GIT primeros pasos

1. Creo un directorio de trabajo
   - mkdir {directorio HOME}/ejemplo/
   - cd {directorio HOME}/ejemplo/
   - git status
   - ls -la (se muestra el contenido del directorio con más información y los archivos ocultos) ***(captura de pantalla 1)***
2. Inicializo el repositorio. “git init”
   - git init
   - ls (se muestra el contenido del directorio)
   - ls -la
   - git status
   - git status -s ***(captura de pantalla 2)***
3. Creo un archivo **index.html** vacío en el directorio de trabajo.
4. Creo un archivo nuevo en ese directorio
   - touch **primer_archivo.txt** (se crea **primer_archivo.txt** vacío)
   - ls
   - git status ***(captura de pantalla 3)***
5. Lo añado al repositorio, para que sea seguido:
   - git add **primer_archivo.txt**
   - Podía hacer git add . (se añadiria al stage area **index.html** y **primer_archivo.txt**)
   - git status -s ***(captura de pantalla 4)***
6. Hago el commit:
   - git commit -m “Añado el primer archivo vacío”.
   - git status
   - git log --oneline ***(captura de pantalla 5)***
7. Creo dos archivos nuevos
   - touch **segundo_archivo.txt**
   - touch **tercer_archivo.txt**
   - git status

- git add **segundo_archivo.txt**
- git status
- git commit -m “Añado el segundo archivo”
- git status -s
- git add **tercer_archivo.txt**
- git status
- git commit -m “Añado el tercer archivo”
- git log
- git log --oneline ***(captura de pantalla 6)***

8. Edito el primer archivo
   - echo “Creo una primera línea en el primer archivo” >> **primer_archivo.txt** (al final del archivo se añade una linea de texto en el fichero **primer_archivo.txt**)
   - git status
   - git add **primer_archivo.txt**
   - git status
   - echo “Creo una primera línea en el segundo archivo.” >> **segundo_archivo.txt**
   - echo “Creo una primera línea en el tercer archivo.” >> **tercer_archivo.txt**
   - git status -s
   - git add **segundo_archivo.txt**
   - git status ***(captura de pantalla 7)***
   - git commit -m “Introduzco una línea en el primer y en el segundo archivo”
   - git status
   - git log
   - git log --oneline ***(captura de pantalla 8)***
   - git add **tercer_archivo.txt**
   - git commit -m “Introduzco una línea en el tercer archivo”
   - git status -s
   - git log –oneline ***(captura de pantalla 9)***
9. Ver cambios con diff
   - echo “Creo una segunda línea en el primer archivo” >> primer_archivo.txt
   - cat primer_archivo.txt
   - git status

- git diff (diferencias entre directorio trabajo y el stage area)
- echo “Creo una segunda línea en el segundo archivo” >> segundo_archivo.txt
- cat segundo_archivo.txt ***(captura de pantalla 10)***
- git status -s
- git diff ***(captura de pantalla 11)***
- git diff segundo_archivo.txt

10. Ver cambios en el stage area
    - git add segundo_archivo.txt
    - git status
    - git diff ***(captura de pantalla 12)***
    - git diff --staged
    - git diff --cached ***(captura de pantalla 13)***

      (--cached y --staged muestran lo mismo, diferencias entre el stage area y el HEAD del repositorio)
    - git diff HEAD (diferencias entre el directorio de trabajo y el HEAD del repositorio) ***(captura de pantalla 14)***
    - git add primer_archivo.txt
    - git status
    - git diff
    - git diff --staged
    - git commit -m “Añado líneas en el primer y en el segundo archivo”
    - git status
    - git diff ***(captura de pantalla 15)***
    - git diff --staged ***(captura de pantalla 16)***
11. Eliminar archivos
    - touch temp1.txt
    - touch temp2.txt
    - git status
    - git add .
    - git status -s
    - git commit -m “Añado dos archivos de prueba para borrar”
    - rm temp1.txt

- git status ***(captura de pantalla 17)***
- git add temp1.txt
- git status ***(captura de pantalla 18)***
- git commit -m “Borro el archivo temp1.txt”
- git status
- git log --oneline
- git rm temp2.txt
- git status ***(captura de pantalla 19)***
- git commit -m “Borro el archivo temp2.txt”
- git status
- git log --oneline ***(captura de pantalla 20)***

12. Mover y renombrar archivos
    - mv primer_archivo.txt archivo_primero.txt
    - git status -s ***(captura de pantalla 21)***
    - git add archivo_primero.txt
    - git rm primer_archivo.txt
    - git status
    - git mv segundo_archivo.txt archivo_segundo.txt
    - git status
    - mkdir dir1
    - git mv tercer_archivo.txt dir1/archivo_tercero.txt
    - git status -s ***(captura de pantalla 22)***
    - git commit -m “Cambios los nombres de los archivos y la estructura de los directorios”
    - git status
    - git log --oneline ***(captura de pantalla 23)***
13. Deshaciendo cambios en la zona de trabajo
    - git status
    - echo “Añado una tercera línea al primer archivo” >> archivo_primero.txt
    - cat archivo_primero.txt
    - git status
    - git diff

- git checkout archivo_primero.txt (machaca el contenido de archivo_primero.txt en el directorio de trabajo por el contenido de archivo_primero.txt en el stage area) ***(captura de pantalla 24)***
- git checkout -- archivo_primero.txt (realiza lo mismo que el comando anterior)
- echo “Añado una cuarta línea al primer archivo” >> archivo_primero.txt
- cat archivo_primero.txt
- git diff (diferencias entre directorio trabajo y stage area)
- git add archivo_primero.txt
- git diff (diferencias entre directorio trabajo y stage area)
- echo “Añado una quinta línea al primer archivo” >> archivo_primero.txt
- cat archivo_primero.txt
- git status -s ***(captura de pantalla 25)***
- git diff (diferencias entre directorio trabajo y stage area) ***(captura de pantalla 26)***
- git checkout archivo_primero.txt
- git status -s
- cat archivo_primero.txt ***(captura de pantalla 27)***
- git checkout HEAD archivo_primero.txt (machaco archivo_primero.txt del directorio de trabajo con archivo_primero.txt del **HEAD** en el repositorio)

14. Quitando archivos del stage area
    - git status
    - echo “Añado una tercera línea al primer archivo” >> archivo_primero.txt
    - cat archivo_primero.txt
    - git status
    - git add archivo_primero.txt
    - git status -s ***(captura de pantalla 28)***
    - git reset archivo_primero.txt (quita del stage area los cambios en archivo_primero.txt) ***(captura de pantalla 29)***
    - git reset . (Quita del stage area todos los ficheros que se hayan añadido anteriormente)
    - git status -s
15. Cambiando el último commit
    - git add archivo_primero.txt
    - git commit -m “Añado una tercera línea en el archivo primero”
    - git log

- git log --oneline ***(captura de pantalla 30)***
- echo “Añado una cuarta línea al primer archivo” >> archivo_primero.txt
- git status -s
- git add archivo_primero.txt
- git commit --amend -m “Añado dos líneas en el archivo primero” ***(captura de pantalla 31)***

16. Recuperando versiones antiguas de archivos
    - git log --oneline
    - cat archivo_primero.txt
    - git checkout **$SHA_commit**(identificador del commit anterior) archivo_primero.txt (recuperamos la versión del commit anterior de archivo_primero.txt, al hacer checkout el directorio de trabajo y el stage area contienen lo mismo) ***(captura de pantalla 32)***
    - git log --oneline (fijarse a donde apunta el HEAD) ***(captura de pantalla 33)***
    - cat archivo_primero.txt
    - git diff --staged (diferencias entre lo que hay en la stage area y el HEAD del repo) ***(captura de pantalla 34)***
    - git checkout HEAD archivo_primero.txt
    - git status -s
17. Revirtiendo un commit al estado anterior
    - git log --oneline
    - git revert HEAD (crea un nuevo commit revirtiendo los cambios del commit anterior)
    - git status -s
    - git log --oneline (fijarse como ha cambiado el historial) ***(captura de pantalla 35)***

18. Deshaciendo commits: comando **reset**

Copia los SHA-1 de todos los commits en algún lugar porque los vas a necesitar después

18.1. Soft reset

git status -s

git log --oneline --all

- git reset --soft $id_commit (4º último commit **HEAD~4**)***(captura de pantalla 36)***

  git status

  git log --oneline (fijarse donde esta apuntando el **HEAD** y la rama **master**)

- git diff --staged (diferencia entre el directorio el contenido del stage area y el **HEAD** del repositorio) (fijarse que con **reset --soft** no se modifica el directorio de trabajo ni el stage area) ***(captura de pantalla 37)***

git diff (diferencia entre el directorio de trabajo y el stage area)

git reset --soft $id_commit (commit donde estábamos al inicio)

git log --oneline --all

18.2. Mixed reset (es lo mismo que hacer **git reset $id_commit**)

git status -s

git log --oneline --all

- git reset --mixed $id_commit (4º último commit **HEAD~4**)***(captura de pantalla 38)***

  git status -s

- git log --oneline (fijarse donde esta apuntando el **HEAD** y la rama **master**) ***(captura de pantalla 39)***

  git diff --staged (diferencia entre el directorio el contenido del stage area y el **HEAD** del repositorio)(fijarse que con **reset --mixed** el directorio de trabajo no se modifica pero el stage area **SI** se actualiza)

  git diff (diferencia entre el directorio de trabajo y el stage area)

  git reset --mixed $id_commit (commit donde estábamos al inicio)

  git log --oneline

18.3. Hard reset

git status -s

git log --oneline --all –graph

- git reset --hard $id_commit (4º último commit HEAD~4) ***(captura de pantalla 40)***

  git status

  git log –oneline

  git diff --staged (diferencia entre el directorio el contenido del stage area y el HEAD del repositorio)(fijarse que con reset --hard tanto el directorio de trabajo como el stage area SI se modifican)

  git diff (diferencia entre el directorio de trabajo y el stage area)

  git reset --hard $id_commit (commit donde estábamos al inicio)

  git log --oneline --all

19. Crear una rama
    - git branch
    - git log --oneline --all
    - git branch desarrollo
    - git branch ***(captura de pantalla 41)***

20. Cambio de rama
    - git branch
    - git checkout desarrollo
    - git log --oneline --all ***(captura de pantalla 42)***

21. Edición en esta nueva rama
    - echo “Inserto una línea en el archivo primero” >> archivo_primero.txt
    - git status -s
    - git add .
    - git commit -m “Introduzco una nueva línea en el archivo_primero.txt de la rama de desarrollo”
    - git log --oneline --all ***(captura de pantalla 43)***
    - git checkout master (cambio de rama)
    - git log --oneline –all
    - cat archivo_primero.txt
    - git checkout desarrollo (cambio de rama)
    - cat archivo_primero.txt
    - git log --oneline –all

22. Crear una rama y cambiarse a ella
    - git checkout master
    - git checkout -b experimento
    - echo “Experimento con una nueva línea en el archivo primero” >> archivo_primero.txt
    - git status -s
    - git add .
    - git commit -m “Experimento con una nueva línea en el archivo_primero.txt de la rama experimento”
    - git log --oneline --all --graph ***(captura de pantalla 44)***
    - cat archivo_primero.txt
    - git checkout master
    - git log --oneline
    - cat archivo_primero.txt
    - git checkout desarrollo
    - cat archivo_primero.txt
    - git log --oneline ***(captura de pantalla 45)***

23. Renombrando una rama
    - git branch
    - git branch -m experimento prueba
    - git branch ***(captura de pantalla 46)***

24. Borrando una rama
    - git branch rama_temporal
    - git branch
    - git branch -d rama_temporal
    - git branch ***(captura de pantalla 47)***
    - git checkout desarrollo

25. Visualizando gráficamente las ramas
    - git log --oneline --graph --all ***(captura de pantalla 48)***

26. Comparando commits
    - git diff prueba..desarrollo
    - git diff desarrollo..prueba ***(captura de pantalla 49)***
    - git diff prueba~2..desarrollo

27. Fusionando ramas
    - git log --oneline --graph –all
    - git checkout master
    - git merge desarrollo (merge tipo fast-fordward) ***(captura de pantalla 50)***

28. Conflicto
    - git merge prueba ***(captura de pantalla 51)***
    - cat archivo_primero.txt

29. Soluciones para resolver Conflictos

- Abortar
- Resolver manualmente

29.1. Abortar

- git merge --abort ***(captura de pantalla 52)***

  git status
- cat archivo_primero.txt ***(captura de pantalla 53)***

29.2. Resolver el problema de forma manualmente

git merge prueba

- cat archivo_primero.txt ***(captura de pantalla 54)***

  Edito el archivo y resuelvo los cambios

  git status

  git add archivo_primero.txt

  git commit -m “Resuelto el conflicto en la línea 4 entre las ramas master y prueba”
- git log --oneline --graph –all ***(captura de pantalla 55)***

30. Etiquetas
    - git tag
    - git tag -a v1.0 $SHA1-commit -m “Versión 1.0” (4º último commit)(crea una etiqueta anotada)
    - git tag ***(captura de pantalla 56)***
    - git show v1.0
    - git tag -a v1.1. -m “Versión 1.1. Fusionadas las ramas de desarrollo y master”
    - git tag
    - git show tag v1.1 ***(captura de pantalla 57)***
    - git tag -d v1.0 (elimina la etiqueta v1.0)
    - git tag ***(captura de pantalla 58)***