Universidad de La Laguna
Facultad de Matemáticas
Área: Lenguajes y Sistemas Informáticos

Técnicas Experimentales
Práctica de laboratorio **#2**

1. Iniciar una sesión de trabajo en GNU-Linux.

2. Abra una terminal.

3. Muestre el árbol de directorios de su HOME (`tree`).

4. ¿Cuáles son las funcionalidades del comando `git`?

5. Configure `git` con el **nombre de usuario** de manera que pueda etiquetar de forma correcta las actualizaciones que este realice. (`git config --global user.name "Nombre Apellido"`)

6. Configure `git` con la dirección de correo electrónico para asociarla a las actualizaciones que se hagan en el repositorio git. (`git config --global user.email "aluXXXXXXXXXX@ull.edu.es"`)

7. `git` permite almacenar la configuración global de un usuario en el archivo `.gitconfig`. Este archivo se encuentra en el directorio HOME del usuario. `git` almacena el remitente y el nombre del autor de un cambio en cada registro en el repositorio. Esta información se puede almacenar en el fichero de configuración global de manera que no se solicite cada vez que se haga un registro. Muestre el contenido del fichero `.gitconfig` (`cat .gitconfig`)

8. Muestre las configuraciones globales de `git`. (`git config --list`)

9. Sitúese en la **Carpeta de Proyecto** de la asignatura Técnicas Experimentales esto es en el directorio *TE* (`cd TE`).

10. Muestre el contenido del directorio actual (`ls -la`).

11. Cree un nuevo directorio denominado *prct02* (`mkdir prct02`). Este será el directorio actual durante la realización de esta práctica.

12. Sitúese en el directorio *prct02* y cree la estructura de directorios que le permita tener subcarpetas para el código y los documentos, es decir:
    - un subdirectorio *src*
    - un subdirectorio *docs*

13. Guarde el fichero PDF que contiene el enunciado de esta práctica en el directorio *docs*.

14. Situado en el directorio actual, inicialícelo para que sea un repositorio `git`. (`git init`)

15. Compruebe que se crea el directorio `.git` (`ls -la`).

16. Cree contenidos en el directorio actual. Ejecute los comandos:

```
touch test01
touch test02
touch src/src01
touch src/src02
ls > test01
```

17. Compruebe el contenido del directorio actual (`ls -la`).

18. Antes de realizar un registro en un repositorio Git es necesario marcar qué cambios se deben registrar añadiendo los nuevos ficheros y los ficheros cambiados al `índice del repositorio git`, esto es, al área de preparación. Esto crea una instantánea de los ficheros afectados. Si después de la instantánea, se cambia uno de los ficheros antes de registrarlos, es necesario añadir el fichero de nuevo al índice para registrar los nuevos cambios. Añada todos los ficheros y subdirectorios del directorio actual al *índice del repositorio git*. (`git add .`)

19. Registre (*commit*) los cambios del índice en el repositorio git local.
    (`git commit -m "First commit"`)

20. Muestre el fichero con los registros realizados en el repositorio hasta el momento. (`git log`)

21. Modifique los ficheros de directorio actual. (

```
echo "Hola desde el fichero test01" > test01
echo "Hola desde el fichero test02" > test02
```

)

22. Compruebe las diferencias entre los ficheros anteriores y los nuevos. (`git diff`)

23. Registre (*commit*) los cambios del índice en el repositorio git local. La opción `-a` permite registrar los cambios de los ficheros modificados, pero no añade ficheros nuevos automáticamente al índice. (`git commit -a -m "Diciendo hola"`)

24. Modifique los ficheros del directorio actual. (

```
echo "Adios desde el fichero test01" > test01
echo "Adios desde el fichero test02" > test02
```

)

25. Muestre el estado del repositorio git local, esto es, qué ficheros han cambiado, cuáles son nuevos y cuáles han sido borrados. (`git status`)

26. Muestre las diferencias entre los ficheros sin registrar y los del último registro. (`git diff`)

27. Añada los cambios al *índice del repositorio git* y regístrelos.
    (`git add . && git commit -m "Maaaas cambios - con un error sintáctico en el mensaje"`)

28. Muestre la historia de los distintos registros (*commits*) en la rama actual. (`git log`)

29. Arregle el error en el mensaje del último registro (*commit*) del apartado 27.
    (`git commit --amend -m "Más cambios - ahora sin errores"`)

30. Cree un fichero y póngalo bajo el control de versiones. (

    ```
    touch sinsentido.txt
    git add . && git commit -m "se ha creado un nuevo fichero sin sentido"
    ```

)

31. Elimine el fichero del directorio. (`rm sinsentido.txt`)

32. Añada los cambios al *índice del repositorio git* y regístrelos.
    (`git add . && git commit -m "se ha eliminado el fichero sinsentido.txt"`)
    ¿Qué sucede? ¿Por qué no funciona?

33. Añada los cambios al *índice del repositorio git* y regístrelos con la opción -A. Con esta opción se consigue borrar un fichero de la instantánea de `git`.
    (`git add -A . && git commit -m "se ha eliminado el fichero sinsentido.txt"`)

34. ¿Qué es `GitHub`?

35. Cree una cuenta en *GitHub*.

    a) Abra en el navegador el sitio de GitHub: http://github.com
    b) Pulse el botón verde que aparece en pantalla.
    c) Introduzca como Nombre de Usuario su aluXXXXXXXXX
    d) Introduzca su dirección de correo electrónico institucional: aluXXXXXXXXX@ull.edu.es
    e) Introduzca su contraseña (utilice la misma que la de su cuenta institucional)
    f) Pulse el botón verde para crear la cuenta.

36. Cree un repositorio en *GitHub*

    a) En la barra de usuario, en la esquina superior derecha de la página, haga clik en el icono de "Crear un repositorio nuevo" (*Create a New Repo*).
    b) Introduzca el nombre `prct02`
    c) Seleccione que quiere hacer el repositorio público.
    d) NO seleccione la casilla de crear el fichero README.md.
    e) Pulse el botón para crear el repositorio (*Create repository*)

37. Cree un repositorio remoto con nombre corto *origin*
    (`git remote add origin git@github.com:aluXXXXXXXX/prct02.git`)

38. Modifique el fichero `test02` (`echo "Se ha añadido un repositorio remoto" > test02`)

39. Ponga los cambios bajo el control de versiones.
    (`git commit -a -m "Esta es una prueba para el nuevo original remoto"`)

40. Empuje los cambios en el repositorio remoto denominado *origin*.
    (`git push -u origin master`)

41. Muestre los detalles del repositorio remoto denominado *origen*. (`git remote show origin`)

42. Muestre los repositorios remotos que están definidos. (`git remote -v`)

43. Sitúese en el directorio HOME y cree un directorio temporal denominado `tmp`.
    (`cd ~; mkdir tmp`)

44. Sitúese en el directorio temporal y clone el repositorio remoto. (

```
 cd ~/tmp
git clone git@github.com:aluXXXXXXXX/prct02.git/ .
```

)

45. Sitúese en el directorio de la práctica actual, es decir, en `prct02` y realice algunos cambios. (

```
 cd ~/TE/prct02
 echo "Hola, hola. Estoy en el repositorio de la práctica 2" > test01
 echo "Adios, adios. Estoy en el repositorio de la práctica 2" > test02
```

)

46. Registre (*commit*) los cambios en el repositorio. (`git commit -a -m "Algunos cambios"`)

47. Empuje los cambios en el repositorio remoto. (`git push -u origen master`)

48. Sitúese en el directorio temporal (`tmp`) que ha creado en el directorio HOME. (`cd ~/tmp`)

49. Extraiga los últimos cambios del repositorio remoto en el clon temporal que ha creado.
    (`git pull`)

50. Realice algún cambio en el clon. (`echo "Un cambio" > test01`)

51. Registre (*commit*) los cambios en el clon. (`git commit -a -m "Un cambio en el clon"`)

52. Empuje los cambios realizados en el clon en el repositorio remoto. (`git push origin master`) Cuando se clona un repositorio se crea automáticamente un repositorio con nombre *origin*.

53. Sitúese en el primer repositorio, es decir, en `prct02`. (`cd ~/TE/prct02`)

54. Extraiga los cambios del repositorio remoto.
    (`git pull git@github.com:aluXXXXXXXX/prct02.git`)

55. Compruebe que los cambios que se realizaron en el clon, también están en el original.
    (`git status`)

56. Escriba la dirección del repositorio que ha creado en GitHub en la tarea habilitada en el campus virtual.

57. Cierre la sesión.