Git Panorámico

Recuerda: git "nombre comando" --help

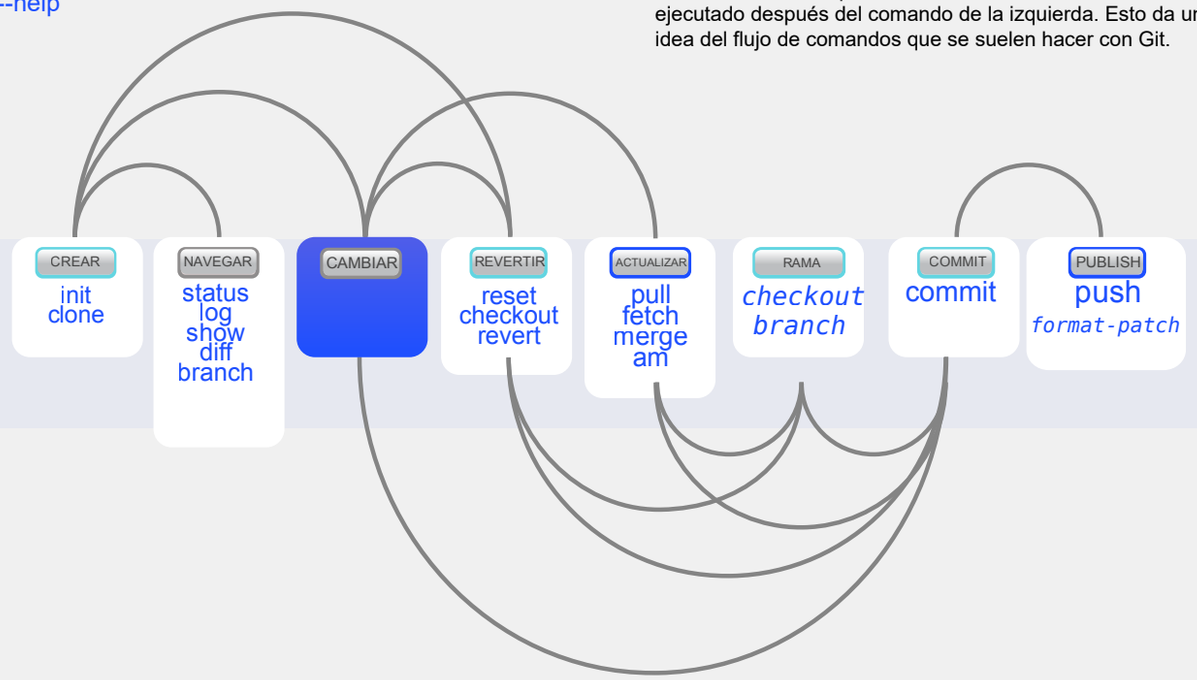
Documentación completa:
https://git-scm.com/docs

Configuración Git local en <directorio-raíz>/.git/config
Configuración Git Global almacenada en $HOME/.gitconfig
Configuración Git Sistema en /etc/gitconfig

git config --help

Secuencia Comandos

Las curvas indican que el comando de la derecha suele ser ejecutado después del comando de la izquierda. Esto da una idea del flujo de comandos que se suelen hacer con Git.



Crear

De datos existentes
```
cd ~/proyectos/miproyecto
git init
git add .
```

De Repo existente
```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://you@host.org/proj.git
```

Mostrar

Archivos cambiados en directorio trabajo
```
git status
```

Cambios en los archivos rastreados
```
git diff
```

Que cambió entre $ID1 and $ID2
```
git diff $id1  $id2
```

Historial de cambios
```
git log
```

Historia de cambios para archivos con diffs
```
git log -p $file $dir/ec/tory/
```

Quién cambió qué y cuándo en un archivo
```
git blame $file
```

Un commit identificado por $ID
```
git show $id
```

Un archivo de un $ID específico
```
git show $id:$file
```

Todas las ramas locales
```
git branch
```
(un '*' asterisco marca la rama actual)

Notaciones usadas

$id : representa bien un id de commit, una rama o un nombre de tag
$file : cualquier nombre de archivo
$branch : cualquier nombre de rama

Conceptos

Lo Básico de Git

```
master   : rama desarrollo predeterminada
origin   : repositorio de subida predeterminado
HEAD     : rama actual
HEAD^    : padre of HEAD
HEAD~4   : el tatarabuelo de HEAD
```

Revertir

Volver al último estado committed
```
git reset --hard
```
⚠ No podemos deshacer un hard reset

Revertir el último commit
```
git revert HEAD
```
Crea un nuevo commit

Revertir commit específico
```
git revert $id
```
Crea un nuevo commit

Arreglar último commit
```
git commit -a --amend
```
(después de editar los archivos erróneos)

Revisar la versión $id de un archivo
```
git checkout $id $file
```

Rama

Cambiar a rama $id
```
git checkout $id
```

Unir rama1 con rama2
```
git checkout rama2
git merge rama1
```

Crear rama llamada $rama basada en el HEAD
```
git branch $rama
```

Crear rama $nueva_rama basada en rama $otra y cambiar a ella
```
git checkout -b $nueva_rama $otra
```

Borrar rama $rama
```
git branch -d $rama
```

Actualizar

Obtener últimos cambios desde origin (subido)
```
git fetch
```
(Pero esto no los une)

Recuperar últimos cambios desde origin (subido)
```
git pull
```
(hace un fetch seguido por un merge)

Aplicar un patch enviado por alguien
```
git am -3 patch.mbox
```
(en caso de un conflicto resolver y usar)
```
git am --resolved
```

Publicar

Commit todos tus cambios locales
```
git commit -a
```

Preparar un patch para otros other desarrolladores
```
git format-patch origin
```

Enviar cambios a origin (subida)
```
git push
```

Marcar una versión / milestone
```
git tag v1.0
```

Comandos Útiles

Encontrar regresiones
```
git bisect start          (para empezar)
git bisect good $id       ($id es la última versión correcta)
git bisect bad $id        ($id es una versión incorrecta)

git bisect bad/good       (para marcarlo como malo o bueno)
git bisect visualize      (para lanzar gitk y marcarlo)
git bisect reset          (una vez hecho)
```

Verificar errores y limpiar repositorio
```
git fsck
git gc --prune
```

Buscar directorio de trabajo para foo()
```
git grep "foo()"
```

Resolver Conflictos Merge

Para ver los conflictos merge
```
git diff          (conflicto diff completo)
git diff --base  $file   (contra archivo base)
git diff --ours  $file   (contra tus cambios)
git diff --theirs $file  (contra otroscambios)
```

Para descartar parche conflictivo
```
git reset --hard
git rebase --skip
```

Después de resolver conflictos, unir con
```
git add $conflicting_file   (hacer para todos los archivos resueltos)
git rebase --continue
```