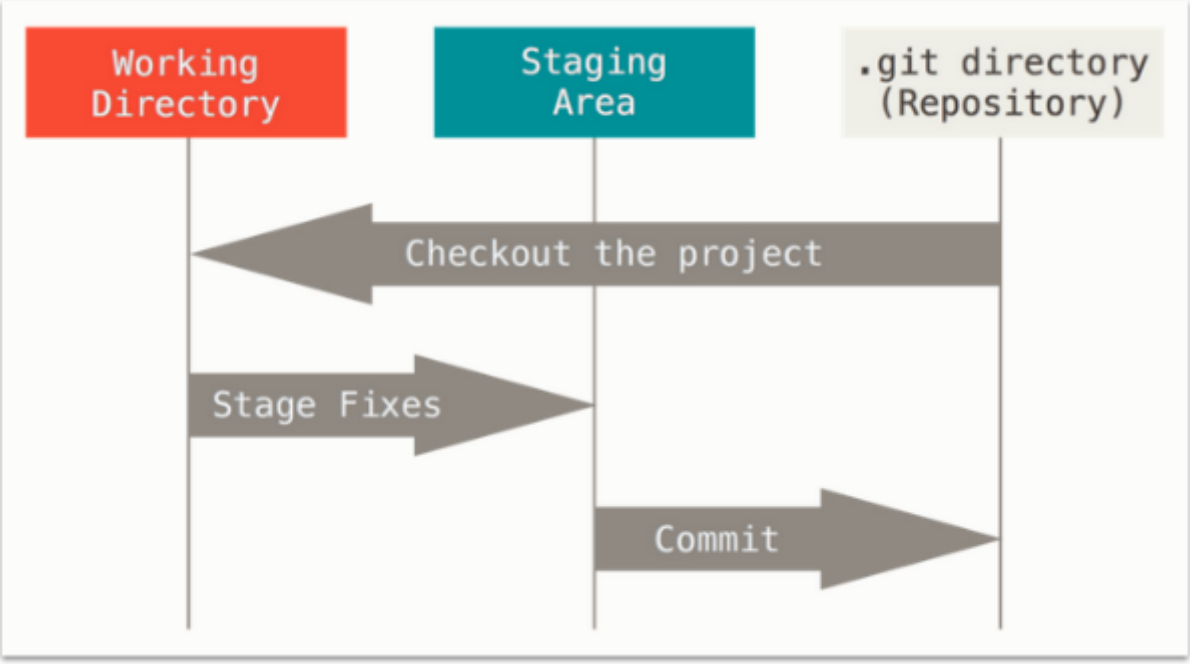
git

info

- https://git-scm.com/book/es/v2

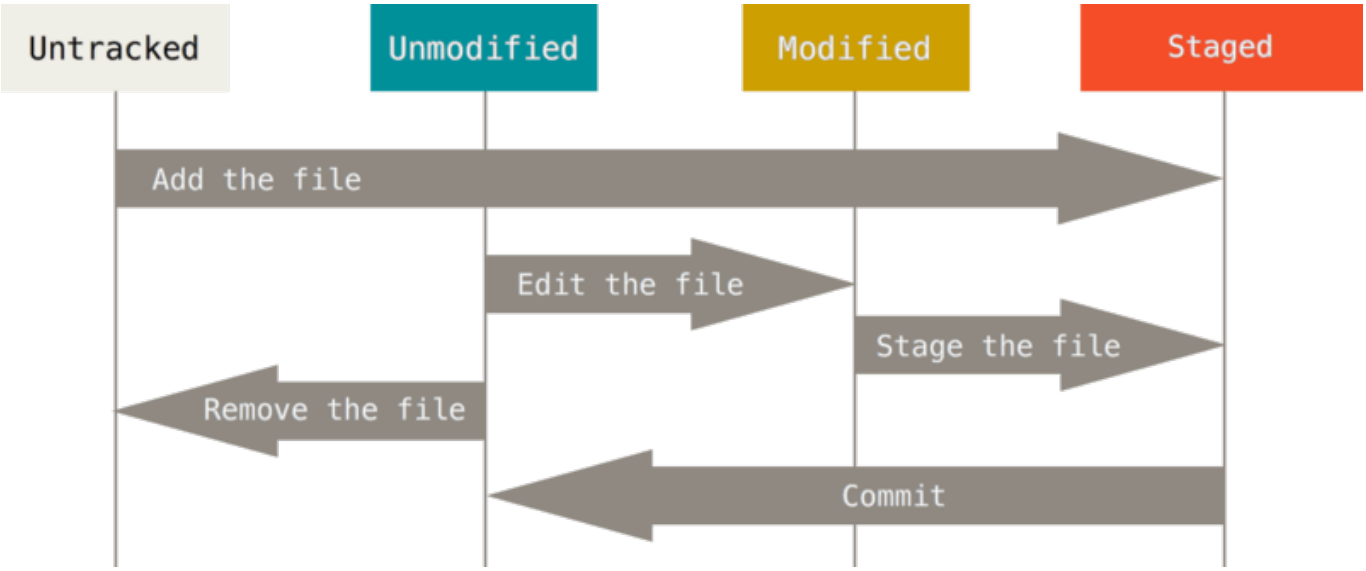
metodología de trabajo en Git

estados



- Working directory: nuestro directorio de trabajo, que puede ser cualquiera de los diferentes **commits** que hemos ido realizando a lo largo del tiempo, aunque se suele trabajar con el **<HEAD>**
- Staging area: es el índice de ficheros de los que se guardará una «copia» o «snapshot» en el próximo commit
- Repository: es la carpeta donde residen los metadatos y los diferentes «snapshots» que tenemos de nuestro proyecto

cambios en ficheros



estados de un fichero:

- sin seguimiento (*untracked*)
- sin modificar (*unmodified*)
- modificado (*modified*)
- preparado (*staged*) –> *cached* como sinónimo

comandos

básicos

- empezar repositorio local:

```
git init
```

- añadir ficheros/directorios a *stage area*(si modificamos el fichero después de haberlo añadido al *stage*, la versión que se guardará con el **commit** será la primera modificación. La segunda modificación quedará pendiente de ser añadida (add) y envíada (commit):

```
git add .
```

- añadir todos los ficheros:

```
git add -A
```

- consultar estado:

```
git status
```

```
git status -s // abreviado
```

- crear un commit (envío/paquete/snapshot):

```
git commint -m "Mensaje descriptivo"
```

- hacer un commit de los ficheros preparados-modificados sin tener que añadirlos explicitamente:

```
git commit -a -m "Mensaje descriptivo"
```

- forzar cambio de usuario en un commit en concreto:

```
git -c “user.name=matebcn” commit
```

- setup sincronización entre repositorio local con repositorio en la nube:

```
git remote add origin <dirección repositorio Git>
```

- enviar commit a repositorio nube:

```
git push -u origin master
```

- clonar repositorio nube existente en local (se establece ORIGIN de manera automática):

```
git clone <dirección repositorio git>
```

con más detalle

puede ser confuso saber que estás comparando dependiendo del estado del fichero (sin seguimiento - sin modificar - modificado - preparado)

- mostrar los cambios con más detalle (en working tree):

```
git diff
```

- (en *stage area*):

```
git diff --staged
```

con archivos

- mover archivo dentro del working tree:

```
git mv <archivo> <nuevo_nombre_archivo>
```

- –> es equivalente a eliminar y dar de alta, git detecta que es el mismo fichero en diferente ubicación:

```
mv <path>/<archivo> <pah2>/<archivo>
git rm <path>/<archivo>
git add <path2>/<archivo>
```

- eliminar fichero:

```
rm <archivo>
git rm <archivo>
```

- eliminar del área de trabajo (pero no del *working tree*):

```
git rm --cached <archivo>
```

- para hacer que ciertos archivos (o patrones de archivos) no se registren, se puede crear un fichero `.gitignore`

  .gitignore

```
# ignora los archivos terminados en .a
*.a

# pero no lib.a, aun cuando había ignorado los archivos terminados en .a
en la linea anterior
!lib.a

# ignora unicamente el archivo TODO de la raiz, no subdir/TODO
/TODO

# ignora todos los archivos del directorio build/
build/

# ignora doc/notes.txt, pero este no doc/server/arch.txt
doc/*.txt

# ignora todos los archivos .txt el directorio doc/
doc/**/*.txt
```

viajes en el tiempo

- deshace el ADD de un fichero:

```
git reset HEAD <fichero>
```

- deja el fichero como en el último commit:

```
git checkout -- <fichero>
```

- recuperar un snapshot determinado (el proyecto vuelve a ese momento en el tiempo):

```
git checkout <codigo_commit>
```

- recuperar el último snapshot:

```
git checkout master
```

log

- consultar últimos cambios:

```
git log
```

-
```
git log --oneline
```

configuración

- prioridad de los ficheros de configuración (más alta, más prioridad):
    1. system: /etc/gitconfig

2. global: ~/.gitconfig
3. local: .git/config

- establecer globalmente el usuario de trabajo:

```
git config [--global] user.name “mate”
```

- establecer globalmente el e-mail de trabajo:

```
git config [--global] user.email miemail@midominio.algo
```

- muestra la información almacenada y la ubicación del fichero:

```
git config --show-origin --get-all user.name
```

- comprobando la configuración:

```
git config --list
```

tags

alias a los commit, en lugar de su código

- aplica el <tag> al actual:

```
git tag <tag>
```

- `git tag <tag> <codigo_commit>`
- listado de tags:

```
git tag
```

- usar los tags para cambiar de *snapshot*:

```
git checkout <tag>
```

alias

- `git config --global alias.cm ‘-c “user.name=matebcn” commit’`

/via: https://git-scm.com/book/en/v2/Git-Basics-Git-Aliases

RESET

> atención a estos comandos, podrías perderlo todo

- no toca la working area:

```
git reset --soft
```

- borra staging area pero no working area:

```
git reset --mixed
```

- CUIDADO! lo borra todo:

```
git reset --hard
```

PTE

pendiente, wiki

> pendiente de colocar de una manera más legible

```
push repositorios remotos
repositorio centralizado
al hacer un clone se establece como remoto
git remote add origin <direccion o directorio> → origin se suele usar en github
para el repositorio
git push origin master → enviamos a origin la master

actualizando repositorios remotos
git branch -all → muestra todas las ramas, incluidas las remotas
git pull origin master → descarga los cambios de origin a master
git pull --rebase → preserva tus cambios y que no sean machacados por el PULL
```

```
ramas
linea de desarrollo de commit uno detrás de otro
ramas: trabajar en características diferentes
git branch “caracteristica” → crea rama
git branch → muestra las ramas
git checkout “caracteristica” → moverte a la rama
git checkout -g “caracteristica” → crea la rama y nos movemos a ella
git branck -d “caracteristica” → elimina rama
fusión de ramas:
git log --oneline --decorate
git checkout master
git merge “caracteristica”
bloqueo pesimista -> bloqueas el archivo
bloqueo optimista -> se supone que hay trabajo repartido, no debería haber
conflicto.
si no se toca la misma línea, GIT puede llegar a resolver. si no, intervención
manual
```

eclipse + bitbucket

- instalar nuevo software (repositorio) : `http://download.eclipse.org/egit/updates`

- Seleccionar **Git Integration for Eclipse** y **Java implementation of Git**
- para acceder, cambiar perspectiva a Git
- también se puede importar un proyecto desde un repositorio GIT (desde Archivo - Import)

- una vez configurado:
    - para importar a GIT, creamos un nuevo repositorio local a partir de nuestro proyecto y lo enviamos al repositorio creado

/via: https://crunchify.com/how-to-configure-bitbucket-git-repository-in-you-eclipse/

From:
https://miguelangel.torresegea.es/wiki/ - **miguel angel torres egea**

Permanent link:
https://miguelangel.torresegea.es/wiki/development:git:start?rev=1528406594

Last update: **07/06/2018 14:23**