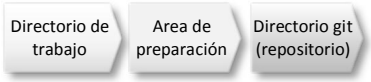
Hoja de Referencia **Git**

pablo d. sánchez

git help <command>
git <command> --help
man git-<command>



CONFIGURAR

git config --list	comprobar configuración
git config --global --edit	abre el fichero de configuración global
git config --global user.name "*username*"	usuario especificado en commits
git config --global user.email *email-address*	email especificado en commits
git config --global core.editor *gedit*	editor de texto
git config --global merge.tool *meld*	herramienta de diff (para conflictos de unión)
git config --global alias.*alias-name command*	crea alias para un comando de git: git config --global alias.glog "log --graph --oneline"

ficheros con patrones de nombres a ignorar:

.gitignore (puedo incluirlo en el repositorio con commit)
.git/info/exclude

git ls-files --other --ignored --exclude-standard	enumera archivos ignorados

git revert:
mantiene el grupo de cambios y usa un nuevo commit para deshacer

git reset:
elimina un grupo de cambios completamente

git status:
permite inspeccionar directorio de trabajo y área de preparación

git log:
sólo opera en la historia confirmada

Estado de los ficheros:

Modificado (modified)	en el directorio de trabajo
Preparado (staged)	marcado para ir al repositorio
Confirmado (commited)	en el repositorio

Conceptos:

master	nombre por defecto de la rama principal
origin	nombre por defecto del repositorio remoto
HEAD	último commit de la rama actual
HEAD^	penúltimo commit
HEAD^^	antepenúltimo commit
HEAD~5	cinco commits anteriores al último

BASICO

git init	crea repositorio local en el directorio actual
git init *path*	crea repositorio git vacío en el directorio especificado
git init --bare *path*	
git clone *url*	obtiene copia de repositorio existente (protocolos: git, http(s), ssh)
git add *filename*	lleva el archivo al área de preparación
git add *dir/*	añade todos los ficheros del directorio
git add --all	añade todos los ficheros nuevos o modificados (también **add .** o **add ***)
git add **.txt*	añade todos los ficheros txt del directorio actual
git add ".txt*"**	añade todos los ficheros txt del proyecto
git rm *filename*	borra el archivo y lo deja en el área de preparación
git rm --cached *filename*	retira el archivo del repositorio (queda en local)
git mv *file_from file_to*	renombra archivo y lo deja en el área de preparación
git commit	lleva los archivos del área de preparación al repositorio
git commit -m "*message*"	
git commit -a	atajo (evita hacer **git add** con ficheros modificados)
git commit --amend	añade al commit más reciente

COMPROBACIONES

git status	estado de los archivos en el área de trabajo y preparación
git diff	diferencias entre el área de trabajo y el repositorio (último commit)
git diff --staged	diferencias entre el área de preparación y el repositorio (último commit)
git diff HEAD	diferencia entre el último commit y el estado actual
git diff HEAD^..HEAD	diferencia entre el penúltimo commit y el último commit
git diff SHA..SHA	diferencia entre dos commits (por su ID de hash)
git diff --since=1.week.ago --until=1.minute.ago	diferencia basada en tiempo

HISTORICO - log

git log	historial de cambios de la rama actual
git log --follow *filename*	historial de cambios del archivo
git log -n *limit*	limita historial al valor especificado
git log --oneline	resume cada commit en una línea
git log --stat	incluye información de los ficheros que han cambiado
git log *file*	muestra commits que incluyan el fichero especificado
git log -p	muestra diff en el historial de cambios
git log --graph --decorate	
git log --grep="*pattern*"	busca commits cuyo mensaje coincide con lo especificado
git log --author="*pattern*"	busca commits de un autor
git log *since..until*	busca en función de los límites especificados

RAMAS

git branch	listado de ramas
git branch -r	listado de ramas remotas
git branch -a	listado de ramas locales y remotas
git branch *branch_name*	crear la rama especificada
git branch -m *branch_name*	renombra la rama actual a *branch-name*
git branch -d *branch_name*	elimina la rama especificada
git branch -D *branch_name*	fuerza eliminación de la rama especificada
git checkout *branch_name*	cambiar a la rama especificada
git checkout -b *branch_name*	crea la rama especificada y salta a ella (mueve HEAD)
git merge *branch_name*	fusiona la rama especificada con la rama actual
git merge --no-ff *branch_name*	fusiona con confirmación
git diff *branch1..branch2*	muestra diferencias entre dos ramas

DESHACER CAMBIOS

git checkout master	vuelve a la rama master (estado "actual" del proyecto)
git checkout ***commit file***	recupera la versión anterior de un archivo convierte el *file* del directorio de trabajo en una copia del que hay en el *commit*
git checkout HEAD ***file***	recupera la versión más reciente
git checkout ***commit***	actualiza los ficheros del área de trabajo para coincidir con el *commit*
git checkout – ***file***	deshace cambios y deja como en el último commit
git revert ***commit***	deshace los cambios introducidos en *commit* (genera un nuevo commit en la rama actual)
git reset	resetea el área de preparación para coincidir con el commit más reciente
git reset ***file***	elimina el fichero del área de preparación
git reset ***commit***	mueve la rama actual a *commit* y resetea el área de preparación
git reset ***commit***	mueve la rama actual a *commit* y resetea el área de preparación
git reset --hard	resetea el área de preparación y el directorio de trabajo
git reset --hard ***commit***	mueve rama actual a *commit* reseteando áreas de preparación y trabajo
git reset --hard HEAD~2	deshace los dos últimos commits
git reset --soft HEAD^	deshace el último commit y lo deja en el área de preparación
git clean –n	muestra que archivos van a eliminarse
git clean –f	elimina archivos sin seguimiento del directorio de trabajo
git clean –f ***path***	elimina archivos sin seguimiento de la ruta especificada
git clean –df	elimina archivos y directorios sin seguimiento del directorio actual
git clean –xf	elimina archivos sin seguimiento e ignorados del directorio actual

TAGS

git tag	lista los tags
git checkout ***tagname***	mueve al commit marcado con *tagname*
git tag –a ***tagname*** **–m "***msg***"**	añade nuevo tag
git push --tags	sube tags a remoto

REPOSITORIO REMOTO

git remote	lista conexiones remotas
git remote –v	lista conexiones remotas con la URL de conexión
git remote show origin	da información de ramas remotas
git remote add ***name url***	crea conexión nueva a un repositorio remoto
git remote rm ***name***	elimina conexión
git remote rename ***oldname newname***	renombra conexión remota
git remote prune origin	limpia las ramas remotas borradas
git push origin :*branch_name*	borra rama remota
git fetch ***name***	recupera todas las ramas del repositorio (sincroniza sin merge)
git fetch ***name branch_name***	recupera la rama especificada
git pull ***name***	recupera la copia remota y la fusiona con la local (= fetch + merge) sincroniza repositorio local y remoto (crea rama origin/master)
git pull --rebase ***name***	usa rebase en lugar de merge
git merge origin/master	une rama origin/master con master en nuevo commit
git push ***name branch_name***	envía la rama especificada a remoto git push origin master
git push –u ***name branch_name***	–u para recordar nombre y rama en siguientes push
git push ***name*** **--force**	fuerza el envío (incluso si resulta en una fusión)
git push ***name*** **--all**	envía todas las ramas locales
git push ***name*** **--tags**	envía todas las etiquetas locales

https://bitbucket.org/
https://github.com/

Buenas prácticas:
Utilizar 4 tipos de ramas:
– **master**: rama principal. Contiene el repositorio que se encuentra publicado en producción, por lo que debe estar siempre estable.
– **development**: rama sacada de master. Es la rama de integración, todas las nuevas funcionalidades se deben integrar en esta rama. Merge sobre master.
– **features**: rama sacada de development. Nueva funcionalidad. Merge sobre development.
– **hotfix**: ramas sacadas de master. Son bugs que surgen en producción, por lo que se deben arreglar y publicar de forma urgente. Merge sobre master y merge de master sobre development.