COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

PROGRAMACIÓN DE APLICACIONES TELEMÁTICAS

PRÁCTICA 1: ENTORNO DE DESARROLLO

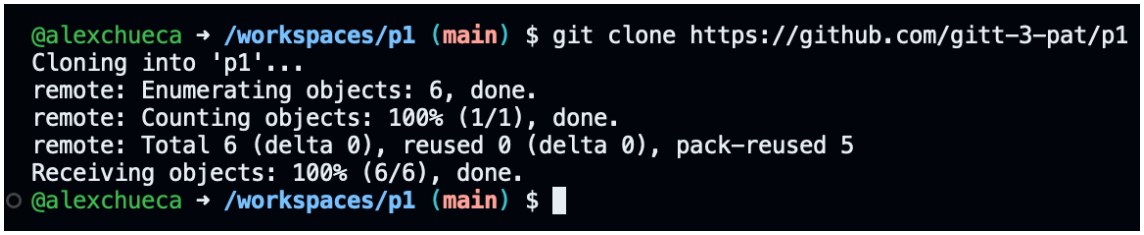
Alejandro Chueca Manzanero

1. INTRODUCCIÓN

En esta práctica, vamos a preparar el entorno de trabajo de cara al resto del trabajo de la asignatura. En este primer informe, vamos a ver los distintos comandos de Github que nos van a ayudar a realizar las prácticas y seguir las clases.

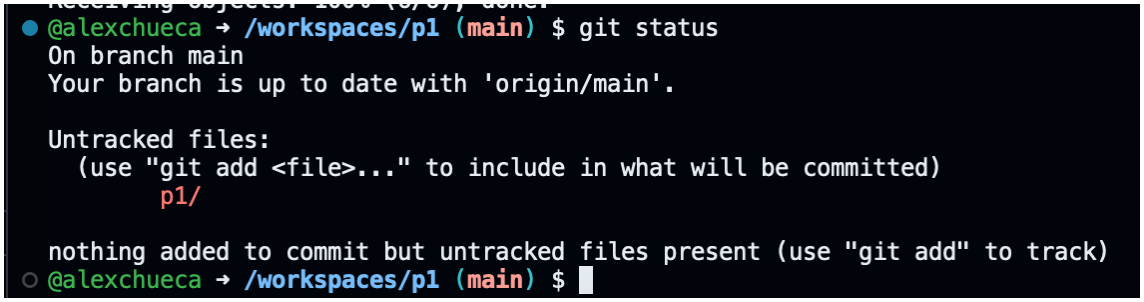
2. COMANDOS

git clone



Con el comando *git clone* creamos una copia de un repositorio público en nuestro repositorio. Entre las utilidades que tiene, se encuentra la posibilidad de trabajar en los archivos del repositorio que copiamos sin alterar lo que está publicado previamente. De esta forma, damos la opción de que otra persona acceda a los archivos previos sin ver lo que nosotros hemos modificado.

git status



El commando *git status* indica el estado del repositorio. En la imagen se ve que nos saca en el terminal la rama en la que estamos, el estado de actualización respecto a 'origin/main' y los archivos que hay en esa rama del repositorio. Nos puede servir para localizar rápidamente archivos, o bien saber si estamos en el sitio correcto.

git add

```
@alexchueca ➜ /workspaces/p1 (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.
@alexchueca ➜ /workspaces/p1 (main) $
```

Este commando *git add* crea un segundo repositorio dentro del repositorio “principal”. Puede servir para guardar archivos en ese repositorio que no se muestran en el repositorio principal y público, información que sabemos gracias a las líneas de hint que se muestran.

git commit

```
@alexchueca ➜ /workspaces/p1 (main) $ git commit -m "TU MENSAJE"
[main 3ff966d] TU MENSAJE
 1 file changed, 1 insertion(+)
 create mode 160000 p1
@alexchueca ➜ /workspaces/p1 (main) $
```

El comando *git commit*, en este caso, nos ha creado un archivo en la rama main que se llama “TU MENSAJE”, y lo guarda. De esta forma, guardamos los cambios en el repositorio, pero necesitamos el siguiente comando para hacerlos públicos

git push

```
@alexchueca ➜ /workspaces/p1 (main) $ git push
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), 284 bytes | 284.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/alexchueca/p1
   268010e..3ff966d  main -> main
@alexchueca ➜ /workspaces/p1 (main) $
```

Después del comando commit, con el comando *git push* subimos los archivos al repositorio público, permitiendo a todo aquel que lo requiera su visualización y/o edición.

git checkout

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

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

Por último, el comando *git checkout* nos mueve entre las ramas del repositorio, por si estuviéramos trabajando en otras ramas distintas.