El Archivo Zpracticagit.bat al ejecutarlo hace todos los comandos y te añade el resultado a un archivo llamado practica.txt, si lo ejecutas se para un segundo y tienes que hacer ctrl+z y enter para que continue ejecutándose

Práctica Git

Jesus Lorenzo Limón

Trabajo con la clonación de un repositorio ya existente

1. Clonar el repositorio ubicado en https://github.com/grayghostvisuals/practice-git.

```
$ git clone https://github.com/grayghostvisuals/practice-git.git
Cloning into 'practice-git'...
remote: Enumerating objects: 639, done.
remote: Total 639 (delta 0), reused 0 (delta 0), pack-reused 639
Receiving objects: 100% (639/639), 565.61 KiB | 2.49 MiB/s, done.
Resolving deltas: 100% (239/239), done.
```

2. Mediante la ejecución de un solo comando (una sola línea), compuesto de un comando Git para listar archivos, usado conjuntamente con el comando de wc de Linux, indicar cuántos archivos contiene el proyecto en su WD. Se deben incluir aquellos archivos que haya en subdirectorios. El directorio .git no se considera un subdirectorio del proyecto.
   Hacerlo de diferentes maneras, al menos de dos.

```
$ git log --oneline | wc -l
271

$ git rev-list --count HEAD
271
```

3. Mostrar la lista de contribuyentes/colaboradores del repositorio, así como el número de commits que ha hecho cada uno.

```

```

4. Mostrar, con un comando, los nombres de los contribuyentes/colaboradores que hay en el repositorio.

```
git shortlog -sc
```

5. Mostrar, con un comando, el número de contribuyentes/colaboradores que hay en el repositorio.

```
git ls-remote | wc -l
```

6. Mediante un comando de Linux, usado conjuntamente con el comando usado anteriormente, mostrar el número de commits existentes en el repositorio. Se trata de hacer la suma de los números que hay dentro de un archivo o flujo de caracteres.

```
git rev-list --count HEAD
```

7. Mostrar el histórico de commits en el repositorio, en modo de un commit por línea, con salida gráfica y para todas las ramas del repositorio.

```
git log --oneline --all
```

8. Mostrar los nombres de ramas que existen.

```
git branch
```

9. Mostrar el estado del repositorio y explicar la información que se obtiene.

```
git status
```

10. Mostrar las direcciones de los servidores en la nube donde se almacena el repositorio, tanto para subir contenidos (push), como para descargarlos (fetch o pull).

```
git remote -v
```

11. Mostrar los commits realizados por el contribuyente grayghostvisuals durante el año 2019.

```
git log --author=grayghostvisuals --since=2019-01-01 --
until=2019-12-31
```

Trabajo con un nuevo repositorio actividad

12. Crear, dentro del directorio reposgit, un repositorio de nombre actividad.

```
git init actividad
```

13. Mostrar el histórico de commits del repositorio. Explicar el resultado obtenido.

```
"git log --oneline"
fatal: your current branch 'main' does not have any commits
yet
```

14. Mostrar los nombres de los contribuyentes al repositorio con el número de commits que ha realizado cada uno. Explicar el resultado obtenido.

```
"git shortlog -s"
```

15. Mostrar la información disponible del último *commit*. Explicar el resultado obtenido.

```
"git show"
fatal: your current branch 'main' does not have any commits
yet
```

16. Mostrar el nombre de las ramas existentes. Explicar el resultado obtenido.

```
"git branch"
```

17. Crear una nueva rama de nombre bifurca. Explicar el resultado obtenido.

```
"git branch bifurca"
fatal: not a valid object name: 'main'
```

18. Mostrar las direcciones de los servidores en la nube donde se almacena el repositorio, tanto para subir contenidos (push), como para descargarlos (fetch o pull).

```
"git remote -v"
```

19. Verificar la conectividad y validez de los objetos de la base de datos de Git. Explicar el resultado obtenido.

```
"git fsck"
notice: HEAD points to an unborn branch (main)
notice: No default references
```

A partir de aquí, se irán realizando operaciones, tras cada una de las cuáles se irá comprobando el estado del repositorio.

20. Crear un archivo de nombre **letras** que contenga la letra **a**.

```
"echo a > letras "
```

```
"git status"
On branch main

No commits yet

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

nothing added to commit but untracked files present (use "git
add" to track)
```

21. Añadir el archivo letras al SA. ¿Ha cambiado el contenido del archivo?

```
"git add letras"
"git status"
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
      new file:   letras
```

22. Ejecutar el comando git rm letras. Explicar la respuesta obtenida.

```
"git rm letras "
error: the following file has changes staged in the index:
    letras
(use --cached to keep the file, or -f to force removal)
"git status"
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
      new file:   letras
```

23. Sacar el archivo del SA (*unstage*).

```
"git rm --cached letras"
rm 'letras'
"git status"
On branch main

No commits yet

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

nothing added to commit but untracked files present (use "git
add" to track)
```

24. Añadir de nuevo el archivo al SA. Realizar un *commit* con el mensaje *Creado el archivo **letras***.

```
"git add letras"
"git commit -m 'Creado el archivo letras.'"
[main (root-commit) 713ebfd] Creado el archivo letras.
 1 file changed, 1 insertion(+)
 create mode 100644 letras
"git status"
On branch main
nothing added to commit but untracked files present (use "git
add" to track)
```

25. Mostrar los nombres de los archivos contenidos en este *commit* recién creado. Únicamente los nombres.

```
"git ls-tree --name-only HEAD"
letras
```

26. Mostrar los nombres de los archivos contenidos en el *commit* recién creado, con sus SHA y su tipo.

```
"git ls-tree -r HEAD"
100644 blob 0d13814d585b11dbeae7a42377d7422f93d2be15  letras
```

27. ¿Cuál es el SHA del archivo **letras** en el WD?

```
"git hash-object letras"
0d13814d585b11dbeae7a42377d7422f93d2be15
```

28. ¿Cuál es el SHA del archivo **letras** en el SA?

```
"git ls-files --stage letras"
100644 0d13814d585b11dbeae7a42377d7422f93d2be15 0     letras
```

29. Añadir la letra **b** al archivo **letras**. Subir el archivo al SA. Sacar el archivo del SA. ¿Ha cambiado el contenido del archivo?

```
"echo b>> letras"
```

30. Añadir de nuevo el archivo **letras** al SA. Añadir la letra **c** al archivo **letras**. Comprobar el estado del repositorio. Hacerlo de nuevo con el comando **`git status -s`**. Explicar la salida devuelta por ambos comandos de comprobación del estado del repositorio.

```
"git add letras"
"git status"
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      modified:   letras

"echo c >> letras"
"git status"
On branch main
```

```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      modified:   letras

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working
directory)
      modified:   letras
```

31. ¿Qué sucede si se ejecuta el comando **`git restore letras`**?

    Si ejecutas el comando git restore letras, se descartarán los cambios realizados en el archivo "letras" y se restaurará al estado del commit más reciente.

32. ¿Qué sucede si se ejecuta el comando **`git restore --staged letras`**?

    Si ejecutas el comando git restore --staged letras, el archivo "letras" volverá al estado en el Working Directory, lo que significa que los cambios que habías añadido al Staging Area se descartarán.

33. Para este apartado, volver al estado en el que se encuentra el repositorio justo detrás del punto 27.
    Mostrar un listado del histórico de *commits*.

```
"git log --oneline"
6e4eab3 Creado el archivo letras.
"git checkout HEAD^"
M     letras
```

34. Mostrar los nombres de los contribuyentes al repositorio con el número de *commits* que ha realizado cada uno. Explicar el resultado obtenido.

```
"git shortlog -sne"
     1      Zarras <jlorenzolimon@gmail.com>
```

35. Mostrar la información disponible del último *commit*. Explicar el resultado obtenido.

```
"git show HEAD"
commit 6e4eab36e91acdc9804d54520813ce2b55d4c492
Author: Zarras <jlorenzolimon@gmail.com>
Date:   Wed Nov 8 20:19:24 2023 +0100

    Creado el archivo letras.

diff --git a/letras b/letras
new file mode 100644
index 0000000..0d13814
--- /dev/null
+++ b/letras
@@ -0,0 +1 @@
```

```
+a
```

36. Mostrar el nombre de las ramas existentes. Explicar el resultado obtenido.

```
"git branch"
* main
```

37. Verificar la conectividad y validez de los objetos de la base de datos de Git. Explicar el resultado obtenido.

```
git fsck
```

38. Realizar un *commit* con el mensaje *Modificado el directorio **letras***.

```
"git commit -m 'Modificado el directorio letras.'"
[main f6438a5] Modificado el directorio letras.
 1 file changed, 1 insertion(+)
"git show"
commit f6438a58c7d9e37238fd61225e564d53a312a981
Author: Zarras <jlorenzolimon@gmail.com>
Date:   Wed Nov 8 20:19:26 2023 +0100

    Modificado el directorio letras.

diff --git a/letras b/letras
index 0d13814..e2262f8 100644
--- a/letras
+++ b/letras
@@ -1 +1,2 @@
 a
+b
```

39. Enmendar el *commit* para que el mensaje sea *Modificado el archivo **letras***.

```
"git commit --amend -m 'Modificado el archivo letras.'"
[main 7b1801c] Modificado el archivo letras.
 Date: Wed Nov 8 20:19:26 2023 +0100
 1 file changed, 1 insertion(+)
```

40. Mostrar el contenido del archivo **letras** en el *commit* actual y en el anterior.

```
git show HEAD:letras
a
b

git show HEAD^:letras
a
```

41. Modificar el contenido del archivo **letras** para que contenga las letras a y c, cada una en una línea, dejando una línea en blanco entre la a y la c.
    Crear un archivo **numeros**, insertando una línea con los números impares 1, 3 y 5 (separados por comas), y otra con los pares 2, 4 y 6 (separados por comas).
    Añadir ambos archivos al SA. Comprobar el estado del repositorio, en el formato largo y en el corto.

```
echo "a\n\nc" > letras

echo "1,3,5\n2,4,6" > numeros

git add letras numeros

git show
```

42. Confirmarlos con el mensaje *Modificado el archivo leras y añadido el archivo números*. Explicar el resultado del comando.

```
git commit -m "Modificado el archivo letras y añadido el
archivo números."

git show
```

43. Mostrar el contenido del archivo números en el commit actual y en el anterior. Explicar el resultado obtenido.

```
"git show HEAD:numeros"
"1,3,5\n2,4,6"
```

44. Mostrar las diferencias del archivo letras en sus versiones del *commit* anterior y la del actual. Utilizar el operado ~.

```
"git show HEAD:numeros"
"1,3,5\n2,4,6"
"git diff HEAD HEAD~1:letras"
diff --git a/letras b/letras
index 85c64e5..e2262f8 100644
--- a/letras
+++ b/letras
@@ -1 +1,2 @@
-"a\n\nc"
+a
+b
```

45. Mostrar las diferencias entre el *commit* anterior y el actual. Utilizar el operado ^.

```
"git diff HEAD^"
diff --git a/letras b/letras
index 85c64e5..e2262f8 100644
--- a/letras
+++ b/letras
@@ -1 +1,2 @@
-"a\n\nc"
+a
+b
diff --git a/numeros b/numeros
deleted file mode 100644
index a5ae4d0..0000000
--- a/numeros
+++ /dev/null
@@ -1 +0,0 @@
-"1,3,5\n2,4,6"
```

46. Modificar el archivo **numeos** para que tenga: en la primera línea, los números impares del 1 al 9, separados por comas; en la segunda, los números pares 2 y 4 separados por comas; la tercera línea en blanco y la cuarta línea, los números 30, 20 y 10 separados por comas. En un solo comando, añadir el archivo al SA y confirmarlo con el mensaje *Retocado el archivo numeros*.

```
"echo "1,3,5,7,9\n2,4\n\n30,20,10""
"git commit -am 'Retocado el archivo numeros'"
[main ccb082b] Retocado el archivo numeros
 1 file changed, 1 insertion(+), 1 deletion(-)
```

47. Mostrar el histórico de confirmaciones en formato de una línea por *commi*t y gráfico.

```
“git log --oneline --graph –all”
* ccb082b Retocado el archivo numeros
* 948883b Modificado el archivo letras y aâ”œâ–’adido el
archivo nâ”œâ•‘meros.
* 7b1801c Modificado el archivo letras.
* 6e4eab3 Creado el archivo letras.
```

48. Eliminar completamente el último *commit* y mostrar el histórico de confirmaciones en formato de una línea por *commit* y gráfico.

```
"git reset --hard HEAD^"
HEAD is now at ccb082b Retocado el archivo numeros
```

49. Arreglar el árbol no referenciado del apartado anterior (su id es **4b825dc**) y volver a probar el comando **`git fsck`**.

```
git commit-tree \
```

50. Verificar la conectividad y validez de los objetos de la base de datos de Git. Explicar el resultado obtenido.

```
git fsck
```

51. Crear una rama de nombre **ramifica**. Cambiarle el nombre a **bifurca**.

```
"git branch ramifica"
"git branch -m ramifica bifurca"
```

52. Crear un archivo llamado **simbolos** en **main**. Añadirle, en líneas separadas, los caracteres **!**, **·** y **$**. Cambiar a **bifurca**, ¿existe el archivo símbolos? Explicar la respuesta.

```
"echo '!\n,\n.\n$' > simbolos"
"git switch -f bifurca"
Switched to branch 'bifurca'
```

53. En **bifurca**, añadir al SA el archivo **simbolos**. Comprobar el estado del repositorio. Listar el contenido del WD. Cambiar a **main**. Comprobar el estado del repositorio. Comparar los resultados obtenidos en una y otra ramas. Explicar la respuesta.

```
git add simbolos

ls

git switch main

git status
```

54. En **main**, confirmar con el mensaje *Añadido el archivo simbolos*. Comprobar el estado del repositorio. Comparar los resultados obtenidos en una y otra ramas. Explicar la respuesta. Listar el contenido del WD.

```
git add simbolos

git commit -m "Añadido el archivo simbolos"

git status

"dir"
 El volumen de la unidad C no tiene etiqueta.
 El n£mero de serie del volumen es: 1A74-37B8

 Directorio de C:\Users\jlore\Desktop\reposgit\actividad

08/11/2023  20:19    <DIR>          .
08/11/2023  20:19    <DIR>          ..
08/11/2023  20:19                11 letras
08/11/2023  20:19                31 numeros
08/11/2023  20:19                45 simbolos
               3 archivos             42 bytes
               2 dirs  99.120.406.528 bytes libres
```

55. Cambiar a **bifurca**. Comprobar el estado del repositorio. Listar el contenido del WD. ¿Coincide el contenido del WD en una y otra rama? Explicar la respuesta.

```
"git switch -f bifurca"
Switched to branch 'bifurca'
"git status"
On branch bifurca
nothing to commit, working tree clean
"dir"
 El volumen de la unidad C no tiene etiqueta.
 El n£mero de serie del volumen es: 1A74-37B8

 Directorio de C:\Users\jlore\Desktop\reposgit\actividad

08/11/2023  20:19    <DIR>          .
08/11/2023  20:19    <DIR>          ..
08/11/2023  20:19                11 letras
08/11/2023  20:19                31 numeros
               2 archivos             42 bytes
               2 dirs  99.120.406.528 bytes libres
```

56. En la rama **bifurca**., crear el archivo simbolos. Añadirle, en líneas separadas, los caracteres **%**, **/** y **=**. Comprobar el estado del repositorio.

```
"echo '\n/\n='"
"git status"
On branch bifurca
Untracked files:
  (use "git add <file>..." to include in what will be
committed)
      simbolos

nothing added to commit but untracked files present (use "git
add" to track)
```

57. Añadir **simbolos** al **SA**. Comprobar el estado del repositorio. Confirmar con el mensaje *Añadido el archivo símbolos en bifurca*.

```
"git add simbolos"
"git status"
On branch bifurca
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      new file:   simbolos
```

58. En **bifurca**, mostrar el histórico de *commits*. Cambiar a **main**. Mostrar el histórico de *commits*, ¿coinciden ambos históricos? Explicar la respuesta.

```
"git log --oneline"
ccb082b Retocado el archivo numeros
948883b Modificado el archivo letras y aâ”œâ–’adido el
archivo nâ”œâ•‘meros.
7b1801c Modificado el archivo letras.
6e4eab3 Creado el archivo letras.
"git switch -f main"
Switched to branch 'main'
"git log --oneline"
ccb082b Retocado el archivo numeros
948883b Modificado el archivo letras y aâ”œâ–’adido el
archivo nâ”œâ•‘meros.
7b1801c Modificado el archivo letras.
6e4eab3 Creado el archivo letras.
```

59. En **main**, mostrar el histórico de *commits* de todas las ramas. Cambiar a **bifurca**. Mostrar el histórico de *commits*, de todas las ramas.

```
"git log --oneline --all -graph"
* ccb082b Retocado el archivo numeros
* 948883b Modificado el archivo letras y aâ”œâ–’adido el
archivo nâ”œâ•‘meros.
* 7b1801c Modificado el archivo letras.
* 6e4eab3 Creado el archivo letras.
"git switch -f bifurca"
Switched to branch 'bifurca'
"git log --oneline --all --graph"
* ccb082b Retocado el archivo numeros
```

```
* 948883b Modificado el archivo letras y aâ”œâ–’adido el
archivo nâ”œâ•‘meros.
* 7b1801c Modificado el archivo letras.
* 6e4eab3 Creado el archivo letras.
```

60. Cambar a **main**. Crear un archivo de nombre **nuevo_en_main** que contenga una línea con el texto *una línea*. Añadirlo al SA. Añadir al archivo recién creado el texto *otra_línea*. Mostrar el estado el simplificado repositorio. Mostrar el contenido del archivo **nuevo_en_main**.

```
"git switch -f main"
Switched to branch 'main'
"echo 'una lÃ-nea' > nuevo_en_main"
"git add nuevo_en_main"
"echo 'otra lÃ-nea' >> nuevo_en_main"
"git status -s"
AM nuevo_en_main
"type nuevo_en_main"
"una lÃ-nea"
"otra lÃ-nea"
```

61. En **main**, listar el contenido del WD. Reservar el trabajo en curso. Listar el contenido del WD. Mostrar el estado del repositorio. Mostrar la lista de reservas. Sacar el trabajo reservado al WD. Listar el contenido del WD. Mostrar el estado del repositorio. ¿Coincide el contenido del WD y esl estado del repositorio cuando se ha hecho la reserva y cuando se deshace? Explicar la respuesta.

```
"dir"
 El volumen de la unidad C no tiene etiqueta.
 El n£mero de serie del volumen es: 1A74-37B8

 Directorio de C:\Users\jlore\Desktop\reposgit\actividad

08/11/2023  20:19    <DIR>          .
08/11/2023  20:19    <DIR>          ..
08/11/2023  20:19                11 letras
08/11/2023  20:19                31 nuevo_en_main
08/11/2023  20:19                31 numeros
               3 archivos             73 bytes
               2 dirs  99.120.402.432 bytes libres
"git stash"
Saved working directory and index state WIP on main: ccb082b
Retocado el archivo numeros
"dir"
 El volumen de la unidad C no tiene etiqueta.
 El n£mero de serie del volumen es: 1A74-37B8

 Directorio de C:\Users\jlore\Desktop\reposgit\actividad

08/11/2023  20:19    <DIR>          .
08/11/2023  20:19    <DIR>          ..
08/11/2023  20:19                11 letras
08/11/2023  20:19                31 numeros
               2 archivos             42 bytes
               2 dirs  99.120.402.432 bytes libres
"git status"
```

```
On branch main
nothing to commit, working tree clean
"git stash pop"
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      new file:   nuevo_en_main

Dropped refs/stash@{0}
(66490c577fd01bd06a00c78615e0bd9913d017e2)
"git stash clear"
"dir"
 El volumen de la unidad C no tiene etiqueta.
 El n£mero de serie del volumen es: 1A74-37B8

 Directorio de C:\Users\jlore\Desktop\reposgit\actividad

08/11/2023  20:19    <DIR>          .
08/11/2023  20:19    <DIR>          ..
08/11/2023  20:19                11 letras
08/11/2023  20:19                31 nuevo_en_main
08/11/2023  20:19                31 numeros
               3 archivos             73 bytes
               2 dirs  99.120.402.432 bytes libres
"git status"
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      new file:   nuevo_en_main
```

62. Fusionar la rama **bifurca** en la rama **main**. ¿Qué sucede?, ¿qué hay que hacer ahora? Ejecutar los comandos necesarios para que no se pierda el trabajo que hubiera en curso, sino que se guarde en el repositorio.

```
"git merge bifurca"
Already up to date.
```

63. Si la rama bifurca no se fusionó a **main** anteriormente, hacerlo ahora. ¿Qué sucede? Explicar el resultado del comando. Muestra el estado del repositorio. ¿Hay que hacer algo para que el repositorio no tenga “nada pendiente”? En caso afirmativo, hacerlo.

```
"git merge --no-ff bifurca"
Already up to date.
```

64. Mostrar el histórico de confirmaciones del repositorio, en formato de un *commit* por línea, gráfico y para todas las ramas.

```
"git log --oneline"
ccb082b Retocado el archivo numeros
948883b Modificado el archivo letras y aâ”œâ–’adido el
archivo nâ”œâ•‘meros.
7b1801c Modificado el archivo letras.
6e4eab3 Creado el archivo letras.
```

65. Mostrar el histórico de confirmaciones del repositorio en los que ha sido modificado el archivo **simbolos**, en formato de un *commit* por línea, gráfico y para todas las ramas.

```
"git log --oneline --graph --all --follow -p -- simbolos"
```

66. El archivo **letras** ha sido confirmado tres veces. Mostrar en contenido en la versión que hay en el WD. Recuperar en el WD la versión que del mismo hay en el segundo *commit*. Mostrar el contenido del archivo letras. Mostrar el estado del repositorio.

```
cat letras

git restore 4504554 letras
```

67. Descartar completamente la versión recuperada del archivo letras, es decir, que el archivo vuelva a estar como antes de recuperar la versión anterior y no quede rastro de la versión recuperada. Mostrar el contenido del archivo letras. Explicar los pasos que se dan, es decir, los comandos ejecutados.

```

```

68. Crear un nuevo archivo de nombre **mar_de_dudas**, que contenga una línea con el texto *No sé qué será de mi vida*. Mostrar el estado del repositorio. Crear un archivo **.gitignore** que contenga como texto únicamente el nombre del archivo recién creado. Mostrar el estado del repositorio. Explicar qué ha sucedido. Confirmar el trabajo que deba ser confirmado. Mostrar el histórico de confirmaciones del repositorio, en formato de un *commit* por línea, gráfico y para todas las ramas.

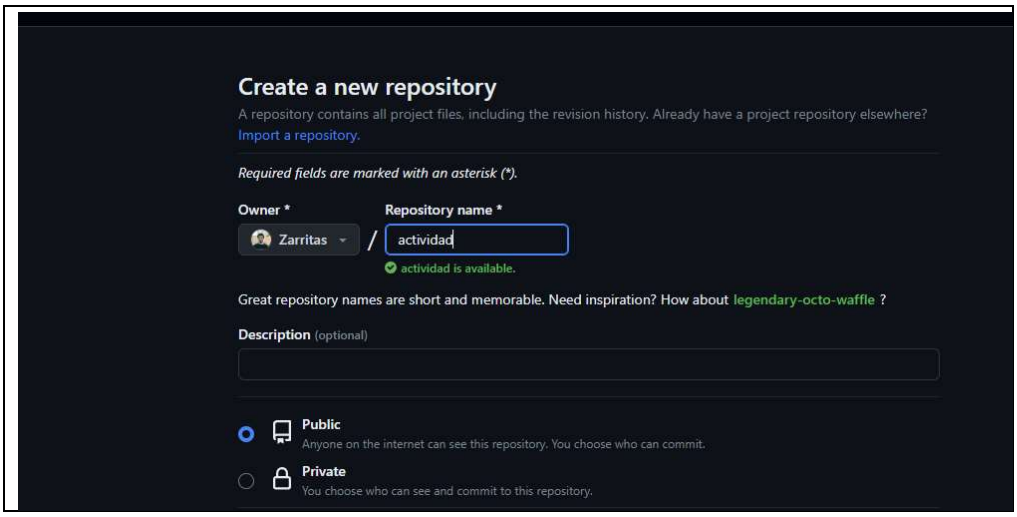
```
"echo "No sÃ© quÃ© serÃ¡ de mi vida" > mar_de_dudas"
"git status"
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      new file:   nuevo_en_main

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

"echo mar_de_dudas > .gitignore"
"git status"
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      new file:   nuevo_en_main

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

```
"git add ."
"git commit -m 'Cosas que se debian confirmar'"
[main e0d3fb8] Cosas que se debian confirmar
 2 files changed, 3 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 nuevo_en_main
"git log --oneline --graph --all"
* e0d3fb8 Cosas que se debian confirmar
* ccb082b Retocado el archivo numeros
* 948883b Modificado el archivo letras y aâ”œâ–’adido el
archivo nâ”œâ•‘meros.
* 7b1801c Modificado el archivo letras.
* 6e4eab3 Creado el archivo letras.
```

Manejo de Git con GitHub

De las acciones que se realicen en GitHub se debe hacer una captura.

69. En GitHub, crear un repositorio público y sin archivos, de nombre **actividad**.



70. Comprobar las conexiones que tiene el repositorio local con repositos remoto. Conectar el repositorio local con el remoto antes creado. Volver a comprobar las conexiones que tiene el repositorio local con repositos remotos.

```
Estos puntos los he unificado abajo del todo
```

71. Descargar el contenido de la rama **main** del repositorio remoto a local. ¿Se descarga? Explicar la respuesta. Si no se pudo realizar la descarga, arrglarlo para que se pueda.

72. Mostrar el histórico de confirmaciones del repositorio (un *commit* por línea, en modo gráfico y para todas las ramas). Mostrar las referencias existentes en el repositorio remoto.

73. Subir la rama **bifurca** a remoto. Explicar el resultado devuelto por el comando y la acción realizada. Mostrar las referencias existentes en el repositorio remoto.

```
"git remote -v"

"git remote add origin
https://github.com/Zarritas/actividad.git"

"git remote -v"
origin      https://github.com/Zarritas/actividad.git (fetch)
origin      https://github.com/Zarritas/actividad.git (push)

"git pull"
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you
can do so with:

    git branch --set-upstream-to=origin/<branch> main


"git log --oneline --graph --all"
* de0b63f Cosas que se debian confirmar
* 34645c7 Retocado el archivo numeros
* 3658f6a Modificado el archivo letras y aâ”œâ–’adido el
archivo nâ”œâ•‘meros.
* 7c01599 Modificado el archivo letras.
* b1b095b Creado el archivo letras.

"git push origin bifurca"
To https://github.com/Zarritas/actividad.git
 * [new branch]      bifurca -> bifurca
```

74. Crear un archivo de nombre **comun**, tanto en local como en remoto. En local contendrá el texto *¿Y ahora qué, remoto?* En remoto contendrá el texto *¿Y ahora qué, local?* Añadir cada archivo al repositorio donde fue creado. Sincronizar ambos repositorios para que contengan la misma información. Explicar todos los pasos y el resultado de cada acción usando para esto los comandos que se vienen usando anteriormente: **`git log`**, **`git status`**…

    En GitHub:

```
aqui solo creamos el archivo, no hacemos nada más
```

    En local:

```
hacemos git pull, nos dice que tenemos distintos commits asiq
tenemos que hacer merge, con un archivo igual y nos manda a
nuestro editor de texto a solucionar el problema, una vez
solucionado el conflicto simplemente hacemos git push y listo
```

75. Mostrar el histórico de *commits* del repositorio local (una línea por commit, modo gráfico, todas las ramas) y las referencias existentes en el repositorio remoto.

```
git log --oneline --graph --all
```

Trabajo con un nuevo repositorio clonaactividad

76. Crear un nuevo repositorio de nombre **clonactividad**, como una clonación del repositorio **LOCAL** de nombre **actividad** que se está usando en la práctica. Entra en el WD del nuevo repositorio **clonactividad**, muestra el estado del repositorio, el histórico de confirmaciones (una línea por commit, modo gráfico, todas las ramas) y los servidores remotos a los que está conectado.

```
Va a salir el mismo numero de comits que el repositorio
creado anteriormente
```

77. Crear un nuevo archivo **me_revierto** con el texto *Me voy a revertir*. Confirmarlo. Cambiarle el nombre a **revertiendo**. Confirmarlo. A partir de aquí, comprobar el estado del repositorio tras la ejecución de cada comando. Cambiar el nombre del archivo **me_revierto** por **revertiendo**. Añadir el cambio al SA. Confirmarlo. Explicar el resultado de las operaciones.

```
simplemente creamos 3 commits de el estado de un archivo y su
nombre para que cuando volvamos para atras sepamos en que
paso estamos gracias al nombre del archivo
```

78. Revertir el último *commit* y mostrar el histórico de *commits*. Explicar las operaciones usando para esto los comandos que se vienen usando anteriormente: **`git log`**, **`git status`**…

```
lo de arriba
```

ANEXO I: Listado de comandos de Git que se usan en la práctica

(algunos no es imprescindible usarlos)

`git init`	`git add`	`git remote`	`git ls-files`
`git clone`	`git commit`	`git rm`	`git ls-tree`
`git help`	`git diff`	`git mv`	`git ls-remote`
`git config`	`git branch`	`git restore`	`git rev-list`
`git status`	`git checkout`	`git reset`	`git hash-object`
`git log`	`git switch`	`git push`	`git fsck`
`git shortlog`	`git stash`	`git pull`	
`git show`	`git merge`	`git fetch`	

ANEXO II: Listado de comandos de Linux que se usan en la práctica

`ls`	`cd`	`echo`	`pwd`	`rm`	`mv`	`cut`	`wc`	`awk`