Antonio Lafont

Práctica 1- PAT

```
@AntonioLafont ➜/workspaces/p1-fork (main) $ git clone https://github.com/gitt-3-pat/p1
Cloning into 'p1'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 5 (from 1)
Receiving objects: 100% (6/6), done.
@AntonioLafont ➜/workspaces/p1-fork (main) $
```

Git clone crea una copia de un repositorio remoto en nuestro ordenador. Lo utilizamos para trabajar en un proyecto ya existente.

```
@AntonioLafont ➜/workspaces/p1-fork (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        p1/

nothing added to commit but untracked files present (use "git add" to track)
@AntonioLafont ➜/workspaces/p1-fork (main) $
```

Git status muestra el estado del repositorio local. Nos informa sobre en que rama nos encontramos, que archivos han sido alterados, que archivos están listos para commit (han sido añadidos/están en staging), y que archivos estan “untracked” (no seguidos por git). Lo utilizamos para evitar errores antes de hacer un commit, como podría ser no añadir todos los archivos deseados.

```
@AntonioLafont ➜/workspaces/p1-fork (main) $ git add .
warning: adding embedded git repository: p1
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> p1
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached p1
hint:
hint: See "git help submodule" for more information.
hint: Disable this message with "git config set advice.addEmbeddedRepo false"
@AntonioLafont ➜/workspaces/p1-fork (main) $
```

Git add añade cambios al git para que estén preparados para el siguiente commit, es decir, los pasa a staging. Añadiendo "." estamos indicando que queremos añadir todo el contenido de la carpeta en la que nos encontramos. Como git no guarda los cambios automáticamente, tenemos que utilizar git add . para indicar cuales queremos guardar antes de hacer commit.

```
@AntonioLafont ➜/workspaces/p1-fork (main) $ git commit -m "Ejemplo mensaje"
[main f4a2460] Ejemplo mensaje
 1 file changed, 1 insertion(+)
 create mode 160000 p1
@AntonioLafont ➜/workspaces/p1-fork (main) $
```

Git commit sirve para guardar una nueva versión del proyecto con todos los cambios que estaban en staging. -m "texto" añade el mensaje que hayamos escrito entre comillas. Este mensaje debería de ser una explicación breve de los cambios añadidos en el commit. Utilizamos los commits para tener un registro ordenado de las versiones del proyecto, facilitando enormemente regresar a versiones anteriores, trabajar con otras personas, y tener documentado en qué momento se añadió cada cambio.

```
@AntonioLafont ➜/workspaces/p1-fork (main) $ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 281 bytes | 281.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/AntonioLafont/p1-fork
   07720b5..f4a2460  main -> main
@AntonioLafont ➜/workspaces/p1-fork (main) $
```

Git push envía todos los commits hechos en local al repositorio remoto. Lo utilizamos para que el resto del equipo con el que trabajamos tenga acceso a nuestras versiones. Añadiendo origin main aclaramos en que repositorio y rama queremos que se suban los cambios, pero si no lo añadimos los commits se mandan a la rama y repositorio usados por defecto. Normalmente estos también son origin y main, y ambos comandos tienen el mismo efecto.

```
@AntonioLafont ➜/workspaces/p1-fork (main) $ git checkout -b feature/1
Switched to a new branch 'feature/1'
@AntonioLafont ➜/workspaces/p1-fork (feature/1) $
```

Git checkout cambia de rama. -b feature/1 crea una rama nueva con el nombre de “feature/1”, así que el comando entero crea una nueva rama y nos cambia a ella. Trabajar con ramas nos permite añadir cambios sin alterar la rama principal, una buena práctica para no causar problemas al resto del equipo.

```
@AntonioLafont ➜/workspaces/p1-fork (feature/1) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@AntonioLafont ➜/workspaces/p1-fork (main) $
```

Git checkout main nos devuelve a la rama principal.