  - **+** **+** en una refspec es equivalente a **-f** o a **--force**
  - **-p** eliminar ramas tracking que no existan ya en el remoto (--prune)
  - **-r** realizar un rebase en vez de merge (--rebase)
  - **-v, -q** modos verboso (--verbose) y sin realimentación (--quiet)
- Ejemplos: integración de ramas remotas con ramas locales
  - **git pull cal_branches square** integra la rama square de cal_branches en la rama activa
  - **git pull https://github.com/jquemada/cal_branches square**
    - integra la rama square de jquemada/cal_branches en la rama activa
  - **git pull origin pull/1/head** Integra el pull_reques #1 en la rama activa

Doc. en https://git-scm.com/docs

push

- **git push [<options>] [<repository> [[+]<local_branch>[:<remote_branch>]]]**
  - actualizar repositorio remoto y objetos asociados con las ramas locales indicadas
    - [[+]<local_branch>[:<remote_branch>]] actualiza la rama remota indicada con los commits de la local
      - **+** fuerza actualización aunque haya commits incompatibles (no-ff) en la rama local (equivale a -f)
  - Algunas opciones de interés
    - **--all** actualizar todas las ramas del repositorio remoto
    - **-f** forzar actualización si hay commits incompatibles (--force) **OJO!** Los commits incompatibles se pierden
    - **-v, -q** modos verboso (--verbose) y sin realimentación (--quiet)
  - Ejemplos: actualización de ramas de un repositorio remoto
    - **git push**
      - Actualiza las ramas remotas de las ramas locales **tracking** definidas en el repositorio local
    - **git push cal_branches sqrt:square**
      - Actualiza la rama remota **cal_branches/square** con los nuevos commits de la local **sqrt**
    - **git push https://github.com/jquemada/cal sqrt:square**
      - Actualiza la rama remota **cal/square** con los nuevos commits de la local **sqrt**
  - Ejemplos: sustitución de ramas de un repositorio remoto
    - **git push -f origin sqrt** sustituye rama sqrt en repositorio origin por la rama local, aunque haya commits incompatibles (**OJO! Peligroso**: Los commits borrados no podrán ser recuperados)
  - Ejemplos: eliminación de ramas de un repositorio remoto
    - **git push origin :sqrt** Borra la rama **sqrt** en el repositorio remoto **origin**
    - **git push origin --delete sqrt** Similar a anterior, opción posible en versiones recientes de git

Doc. en https://git-scm.com/docs

rebase

✦git rebase [<options>] <commit>

- cambiar el commit de comienzo (base) de una rama (rehaciendo commits)
  - Rebase rehace todos los commits de la rama utilizando un bucle, donde cada iteración integra un commit de la antigua rama en la nueva rama
    - La integración es similar a la de **git merge ..**, pudiendo haber auto-merge, fast-forward, ..
    - Si hay conflictos, Git los marca y para el proceso (bucle). Los conflictos deben integrarse con un editor y el proceso (bucle) debe continuarse con **git rebase --continue | --skip | ..** para procesar el siguiente commit. Y así hasta integrar todos los commits o abortar.
- Algunas opciones de interés
  - **--continue** continuar el bucle de rebase después de editar conflictos de integración de commit
  - **--skip** continuar el bucle de rebase integrando commit actual con siguiente
  - **--abort** abortar operación de rebase y volver a estado inicial
  - **-i --interactive** rebase que muestra el plan de rebase al comenzar, permitiendo eliminar o reorganizar commits en la nueva rama. Muy efectivo para reorganizar proyectos.
- Ejemplos:
  - **git rebase master** cambiar comienzo de rama actual (HEAD) a master
  - **git rebase -i master** cambiar comienzo de rama actual (HEAD) a master interactivamente
  - **git rebase 4e06d4b** cambiar comienzo de rama actual (HEAD) a commit 4e06d4b
  - **git rebase --continue** continuar rebase después de editar conflictos de integración
  - **git rebase --skip** continuar rebase integrando commit actual con siguiente

Documentación completa: https://git-scm.com/docs.
Merging vs. Rebasing: https://www.atlassian.com/git/tutorials/merging-vs-rebasing
Merge: https://git-scm.com/docs/git-merge
Rebase: https://git-scm.com/docs/git-rebase

remote

✦git remote [-v]

- gestionar repositorios remotos (remote) relacionados, por ejemplo
  - **git remote** muestra repositorios remotos (remotes) definidos
  - **git status -v** muestra enlace de repositorios remotos (remotes) definidos (--verbose)

✦git remote add <name> <URL>

- Crea la referencia <remote_**name**> a repositorio remoto identificado por un **URL**, p.e.
  - **git remote add cal_inic https://github.com/jquemada/cal_inic**

✦git remote remove <name>

- Elimina la definición del repositorio **<remote_name>**, por ejemplo
  - **git remote remove cal_inic**

✦git remote rename <old_name> <new_name>

- cambia el nombre de **<remote_name1>** por **<remote_name2>**, por ejemplo
  - **git remote rename cal_inic cal_initialize**

✦git remote set-url <name> <URL>

- Cambia el URL del repositorio **<remote_name>**, por ejemplo
  - **git remote rename cal_initialize https://github.com/jquemada/cal_initialize**

✦git remote set-branches [--add] <name> <branches>

- Sustituye o añade (opción --add) tracking branches de **<remote>**, por ejemplo
  - **git remote set-branches origin master** sustituye tracking branches actuales por master
  - **git remote set-branches --add origin sqrt** añade sqrt a tracking branches

reset

✦git reset [-q] [<files>]

- Operación inversa a git add .., ficheros (<files>) dejan de estar **staged**
- Ejemplo: borrar ficheros del índice (de **staged** a **modified** o **untracked**)
  - **git reset file1.js** extrae file1.js del índice, preserva los cambios (inverso **git add <files>**)
  - **git reset** extrae todos los ficheros del índice, preserva cambios (inverso **git add .**)

✦git reset [-q] <commit>

- Cambia el puntero de rama y HEAD a <commit> y deja diferencias en ficheros
- Ejemplos:
  - **git reset 4e06d4b** resetea la rama actual a **4e06d4b** eliminando los commits posteriores y preservando cambios acumulados en el directorio de trabajo
  - **git reset --mixed 4e06d4b** equivalente a ejemplo anterior, --mixed es la opción por defecto

✦git reset --hard [-q] <commit>

- Cambia el puntero de rama y HEAD a <commit>
- Ejemplo: (**OJO!** commits se pierden y no se pueden recuperar)
  - **git reset --hard 4e06d4b** resetea la rama actual a **4e06d4b** eliminando los commits posteriores sin preservar cambios (**OJO!** Todos los commits eliminados se pierden)

Doc. en https://git-scm.com/docs

rm

✦git rm <files>

- borra ficheros del directorio de trabajo y del índice
  - **OJO!** El comando **rm** de **UNIX** no debe utilizarse porque borra un fichero del directorio, pero el fichero sigue en el índice y en los commits siguientes
- Algunas opciones de interés
  - **-f** forzar borrado si fichero es modified (--force)
  - **-n** simula ejecución sin hacer cambios (--dry-run)
- Ejemplos de borrado de ficheros del directorio de trabajo
  - **git rm file1.js file2.js** borra file1.js y file2.js del directorio de trabajo y del índice
  - **git rm -f file1.js** borra file1.js del directorio de trabajo y del índice, aunque sea modified
  - **git rm -n f*.*** muestra que ficheros borraría el comando, sin borrar dichos ficheros

✦git rm (--cached | --staged) <files>

- borra los ficheros (<files>) del índice, pasándolos de **staged** a **untracked**
- Ejemplos:
  - **git rm --cached file1.js** borra file1.js del índice, pasando de **staged** a **untracked**.

Doc. en https://git-scm.com/docs

show

✦**git show [<options>] [<commit>] [[--] <file>]**

- Muestra metadatos de un commit y diferencias con el anterior
- Algunas opciones de interés
  - **--oneline**: muestra metadatos resumidos en una linea solo con id_corto y mensaje
- Ejemplos:
  - **git show** muestra metadatos de HEAD y diferencias con el commit anterior
  - **git show master** muestra metadatos del último commit de master y diferencias con anterior
  - **git show HEAD~1** muestra metadatos de commit anterior a HEAD y diferencias
  - **git show d154fc4** muestra metadatos de **d154fc4** y diferencias con el commit anterior
  - **git show HEAD file1.js** muestra metadatos HEAD y diferencias de file.js con commit anterior
  - **git show d154fc4:file1.js** muestra contenido del fichero **file.js** del commit **d154fc4**
  - **git show HEAD~2:package.json** contenido de package.json del 2o ancestro de HEAD

Doc. en https://git-scm.com/docs

stash

✦git stash [<name>]

- Guarda en una pila las modificaciones del directorio de trabajo y en el índice,
- deja restaurados el directorio de trabajo y el índice, por ejemplo
  - **git stash** guarda las modificaciones un la pila

✦git stash list

- Lista el contenido de la pila de stashed p.e.
  - **git stash**

✦git stash apply [<name>] [<options>]

- Aplica los cambios del último stash guardado, o los del stash llamado **name**,
- a los ficheros del area de trabajo,
- y no actualiza el índice, excepto si se usa la opción **—index**,
- y no elimina el stash aplicado de la pila. Por ejemplo:
  - **git stash apply**

✦git stash drop [<name>]

- Elimina el último stash de la pila (o el indicado por **name**), por ejemplo
  - **git stash drop**

✦git stash pop [<name>]

- Aplica el último stash (o el indicado por **name**) y lo elimina de la pila, por ejemplo
  - **git stash pop**

Doc. en https://git-scm.com/docs

status

✦**git status [-s]**

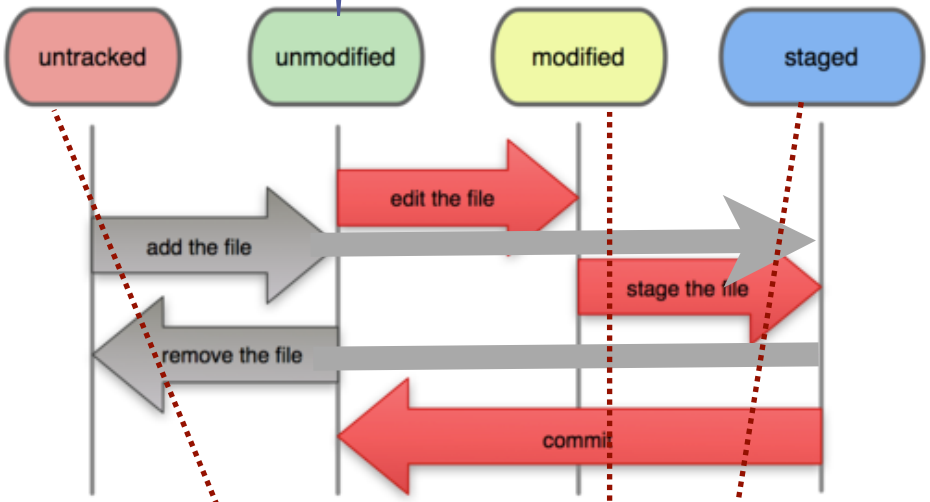
- muestra el estado del directorio de trabajo
  - los ficheros pueden estar **untracked**, **modified**, **staged**, **unmerged**, …
- Algunas opciones de interés
  - **-s**: muestra estado en formato corto (1 linea por fichero)
- Ejemplos de uso
  - **git status** muestra estado de cambios del directorio de trabajo
  - **git status -s** muestra estado de cambios del directorio de trabajo en formato corto

Doc. en https://git-scm.com/docs

"git status"

Ficheros de la versión anterior no modificados para la siguiente

*S. Chacon, B. Straub: https://git-scm.com/book/es/v1



git status muestra el estado del índice o area de cambios:

- ficheros staged (en la versión)
- ficheros modified (fuera de versión)
- ficheros untracked (fuera de versión)

opción -s: salida corta (short)

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#             modified:  README
#             new file:   CharIO.java
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#
#             modified:   benchmarks.rb
#
# Untracked files:
#   (use "git add <file>..." to include what will be committed)
#
#             merge.java
#             library/lib.js
```

Ficheros modificados incluidos en próxima versión

Ficheros modificados no incluidos en próxima versión

Ficheros excluidos de versión

tag

✦git tag [-a] [<options>] <name> [<commit>]

- Crea un tag con el nombre <**name**> en el commit <**commit**>, o en el commit actual,
- de tipo ligero si no se usa la opción **-a** (para uso temporal),
- o de tipo anotado si se usa la opción **-a** (se crea un commit nuevo).
- Por ejemplo, crear un tag anotado llamado v1.4 con un mensaje para el commit:
  - **git tag -a v1.4 -m "Version 1.4"**

✦git tag [-l <pattern>]

- Lista todos los tags existentes,
- o los que encajan con el patrón dado con la opción -l, por ejemplo
  - **git tag** lista todos los tags existentes.
  - **git tag -l v1.*** lista los tags que encajan con el patrón v1.*

Doc. en https://git-scm.com/docs

Final del tema