4.C. GIT

5. Resumen de comandos GIT.

A continuación se muestran algunos de los **comandos más utilizados en GIT**.

- **Init**
  Crea un nuevo repositorio en el directorio inicialmente vacío. Se almacena en el directorio oculto .git.

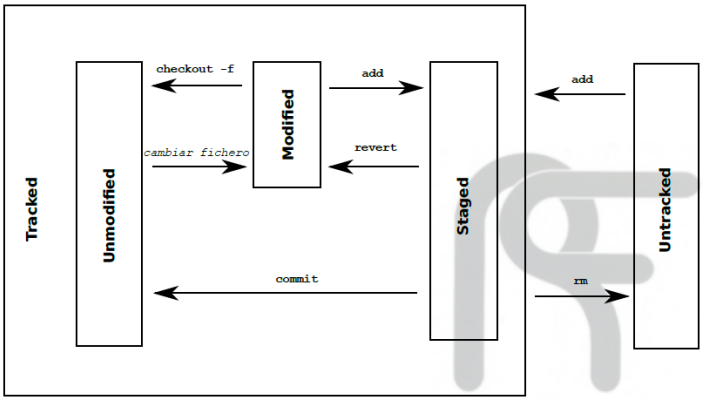
```
alvaro@debian8-64-alvarogonzalez:~/aplicacion-web$ git init
Initialized empty Git repository in /home/alvaro/aplicacion-web/.git/
```

- **clone**
  Crea un nuevo repositorio, copia de uno ya existente. El repositorio replicado se indica mediante su *URL*. La copia de trabajo inicial será la de la rama *MASTER*.

```
alvaro@debian8-64-alvarogonzalez:~/aplicacion-web$ git clone \
     https://github.com/alvarogonzalezsotillo/aplicacion-php.git
Cloning into 'aplicacion-php'...
remote: Counting objects: 50, done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 50 (delta 21), reused 39 (delta 12), pack-reused 0
Unpacking objects: 100% (50/50), done.
Checking connectivity... done.
```

- **add**, **rm**, **mv**
  Permite identificar los ficheros del área de trabajo que van a ser incluidos en el repositorio. Para seleccionar ficheros se usa el comando add, para deseleccionarlos el comando a utilizar será rm. El borrado de ficheros no tiene carácter retroactivo sobre las versiones anteriores.

- **status**
  Imprime un resumen del estado de los ficheros de la copia de trabajo. Los ficheros se pueden encontrar en los siguientes estados:
  - **Untracked**: no guardados en el repositorio. GIT ignora el fichero.
  - **Unmodified**: igual que en la rama activa del repositorio.
  - **Modified**: con cambios respecto al repositorio, pero que no está previsto que sean guardados al hacer *commit*.
  - **Staged**: el fichero está en el index. Será parte del próximo changeset que se añadirá al repositorio.



- **commit**
  Crea una nueva versión en el repositorio local, incluyendo los cambios en estado *Staged* procedentes del área de trabajo.

  Los ficheros pasan a estar seleccionados como parte de las nuevas versiones con el comando *add*. El comando *git commit -a* incluye en el index todos los ficheros del área de trabajo diferentes a los disponibles en el repositorio local, a continuación aplica un commit creando una nueva versión.

```
alvaro@alvaro-vaio:~/aplicacion-web$ git commit -a
[master 906fa20] Cambiados varios permisos de ejecución
 5 files changed, 411 insertions(+), 2 deletions(-)
 create mode 100644 09/estados-fichero-git.svg
 mode change 100755 => 100644 borrar-temporales-latex.sh
 mode change 100755 => 100644 generar-pdf.sh
 mode change 100755 => 100644 generar-pdfs.sh
 mode change 100755 => 100644 publicar-pdfs.sh
```

- **push**
  Sube las versiones del repositorio local a un repositorio remoto. El repositorio remoto puede establecerse con:
  - git clone (en este caso se denomina origin).
  - git remote add

  El repositorio debe tener todas las versiones del repositorio remoto, en otro caso, deberá realizarse un git pull previo.
  Aquí tienes una referencia al comando git remote.

```
alvaro@debian8-64-alvarogonzalez:~/aplicacion-web$ git push
Password for 'https://alvarogonzalezsotillo@bitbucket.org':
To https://alvarogonzalezsotillo@bitbucket.org/alvarogonzalezsotillo/aplicacion-web.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to
'https://alvarogonzalezsotillo@bitbucket.org/alvarogonzalezsotillo/aplicacion-web.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

  Observar en la figura que el comando *push* ha dado error, el motivo es que el repositorio remoto tiene recogidos cambios de otros colaboradores que no han sido volcados al repositorio local. Si no se combinan ambos, podría producirse pérdida de información, por lo que sugiere hacer un *pull* previo al *push*.

- **checkout**
  Extrae versiones del repositorio local a la copia de trabajo, la acción se puede realizar sobre diferentes elementos del repositorio:
  - **Un fichero**: git checkout fichero.
  - **Una versión**: git checkout hashdeversión.
  - **Un branch**: git checkout branch.

  También permite crear nuevas ramas, con *git checkout -b nombrerama*.

- **branch**
  Lista, crea o elimina ramas en un repositorio.

- **Otros comandos de interés son:**
  - **git log**: historia de cambios de un repositorio.
  - **git diff**: Muestra los cambios de los ficheros en estado modified.
  - **git merge**: fusión de dos ramas.
  - **git gui**: interfaz gráfica para utilizar GIT.