

Tema: II git Prácticas.

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

Curso 2014-2015 , Copyleft © 2011-2015 .
Reproducción permitida bajo los términos de la licencia de
documentación libre GNU.



1 / 18

Contenido

- 1 Primeros pasos con git
- 2 Modificación de archivos (I)
- 3 Modificación de archivos (II)
- 4 Añadido de archivos
- 5 Log del repositorio
- 6 Borrado de archivos
- 7 Deshacer acciones (I)
- 8 Deshacer acciones (II)
- 9 Renombrar ficheros
- 10 Ramas: Actual, crear, cambiar de rama, diferencias
- 11 Ramas: Diferencias a nivel de commits
- 12 Ejercicio con ramas
- 13 Ramas: mezclar, rebasar
- 14 Clonado de repositorios
- 15 Objetivos...
- 16 Entrega...



2 / 18

Primeros pasos con git

```
1  # Comprobamos la version instalada
   git --version
3
   # Partimos del codigo de la practica 1 del juego de
   # adivinar un numero
5  # Creamos el repositorio inicial para poner bajo
   # control de versiones
   # los archivos de ese proyecto.
7  cd Juego; git init
9
   # Anyadimos al repositorio los archivos del
   # directorio actual y subdirectorios
   git add .
11 # Comprobamos el estado despues de anyadirlos
   git status
13 # Confirmamos que los hemos anyadido (commit)
   git commit -m 'Primer commit.' -m "Descripcion
   detallada." -a
```



3 / 18

Modificación de archivos (I)

```
   # Modificamos algun archivo
2  gedit main.vala Makefile
4
   # Vemos las diferencias de la copia de trabajo con
   # el repositorio
   git diff
6
   # aceptamos los cambios, podemos hacerlo de varias
   # maneras:
8
   # 1- Archivo a archivo
10 git commit -m "Correccion de errores en el prog.
   ppal." main.vala
   git commit -m "Anyadido objetivo de depuracion."
   Makefile
```



4 / 18

```

1  # 2- Preparamos el 'escenario' o 'stage' que
    formara el commit
    # A
3  git add Makefile
   git commit -m "Modificado Makefile."
5  # B
   git add Makefile main.vala
7  git commit -m "Un solo commit con varios archivos."
    
```

```

1  # Creamos el archivo
   gedit autores.txt
3
   # Lo anyadimos...
5  git add autores.txt

7  # Confirmamos el anyadido...
   git commit -m "fichero autores.txt" autores.txt
9
   # Que archivos hay bajo control de versiones?
11 git ls-files
    
```

Log del repositorio

Borrado de archivos

```

1  # Que operaciones hemos hecho sobre el repositorio?
   git log
3
   commit dbafcc771eb504a541db32d7a7b5287a470516aa    # HEAD
5  Author: hada <hada@dlsi.ua.es>
   Date:   Fri Jan 20 18:26:22 2012 +0100
7
   fichero autores.txt
9
   commit 18442c5369b1f6f7920a4ebea39ec38bf9b62555    # HEAD~1
11 Author: hada <hada@dlsi.ua.es>
   Date:   Fri Jan 20 17:54:48 2012 +0100
13
   Cambios al makefile.
15
   commit e3911a1778c405ecce14bae1c1a97ec81832242b    # HEAD~2
17 Author: hada <hada@dlsi.ua.es>
   Date:   Fri Jan 20 17:54:12 2012 +0100
19
   # Cada commit tiene su numero SHA-1 propio
21 # Los numeros SHA-1 se pueden resumir hasta que sean distinguibles
   # Podemos referirnos al ultimo commit hecho (el primero de la lista) por el
     alias: HEAD
23
   # Etiquetamos HEAD con v1.0 y HEAD~1 con v0.9
25 git tag v1.0
   git tag v0.9 HEAD~1
27 git tag -l
   # Lo visualizamos graficamente con gitk
29 gitk
    
```

```

1  # Borramos el archivo
   git rm autores.txt
3
   # Vemos el estado del repositorio
5  git status

7  # Confirmamos el borrado...
   git commit -m "fichero autores.txt borrado"
9
   # Que dice el log del repositorio?
11 git log

13 commit 5a369acdefc3c1d28c7d1c9561f7bb26d9dae3
   Author: hada <hada@dlsi.ua.es>
15 Date:   Fri Jan 20 18:41:56 2012 +0100

17 Archivo autores.txt borrado.
    
```

Hay varias formas de hacerlo, usaremos 'git revert':

```

1 # Borramos el archivo
2 git rm autores.txt
3 git commit ...
4
5 # NOOO!!! es un error!!!, lo puedo recuperar?
6 # Que dice el log del repositorio?
7 git log
8
9
10 commit 5a369acdefc3c1d28c7d1c9561f7bb26d9dae3d3 # HEAD
11 Author: hada <hada@dlsi.ua.es>
12 Date: Fri Jan 20 18:41:56 2012 +0100
13
14 Archivo autores.txt borrado.
15
16 commit dbafcc771eb504a541db32d7a7b5287a470516aa # HEAD~1
17 Author: hada <hada@dlsi.ua.es>
18 Date: Fri Jan 20 18:26:22 2012 +0100
19
20 Correccion de errores.

```

```

1 # revert deshace un commit creando un commit 'inverso'
2 # con la opcion '-n' hace todo menos crear el commit 'inverso'
3 # lo hacemos asi para ver paso a paso el estado del repositorio
4 git revert -n HEAD
5
6 git status
7
8 # On branch master
9 # Changes to be committed:
10 #   (use "git reset HEAD <file>..." to unstage)
11 #
12 #       new file:   autores.txt
13 #
14
15 # Que dice AUN el log del repositorio?
16 git log
17
18 commit 5a369acdefc3c1d28c7d1c9561f7bb26d9dae3d3 # HEAD
19 Author: hada <hada@dlsi.ua.es>
20 Date: Fri Jan 20 18:41:56 2012 +0100
21
22 Archivo autores.txt borrado.
23
24 # PREGUNTA: Que habria que hacer ahora?
25
26 # Otra forma de hacerlo: git reset
27 # Investiga que hace y trata de usarlo para resolver esta situacion.

```

Renombrar ficheros

Ramas: Actual, crear, cambiar de rama, diferencias

```

1 git mv autores.txt AUTHORS
2 git status
3
4 # On branch master
5 # Changes to be committed:
6 #   (use "git reset HEAD <file>..." to unstage)
7 #
8 #       renamed:   autores.txt -> AUTHORS
9 #
10
11 # PREGUNTA: Que habria que hacer ahora?
12
13 # Solo un commit que abarque la operacion de renombrado
14 git commit -m "Renombrado archivo autores.txt a AUTHORS."

```

```

1 # Comprobar la actual, remotas o todas
2 git branch [-r] [-a]
3 * master
4
5 # Crear una rama llamada 'devel' basada en la actual
6 git branch devel
7 # Ramas existentes, en la que estamos lleva un '*'
8 git branch
9 devel
10 * master
11 # Cambiar a 'devel'
12 git checkout devel
13 Switched to branch 'devel'
14
15 # Comprueba que estas en la rama 'devel'
16 # Haz cambios en AUTHORS aqui y guardalos (commit)
17 # cambiate de nuevo a 'master'
18 git checkout master
19 Switched to branch 'master'
20
21 # Diferencias entre 'devel' y 'master' (estamos en master)
22 git diff devel
23 # o tambien
24 git diff master devel
25 git diff devel master

```

Lo hacemos con 'git log [-p]'

```
1 # Podemos verlas de varias maneras
2 git log master..devel # rama origen es master, rama destino es devel
3 git log devel..master
4 git log master..
5 git log ..master # rama origen es master, rama destino la actual
6 # rama origen la actual, rama destino master
7 # tambien podemos emplear show-branch, p.e.:
8 git show-branch master devel
9 ! [master] Renombrado archivo.
10 * [devel] Cambios en README.
11 ---
12 * [devel] Cambios en README.
13 +* [master] Renombrado archivo.
```

- A estas alturas debes haber creado la rama 'devel'.
- Debes estar trabajando en ella y no en 'master'.
- Modifica los archivos necesarios para que en el juego se permita un número máximo de intentos para adivinar el número, por ejemplo 3.
- Al finalizar esta parte debes tener todos los cambios incorporados a la rama 'devel' (haber hecho el o los commits necesarios)

Ramas: mezclar, rebasar

- Supongamos que queremos pasar nuestros cambios de la rama 'devel' a 'master'.
- Tenemos dos opciones: **mezclarlas** o **rebasarlas**

```
1 # Opcion Mezcla
2 git merge devel
3 Updating ddeebdc..646eba9
4 Fast-forward
5 README | 1 +
6 1 files changed, 1 insertions(+), 0 deletions(-)
7
8 # Opcion Rebase
9 git rebase devel
10 First, rewinding head to replay your work on top of it...
11 Fast-forwarded master to devel.
```

Clonado de repositorios

- Nos permite 'copiar' un repositorio local o remotamente (http/s, ssh, git, git+ssh).
- Se establece un enlace entre ambos que permite realizar operaciones **pull**, **fetch**, **merge** y **push**.

```
1 # clona el repositorio con el codigo de la practica0
2 # lo hacemos en un directorio llamado practica1b
3 git clone Juego Juegob
4 Cloning into 'Juegob'...
5 done.
6
7 cd Juegob; ls # debe estar todo copia-trabajo + repositorio (.git)
8
9 # Si intentamos hacer un push obtendremos un error. El repositorio origen
10 # contiene copia de trabajo (no es bare -solo directorio .git-)
```

El clonado de repositorios es muy util cuando se clona un repositorio solo con datos (bare) y no con copia de trabajo.

El alumno sabe:

- ☐ Crear un repositorio con git y añadir los archivos que pondrá bajo control de versiones.
- ☐ Hacer commits de las acciones que lleve acabo (modificaciones en archivos, añadido de nuevos archivos, borrado de archivos, renombrado de archivos, etc. . .)
- ☐ Ver el 'log' de las acciones llevadas a cabo.
- ☐ Poner etiquetas a una determinada versión de los archivos.
- ☐ Deshacer acciones, p.e. recuperar un archivo borrado.
- ☐ Crear ramas, cambiar de rama, ver diferencias entre las ramas.
- ☐ Importar cambios de una rama a otra.
- ☐ Clonar repositorios.

- La entrega de esta práctica consiste en el directorio de trabajo (Juego) comprimido en un fichero llamado juego.tgz.
- Recuerda que este directorio ya contiene la copia de trabajo y el directorio '.git'.