Ejercicios de comandos 5

Recuperación del estado de un archivo existente en un commit anterior

Objetivos:
- Ejecución de *commits*.
- Creación de un repositorio remoto.
- Vinculación de un repositorio local con uno remoto.
- Subida de ramas del repositorio local al remoto.

Pre-requisitos:
- Trabajar con un usuario de nombre *alumno*.
- Tener instalado el entorno Git.
- Haber realizado los ejercicios de comandos 2 y 3.

Comandos Git utilizados:
- `git add`
- `git commit`
- `git checkout`
- `git status`
- `git diff`
- `git rm`
- `git reset`
- `git commit`
- `git log`
- `git rev-list`

Comandos Linux utilizados:
- `nano`
- `rm`
- `ls`

1) Posicionarse dentro del repositorio en el que se va a trabajar.
```
cd /home/alumno/Documentos/gitRepos/repo_uno
```

2) Crear 3 archivos: `arch_01`, `arch_02`, `arch_03`
```
nano arch_01
nano arch_02
nano arch_03
```

3) Añadirlos al SA.
```
git add .
```

4) Confirmar los cambios:
```
git commit -m "Creación de arch_01, arch_02 y
arch_03"
```

5) Se borra el archivo llamado `arch_02`.
```
rm arch_02
```

Comprobar que ha sido eliminado.

```
ls -la
```

6) Descartar (deshacer) el cambio (la eliminación) en el directorio de trabajo.

```
git checkout -- arch_02
```

Comprobar que se ha recuperado.

```
ls -la
```

7) Volver a eliminarlo (esta va a ser la buena).

```
rm arch_02
```

Comprobar el estado (aparece `deleted: arch_02`).

```
git status
```

8) Comprobar las diferencias en los archivos modificados.

```
git diff
```

El resultado es:

```
diff --git a/arch_02 b/arch_02
deleted file mode 100644
index dc2f5f5..0000000
--- a/arch_02
+++ /dev/null
@@ -1 +0,0 @@
-Este es el archivo 2
```

Se indica que el archivo llamado `arch_02` ha sido alterado (borrado, pasa a estar en `/dev/null`), y se muestra el contenido que tenía antes (estado `a`) del borrado.

9) Actualizar que se confirmará el borrado.

```
git rm arch_02
```

Comprobar el estado:

```
git status
```

10) Para sacar el archivo del SA (*unstage*):

```
git reset HEAD arch_02
```

Comprobar el estado:

```
git status
```

11) Volver a actualizar que se confirmará el borrado (esta va a ser la buena).

```
git rm arch_02
```

Comprobar el estado:

```
git status
```

12) Confirmar los cambios:

```
git commit -m "Eliminación de archivo arch_02"
```

Comprobar el estado.

```
git status
```

13) Modificar el archivo llamado `arch_01`.

```
vi arch_01
```

14) Comprobar los cambios producidos.

```
git diff
```

El resultado es:

```
diff --git a/arch_01 b/arch_01
index 220853c..9b50d32 100644
--- a/arch_01
+++ b/arch_01
@@ -1 +1 @@
-Este es el archivo 1
+Este es el archivo 1 modificado
```

Se indica que el archivo llamado arch_01 ha sido alterado (modificado) y se muestra el contenido que tenía antes (estado `a`) y después (estado `b`) de la modificación.

15) Se añade el archivo llamado `arch_01` al SA.

```
git add arch_01
```

16) Se confirman los cambios:

```
git commit -m "Modificación del archive arch_01"
```

17) Se listan todos los *commits* llevados a cabo.

```
git log
```

18) Ahora se quiere recuperar el archivo `arch_02` de un *commit* pasado. Para ello, hay que determinar en qué *commit* se borró.

```
git rev-list -n 1 HEAD -- arch_02
```

El resultado es el id del ultimo commit en el que se alteró su estado (se borró).

```
5393c47dc157397b1100dcbffc6155da7a7395dc
```

19) Una vez que se sabe el id del *commit*, se recupera el archivo.

```
git checkout 5393c47^ -- arch_02
```

Comprobar el estado (el archivo `arch_02` aparecerá como `new file: arch_02`). Se ofrece la posibilidad de sacarlo del *staging area* ejecutando `git reset HEAD arch_02`.

Se comprueba el estado.

```
git status
```

Se hace un listado.

```
ls -la
```

20) Se podrían haber realizado ambos comandos en uno solo, combinándolos.

```
git checkout $(git rev-list -n 1 HEAD -- arch_02)^
     -- arch_02
```