El primer paso es hacer un fork (copiar el repositorio dado para poder realizar cambios sin modificarlo).

Situado en el fork creado se ha procedido a aplicar los diferentes comandos:

Clone:

Este comando copia el repositorio en cuestión, en un directorio dado o en una máquina virtual. En este caso lo está copiando en un directorio (p1) para la práctica 1.

Status:

Informa acerca de si los cambios en tu rama (en tu trabajo local) han sido o no han sido "publicados" en el repositorio remoto.

```
@Ozuco →/workspaces/p1 (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)
        src/
```

Aquí gracias al comando status puedo ver que los cambios (la creación del directorio src/) no han sido publicados/actualizados en el repositorio remoto.

Add, Commit y Push:

Todos ellos tienen en común que son necesarios para lanzar/actualizar los cambios realizados desde el repositorio local al repositorio remoto. No obstante, esos cambios no se lanzan directamente, sino que se realizan "poco a poco".

Así, el primer paso es hacer un git "add" de los cambios (en el staging area).

El siguiente paso que es el de confirmarlos con el comando "commit". Este paso guarda los cambios en el repositorio local. (Usando -m se puede enviar un mensaje/comentario aclarativo).

Por último, el comando "push" coge los cambios confirmados localmente con el comando anterior y los envía al repositorio remoto actualizando la rama remota.

Un ejemplo de implementación de dichos comandos es el siguiente:

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

nothing added to commit but untracked files present (use "git add" to track)
@Ozuco →/workspaces/p1/src (main) $ git add .
@Ozuco →/workspaces/p1/src (main) $ git commit -m "archivo HTML"
[main 9ab6ea1] archivo HTML
 1 file changed, 5 insertions(+)
 create mode 100644 src/index.html
@Ozuco →/workspaces/p1/src (main) $ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 408 bytes | 408.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Ozuco/p1
   07720b5..9ab6ea1  main -> main
```

Checkout:

El comando “checkout” tiene varios usos diferentes, aunque principalmente se usa para cambiarse de rama cuando sea necesario.

Con checkout -b se puede crear una nueva rama a la que cambiar para realizar cambios y así, no hacerlo directamente en la rama main.

Checkout (a secas) es usado para restaurar archivos en un estado anterior o para cambiar a una rama especificada.

Por último, el comando pull se utiliza para recuperar cambios desde el repositorio remoto e implementarlos en la rama actual. Como se ha realizado en este ejemplo:

```
@Ozuco →/workspaces/p1 (main) $ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
@Ozuco →/workspaces/p1 (main) $ git pull origin main
From https://github.com/Ozuco/p1
 * branch            main       -> FETCH_HEAD
Already up to date.
```