Ejercicios básicos de Git

0. *Directorio de trabajo*

Al igual que hicimos con svn, crearemos las carpetas "home/usuario/GIT/Básico". Sitúate dentro de la carpeta "Básico" para realizar esta lista de ejercicios con git.

1. *Configuración*

Como git guarda nuestro nombre y dirección de correo electrónico cada vez que realizamos un commit, el primer paso es decirle a git cuáles son. Ya que estás en el proceso de configuración, indica que tu editor por defecto es "emacs".

2. *Creando un proyecto "Hello, world!"*

En el directorio de trabajo, crea una carpeta llamada "ProyectoWorld", luego crea un fichero que se llame "hello.txt" que contenga: "Hello, world!". Crea un repositorio git de este directorio, añade el fichero al mismo y luego guárdalo en el repositorio (mensaje: "Primer commit"). Tras esto, comprueba el estado del repositorio.

3. *Haciendo y añadiendo cambios en el proyecto*

Procede a cambiar el fichero "hello.txt" y en vez de contener: "Hello, world!", va a contener "Hello, folks!". Guarda los cambios y comprueba el estado del repositorio. Indica a git que añada dichos cambios y comprueba de nuevo el estado del repositorio. Finalmente guarda los cambios en el repositorio (mensaje: "Modificando hello.txt").

4. *Commits separados*

En el directorio "ProyectoWorld", crea los siguientes ficheros: "hola.txt", "bonjour.txt", "ciao.txt" y "hallo.txt". Cada uno debe contener (respectivamente): "¡Hola, mundo!", "Bonjour tout le monde!", "Ciao, mondo!" y "Hallo, welt!". Añade los ficheros "hola.txt" y "ciao.txt" y guárdalos en el repositorio (mensaje: "Saludos latinos") y comprueba el estado del repositorio. A continuación añade el fichero "bonjour.txt" y comprueba el estado del repositorio. Modifica el fichero "bonjour.txt" y elimina las palabras "tout le", comprueba el estado del repositorio. Indica a git que añada dichos cambios y comprueba de nuevo el estado del repositorio. Guarda los cambios de este fichero en el repositorio (mensaje: "Saludo francés") y por último añade el fichero "hallo.txt" y guarda el fichero en el repositorio (mensaje: "Saludo alemán").

5. *Cambios en varias líneas*

Vuelve al fichero "hallo.txt", añade tras el mensaje que contiene "Hallo, welt" (cada frase en una línea): "Hallo an alle!" y "Hallo zusammen!". Comprueba el estado del repositorio. Añade ahora antes de "Hallo,welt" (cada frase en una línea): "Hallo Ihr Lieben!" y "Hallo Leute!" y elimina "Hallo, welt". Visualiza las diferencias ¿qué observas? Crea otro fichero que se llame "hei.txt" que contenga la frase: "hei maailma". Añade los cambios con la

instrucción "`git add .`" y comprueba el estado del repositorio. Finalmente guarda los cambios de este fichero en el repositorio (mensaje: "Más saludos europeos").

6. *El historial*

Visualiza el historial de tu repositorio. Comprueba los resultados de las siguientes entradas:

```
git log --pretty=oneline --max-count=2
git log --pretty=oneline --since='5 minutes ago'
git log --pretty=oneline --until='5 minutes ago'
git log --pretty=oneline --author=<your name>
git log --pretty=oneline --all
```

¿Cómo sería la entrada para visualizar los cambios realizados en la última semana? ¿Y si quisieras ver los cambios que TÚ has realizado la última semana?

7. *Opciones del historial*

¿Cómo buscarías los commits realizados que contuviesen el mensaje "Saludo"?

8. *Formatos en el historial*

Si quisiéramos visualizar con un formato a la fecha del listado de cambios del historial, escribiríamos lo siguiente:

```
git log --all --pretty=format:"%h %cd %s (%an)"
```

--pretty="..." defines the format of the output.
%h is the abbreviated hash of the commit
%d are any decorations on that commit (e.g. branch heads or tags)
%ad is the author date
%s is the comment
%an is the author name
--graph informs git to display the commit tree in an ASCII graph layout
--date=short keeps the date format nice and short

¿Qué escribirías para obtener un formato como el siguiente?

```
* 55d2671 2012-12-06 | Saludo frances [Lorena]

* 4f5aea5 2012-12-06 | Saludos latinos [Lorena]

* 23170f3 2012-12-06 | Modificando hello.txt [Lorena]

* 5803e4b 2012-12-06 | Primer Commit [Lorena]
```

9. *Alias*

Cambia el alias a las siguientes instrucciones en tu directorio $home:
El alias de commit: ci
El alias de status: st

El alias de log con el formato del ejercicio anterior: hist

Para ello dirígete al fichero .gitconfig del directorio $home y escribe:

```
[alias]

  ci = commit
  st = status
  hist = log --pretty=format:…
```

10. *Alias del shell*

Dirígete al fichero .profile del directorio $home y añade las siguientes líneas (algunas de las instrucciones todavía no se han visto)

```
alias gst='git status '
alias ga='git add '
alias gb='git branch '
alias gc='git commit'
alias gd='git diff'
alias go='git checkout '
alias gk='gitk --all&'
alias gx='gitx --all'

alias got='git '
alias get='git '
```

Para recargar la información del fichero escribe en consola "source .profile" ¿Qué diferencia existe entre escribir en consola 'gst' y 'git st'?

11. *Etiquetando*

Supongamos que con los ficheros que tenemos en nuestro repositorio, hemos llegado a la primera versión de nuestra aplicación (v1.0.0). Etiqueta esta versión de nuestro repositorio con el mensaje "Creando la primera versión oficial.". Si queremos visualizar la información almacenada sobre una versión, (v1.0.0 en nuestro ejemplo) tenemos que escribir "`git show v1.0.0`".

12. *Volviendo al estado original*

Para comprobar cómo funciona el siguiente comando, primero haremos una prueba: queremos volver a la versión original del fichero "hello.txt", que contenía la frase "Hello, world!". Para ello escribiremos:

```
git checkout <hash> //El sha1 del fichero en esa versión
```

Para ver el hash de ese primer commit tendrás que ver el historial. Comprueba el contenido del fichero "hello.txt" y el contenido de nuestro directorio de trabajo "ProyectoWorld". Tras la comprobación, realizaremos lo siguiente y seguiremos con el siguiente ejercicio: vuelve a la versión original del fichero "hallo.txt".

13. *Etiquetando una versión anterior*

Ahora que estamos en la versión en la que todavía no hemos introducido más saludos en el fichero "hallo.txt", vamos a etiquetar esta versión como "v1.0.0-beta". Una vez etiquetados correctamente, vuelve a la versión "v1.0.0". Comprueba las etiquetas disponibles utilizando el comando de git "`git tag`". Visualiza el historial del repositorio ¿Ves las etiquetas? Si no es así, vuelve al ejercicio 8 y comprueba que te falta.

14. *Nuevo fichero para la siguiente versión*

Ya tenemos etiquetada nuestra primera versión de nuestro programa, para ver en qué versión estamos (en el caso de que nos hayamos olvidado) debemos escribir "`git describe --tags`". Tras escribir este comando en consola, crea un nuevo fichero que se llame "hej.txt" que contenga "Hej, verden!" y guárdalo en el repositorio (mensaje: Fichero hej.txt"). ¿Qué ocurre cuando quieres ver la versión en la que estás tras añadir este nuevo fichero? ¿Qué crees que significa?

15. *Etiquetando desde master*

Estando en el último commit de nuestro proyecto (HEAD), se nos ocurre etiquetar el primer commit que hicimos, en el que sólo teníamos el fichero "hello.txt" como v0.0.0 "Versión de prueba". ¿Cómo lo harías sin tener que cambiar a ese antiguo estado?

16. *¿Quién hizo un commit de esto?*

Cuando queremos saber quién introdujo un fallo en nuestra aplicación o quién implementó un trozo de código eficiente, podemos utilizar el comando "`git blame`". Aunque ahora mismo estamos trabajando en local y el responsable de los commits solo somos nosotros, utiliza este comando con el fichero "hallo.txt".

17. *Escondiendo los cambios*

Edita el fichero "hola.txt" añadiendo las siguientes líneas: "¡Hola a todos!", "¡Hola a todo el mundo!"… Todavía no hemos terminado de hacer los cambios en este fichero, pero nos hemos dado cuenta que hay un error, o que nos falta añadir código, en el fichero "hello.txt". Como no queremos que se nos olvide, y no queremos dejar nuestros cambios a la mitad, vamos a "esconder" los cambios de "hola.txt" para luego seguir por donde íbamos. Guarda los cambios del fichero y comprueba el estado del repositorio. Para esconder los cambios usamos "`git stash save "Añadiendo saludos en hola.txt"`". Comprueba que volvemos a estar de nuevo en el estado anterior de cambiar "hola.txt". Abre el fichero "hello.txt" e inserta las líneas "Hello everybody!", "Hi, all!"… Otra vez nos hemos dado cuenta que en "bonjour.txt" también tenemos que añadir código; esconde de nuevo los cambios (mensaje: "Añadiendo saludos en hello.txt"). Para comprobar que los cambios escondidos se están guardando correctamente escribe "`git stash list`". Añade en "bonjour.txt": "Salut!" y "Salut à tous!". Añade los cambios y guárdalos en el repositorio (mensaje: "Añadiendo saludos en bonjour.txt"). Para retomar los cambios escondidos por segunda vez: "`git stash apply stash@{Num}`" y podemos seguir añadiendo en "hello.txt" "Hello, guys!" y "Hi, mate!". Añade los cambios y guárdalos en el repositorio

(mensaje: "Añadiendo saludos en hello.txt"). Finalmente recupera los primeros cambios escondidos y añade en "hola.txt": "¿Qué tal?" y "¡Buenas!". Guarda los cambios en el repositorio (mensaje: "Añadiendo saludos en hola.txt"). Comprueba que están todavía la lista de cambios, para borrar un elemento de la lista "`git stash drop stash@{Num}`" y para borrarla entera escribiremos "`git stash clear`". Borra la lista entera. Comprueba

18. *Deshaciendo cambios locales*

Asegurándonos que estamos en el último commit, procedemos a modificar el fichero "ciao.txt". Escribe "Ciao a todos!" y guarda los cambios del fichero. Comprueba el estado del repositorio y nos aparecerá que el fichero "ciao.txt" ha sido modificado. Tras esto, nos damos cuenta que esta línea está mal y queremos volver al estado anterior. En vez de abrir el fichero y deshacer los cambios, podemos volver al estado anterior del fichero con "`git checkout ciao.txt`". Comprueba el estado del repositorio y el contenido del fichero.

19. *Deshaciendo cambios antes del commit*

Vuelve a hacer el mismo procedimiento de antes, pero además de guardar los cambios en el fichero "ciao.txt", añádelos al repositorio y comprueba su estado. Tal y como nos pasó antes, ahora queremos deshacer los cambios que hemos incluido en este fichero. Para ello, aunque el mismo repositorio nos lo dice, utilizaremos "`git reset HEAD ciao.txt`". De este modo volvemos al estado justo antes de añadirlo, luego para volver a su estado inicial, haremos lo mismo que en el ejercicio anterior.

20. *Deshaciendo cambios después del commit*

Realiza otra vez los cambios en "ciao.txt", añade los cambios en el repositorio y guárdalos en él (mensaje: Modificaciones en ciao.txt). Para revertir este commit existe el comando "`git revert HEAD`", esto lanzará el editor "emacs" (o el editor que hayas configurado por defecto) con el estado antes de hacer el commit, dándonos la posibilidad de cambiar el mensaje del commit para indicar que se está deshaciendo (cambia el mensaje, que es la primera línea que aparece, a: Ups!, commit incorrecto). Guarda los cambios en este fichero y comprueba el historial verás cómo aparecen ambos. ¿Qué estado tiene el repositorio?, ¿Qué contiene "ciao.txt"?

21. *Eliminar ficheros del área de preparación*

Crea un nuevo fichero que se llame "erase.txt", escribe en él "Fichero de prueba a borrar", y añade los cambios en el repositorio. Ahora vamos a eliminar el fichero "erase.txt", lo podemos realizar de dos modos, el primero de ellos es escribiendo "`git rm -f erase.txt`", comprueba que se ha eliminado del área de preparación y también del directorio de trabajo. ¿Cómo se te ocurre que podría ser la segunda forma de eliminar el fichero tanto del directorio de trabajo como del área de preparación? Compruébalo (créate otro fichero "erase.txt" para hacer la prueba).

22. *Eliminar y renombrar de ficheros*

Crea un nuevo fichero que se llame "hóla.txt", escribe en él "Olá mundo", y añade los cambios. Como puedes comprobar nos hemos confundido en el nombre del fichero, para solventarlo, debemos escribir "`git rm --cached hóla.txt`" y comprueba que sigue en el directorio de trabajo. Renombra el fichero como "óla.txt", añade los cambios y guarda el fichero en el repositorio (mensaje: Saludo portugués).

23. *Moviendo ficheros*

En git también existe el comando mv. Crea un directorio dentro de ProyectoWorld que se llame "España", y mueve el fichero "hola.txt" al mismo utilizando el comando "`git mv hola.txt España`". Comprueba el estado ¿Qué pasa? ¿Qué es lo que se comunica a git? Guarda los cambios (mensaje: Directorio España).
Crea un fichero en el directorio de trabajo actual (ProyectWorld) que se llame "kaixo.txt" y que contenga la frase "kaixo mundua", añade el fichero al repositorio y guárdalo en él (mensaje: saludo euskera). ¿Qué instrucciones tenemos que seguir para no utilizar "`git mv`" para que este fichero esté dentro del directorio España?. Finalmente guarda los cambios en el repositorio (mensaje: kaixo.txt en España).

24. *Eliminando etiquetas*

Tras añadir los anteriores ficheros a nuestro repositorio, vamos a etiquetar esta versión como "v1.0.1" (mensaje: Idiomas castellanos). Nos hemos dado cuenta que falta un fichero, todavía no creado, para que sea la versión "v1.0.1". Accede a la ayuda del comando "tag" para averiguar cómo podemos eliminar la etiqueta anterior. Tras eliminarla, y comprobar que se ha eliminado, crea el fichero "holaCat.txt" que contenga la frase "hola món". Éste debe guardarse en la carpeta "España", añádelo al repositorio y guarda los cambios en el repositorio (mensaje: saludo catalán). Ahora sí podemos etiquetar como "v1.0.1" (con el mismo mensaje anterior).

25. *Arreglando commits*

Seguimos añadiendo saludos a nuestro repositorio. Crea un nuevo fichero que se llame "halo.txt" y que contenga la frase "halo svijet". Añade el fichero y guarda los cambios en el repositorio (mensaje: saludo). ¡Vaya! Otra vez nos hemos confundido y el commit no lo hemos hecho correctamente. Aprovechamos antes de arreglar el commit para añadir en el fichero halo.txt la frase "Pozdrav svima", añade los cambios al repositorio. Busca en la ayuda (`git help commit`) qué debemos hacer en este siguiente commit para arreglar el anterior (con el mensaje saludo), para que no aparezca en el historial y ponerle otro mensaje. El mensaje del nuevo commit será "saludo croata". Tras arreglar el commit, comprueba en el historial que efectivamente no aparece el commit anterior con el mensaje "saludo".

26. Referencia a objetos (diferencias)

Utilizando las referencias a objetos que hemos visto, visualiza las diferencias entre:

- HEAD y dos commits antes del HEAD

- 4 commits antes del master y HEAD

27. *Referencia a objetos (visualización)*

Al igual que antes, utilizando las referencias a objetos que hemos visto, visualiza:

- Lo que hiciste la semana pasada (sólo el área de preparación)
- El master de hace 3 cambios
- El master de hace 6 días

28. *Referencia a objetos (historial)*

Ahora queremos el historial desde:

- Hace 7 commits
- El lunes de la semana pasada, hasta el día de hoy
- Su creación, del fichero "hello.txt"

29. *Búsquedas*

Busca en el repositorio:

- Ficheros que contengan "mundo"
- Ficheros que contengan "mundo", pero que aparezca el nombre del fichero y la línea donde está escrita la palabra.

30. *Visualización del repositorio*

Para visualizar de forma gráfica nuestro repositorio, podemos utilizar el comando "`git instaweb`". Visualiza tu repositorio e investiga por la web.

Fuentes utilizadas para la redacción de algunos de los ejercicios:

http://gitimmersion.com/index.html

http://gitref.org/index.html

http://gitready.com/