|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **git help <command>** |  |
| **git clone <uri> namedir** | *clona usando como nombre de directorio namedir.* |
| **git add <dir>** | *añade recursivamente todos los archivos del dir.* |
| **git diff --staged** | *compares staged changes with last commit* |
| **git commit -v** | *muestra el diff en el editor* |
| **git commit -a -m ”** | *automatically stage tracked files. No hace falta git add* |
| **git rm --cached <file or regexp>** | *Git no realiza un seguimiento del archivo, pero los deja en el directorio de trabajo. Útil cuando se olvida añadir archivos al .gitignore y ya hemos agregado dichos archivos al repositorio.* |
| **git rm <file>** | *borrarlos con git siempre.* |
| **git rm -f <file>** | *si ya está modificado y en el index.* |
| git mv <file> <renamed\_file> |  |
| **gitk** | *tcl/tk. Herramienta gráfica para git* |
| **git commit --amend** | *Modificar el mensaje del último commit* |
| **git reset HEAD <file>** | *to unstage* |
| **git checkout -- <file>** | *Descartar cambios en el directorio de trabajo.* |
| **git add -i** | *interactive staggin* |
| **git add -p** | *crea patch* |
| **git stash** | *guarda el estado en una pila y limpia el directorio para poder cambiar de rama* |
| **git stash list** | *muestra la pila* |
| **git stash apply** | *vuelve al estado original del dir. Stash{n} especifica uno concreto Y --index reaplica los cambios stagged* |
| **git stash pop** | *elimina el primero en la pila. O drop* |
| **git log -p -2** | *Muestra 2 últimos commits con diff* |
| **git log --stat** |  |
| **git log --pretty** |  |
| **git log --pretty=format:”%h - %an, %ar : %s”** |  |
| **git log --pretty=format;”%h %s” --graph** |  |
| **git log --since=2.weeks** |  |
| **git log <branch> --not master** | *Muestra commit de <branch> sin incluir los de master* |
| git log --abbrev-commit --pretty=oneline |  |
| **git diff master…contrib** | *Muestra solo el trabajo que la rama contrib actual ha introducido desde su antecesor común con master* |
| **git log <branch1>..<branch2>** | *Commits de branch2 que no están en branch1* |
| **git log origin/master..master** | *Muestra qué commits se van a enviar al servidor* |
| **git log origin/master..** | *Igual que el anterior. Se asume master o HEAD* |
| **git log refA refB --not refC** | *commits en refA y refB que no están en refC* |
| **git log master…experiment** | *commits de master o experiment, pero sin ser comunes. Con --left-right indica a qué rama pertenece cada uno* |
| **git remote -v** | *lista los repos remotos* |
| **git remote add [shortname] [url]** | *crea nuevo remote, es posible descargar el contenido de ese repo con git fetch [shortname]. Master branch en [shortcode]/master* |
| **git fetch <remote>** | *descarga trabajo nuevo a máquina local, no sobreescribe nada tuyo. ( git pull sí hace merge automaticamente si se esta realizando un seguimiento de esa branch)* |
| **git push [remote-name] [branch-name]** | *sii nadie ha hecho push antes* |
| **git remote show [remote-name]** | *inspecciona remote.* |
| **git remote rename <old-name> <new-name>** | *también renombra branches: quedaría <new-name>/master* |
| **git remote rm <remote-name>** | *p.e si el contribuidor ya no contribuye más* |
| **git remote add bitbucket git@bitbucket.org:algui91/grado\_informatica\_tsi\_practicas.git** | *Añadir un nuevo repositorio remoto con el nombre deseado. Por ejemplo si ya tenemos uno en github y queremos añadir otro para bitbucket* |
| **git push -u bitbucket –all** | *Subir el proyecto a bitbucket. A partir de ahora se puede seleccionar a qué repo publicar con***git push nombre\_repo\_remoto** |
| **git tag** | *muestra las etiquetas actuales* |
| **git tag -l ‘v1.4.2.\*’** | *acepta regex* |
| Dos tipos de tag: |  |
| **Lightweight** : puntero a commit ( branch que no cambia ) |  |
| **Annotated** : se almacenan como objetos en la db, con checksum, nombre del creador, email, fecha, mensaje, posibilidad de firmarla con[GPG](http://elbauldelprogramador.com/seguridad/como-cifrar-correos-con-gpg-con-mailvelope/). ( recomendada ) |  |
| **git tag -a <tagname> -m ‘mensaje’** | *annotated tag* |
| **git show <tag-name>** | *muestra información asociada.* |
| **git tag -s <tag-name> -m ‘message’** | *la firma con gpg* |
| **git tag <tag-name>** | *lightweight tag* |
| **git tag -v <tag-name>** | *verifica tags firmadas* |
| **git tag -a <tag-name> [commit-chksum]** | *crea tag para commit con dicho chksum* |
| Por defecto no se transfieren los tags, para subirlos al servidor: |  |
| **git push origin [tag-name]** | *una sola* |
| **git push origin --tags** | *Enviar todas* |
| Para usar GPG y firmar tags, hay que subir la clave pública al repositorio: |  |
| **gpg --list-keys** | *Coges la id pública* |
| **gpg -a --export <id> | git hash-object -w --stdin** | *Copia el SHA-1 devuelto* |
| **git tag -a maintainer-gpg-pub <SHA-1>** |  |
| **git push --tags** | *Comparte la clave con todos los usuarios* |
| **git show maintainer-gpg-pub | gpg --import** | *Cada usuario importa la clave así* |
| **git show <tag>** | *Devuelve más información sobre la etiqueta* |
| **git tag -d nombre\_tag** | *eliminar la etiqueta* |
| **git push origin :refs/tags/nombre\_tag** | *Eliminar la etiqueta del repositorio remoto.* |
|  |  |
| **git branch <nombre-rama>** | *crea rama. Puntero al commit actual* |
| **git checkout <nombre-rama>** | *cambiar a la rama especificada.* |
| **git checkout -b <nombre-rama>** | *crea y cambia de rama* |
| **git merge <rama>** | *Mezcla la rama actual con <rama>* |
| **git branch -d <rama>** | *elimina la rama* |
| **git push origin --delete <branchName>** | *Elimina una rama del servidor* |
| **git mergetool** | *Herramienta gráfica para resolver conflictos* |
| **git branch** | *lista ramas* |
| **git branch -v** | *lista ramas mostrando último commit* |
| **git branch --merged** | *lista ramas que han sido mezcladas con la actual. Si no tienen un \*, pueden borrarse, ya que significa que se han incorporado los cambios en la rama actual.* |
| **git branch --no-merged** | *lista ramas que no han sido incorporadas a la actual.* |
|  |  |
| **git fetch origin** | *Descarga el contenido del servidor* |
| **git push <remote> <branch>** | *Las ramas no se suben por defecto, has de subirlas explícitamente* |
| **git push <remote> <branch>:<nuevoNombre>** | *Igual que la de arriba, pero en el servidor se llama a la rama con nuevoNombre en lugar de branch* |
|  | Cuando se hace un git fetch que trae consigo nuevas ramas remotas, no se disponen de ellas localmente, solo se dispone de un puntero a la rama remota que no es editable. Para poder trabajar sobre esa rama, es necesario crearla Por ejemplo: |
| **git fetch origin** | *Tras ejecutarlo, notamos que se ha creado una rama nueva (rama\_nueva)* |
| **git checkout -b rama\_nueva origin/rama\_nueva** | *Crea una rama local a partir de la remota* |
| **git merge origin/nueva\_rama** | *Equivalente a la de arriba, pero sin establecer el tracking a la rama* |
| **git push [remotename] :[branch]** | *elimina una rama remota* |
| **git push [remotename] [localbranch]:[remotebranch]** | *La rama en el servidor tiene distinto nombre a la local* |
| **git checkout --track origin/rama** | *Equivalente a -b rama\_nueva origin/rama\_nueva* |
| **git chekout -b <nuevo\_nombre> origin/<rama>** | *Establece un nombre distinto para la rama local* |
| **git checkout <una rama>** |  |
| **git rebase master** | *aplica todos los cambios de <una rama> a master* |
| **git merge master** | *hay que hacer un merge de tipo fast forward* |
|  | Tenemos 3 ramas, master, client y server, en server y client tenemos varios commit y queremos mezclar client en master pero dejar server intacta: |
| **git rebase --onto master server client** | *adivina los patches del antecesor común de las ramas server y client y aplica los cambios a master.* |
| **git checkout master** |  |
| **git merge client** | *fast-forward. Client y master en el mismo snapshot* |
|  | Si se quiere aplicar también los cambios de server, basta con: |
| **git rebase master server** |  |
| **git checkout master** |  |
| **git merge server** |  |
| **git rebase [basebranch] [topicbranch]** | *sintaxis de rebase* |
| **git rebase -i** | *Rebase interactivo* |
| **git instawew** | *Muestra una interfaz web con los commits* |
| **git describe master** | *Solo funciona para tags creadas con -s ó -a* |
| **git archive master -- prefix=”project/’ | gzip > `git describe master`.tar.gz** |  |
| **git archive master -- prefix=”project/’ --format=zip | `git describe master`.zip** |  |
| **test/ export-ignore** | *Al crear el tarball no incluye el directorio test/* |
|  |  |
| **git shortlog --no-merges master --not <tag>** | *Recopila todos los commits desde <tag> y los agrupa por autor* |