Tema: II
git
Prácticas.

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

lsi Department of Software and Computing Systems

Contenido

Primeros pasos con git

```
1   # Comprobad 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
```

Modificación de archivos (I)

```
# Modificamos algun archivo
gedit main.vala Makefile

# Vemos las diferencias de la copia de trabajo con el repositorio
git diff

# aceptamos los cambios, podemos hacerlo de varias maneras:

# 1- Archivo a archivo
git commit -m "Correccion de errores en el prog. ppal." main.vala
git commit -m "Anyadido objetivo de depuracion." Makefile
```

Modificación de archivos (II)

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

Añadido de 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

```
# Que operaciones hemos hecho sobre el repositorio?
git log

commit dbafcc771eb504a541db32d7a7b5287a470516aa    # HEAD
Author: hada <hada@dlsi.ua.es>
Date:   Fri Jan 20 18:26:22 2012 +0100

fichero autores.txt

commit 18442c5369b1f6f7920a4ebea39ec38bf9b62555    # HEAD~1
Author: hada <hada@dlsi.ua.es>
Date:   Fri Jan 20 17:54:48 2012 +0100

Cambios al makefile.

commit e3911a1778c405ecce14bae1c1a97ec81832242b    # HEAD~2
Author: hada <hada@dlsi.ua.es>
Date:   Fri Jan 20 17:54:12 2012 +0100

# Cada commit tiene su numero SHA-1 propio
# 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

# Etiquetamos HEAD con v1.0 y HEAD~1 con v0.9
git tag v1.0
git tag v0.9 HEAD~1
git tag -l
# Lo visualizamos graficamente con gitk
gitk
```

Borrado de archivos

```
# Borramos el archivo
git rm autores.txt

# Vemos el estado del repositorio
git status

# Confirmamos el borrado...
git commit -m "fichero autores.txt borrado"

# Que dice el log del repositorio?
git log

commit 5a369acdefc3c1d28c7d1c9561f7bb26d9daead3
Author: hada <hada@dlsi.ua.es>
Date:   Fri Jan 20 18:41:56 2012 +0100

Archivo autores.txt borrado.
```

Deshacer acciones (I)

```
    # Borramos el archivo
 2  git rm autores.txt
    git commit ...
 4
    # NOOO!!! es un error!!!, lo puedo recuperar?
 6
    # Que dice el log del repositorio?
 8  git log

10  commit 5a369acdefc3c1d28c7d1c9561f7bb26d9daead3   # HEAD
    Author: hada <hada@dlsi.ua.es>
12  Date:   Fri Jan 20 18:41:56 2012 +0100

14  Archivo autores.txt borrado.

16  commit dbafcc771eb504a541db32d7a7b5287a470516aa   # HEAD~1
    Author: hada <hada@dlsi.ua.es>
18  Date:   Fri Jan 20 18:26:22 2012 +0100

20  Correccion de errores.
```

Deshacer acciones (II)

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

```
1  # reverte deshace un commit creando un commit 'inverso'
   # con la opcion '-n' hace todo menos crear el commit 'inverso'
3  # lo hacemos asi para ver paso a paso el estado del repositorio
   git revert -n HEAD
5
   git status
7
   # On branch master
9  # Changes to be committed:
   #   (use "git reset HEAD <file>..." to unstage)
11 #
   #       new file:   autores.txt
13 #

15 # Que dice AUN el log del repositorio?
   git log
17
   commit 5a369acdefc3c1d28c7d1c9561f7bb26d9daead3   # HEAD
19 Author: hada <hada@dlsi.ua.es>
   Date:   Fri Jan 20 18:41:56 2012 +0100
21
   Archivo autores.txt borrado.
23
   # PREGUNTA: Que habria que hacer ahora?
25
   # Otra forma de hacerlo: git reset
27 # Investiga que hace y trata de usarlo para resolver esta situacion.
```

Renombrar ficheros

```
1   git mv autores.txt AUTHORS
    git status
3   # On branch master
    # Changes to be committed:
5   #   (use "git reset HEAD <file>..." to unstage)
    #
7   #       renamed:    autores.txt -> AUTHORS

9   # PREGUNTA: Que habria que hacer ahora?

11  # Solo un commit que abarque la operacion de renombrado
    git commit -m "Renombrado archivo autores.txt a AUTHORS."
```

Ramas: Actual, crear, cambiar de rama, diferencias

```
# Comprobar la actual, remotas o todas
git branch [-r] [-a]
* master

# Crear una rama llamda 'devel' basada en la actual
git branch devel
# Ramas existentes, en la que estamos lleva un '*'
git branch
devel
* master
# Cambiar a 'devel'
git checkout devel
Switched to branch 'devel'

# Comprueba que estas en la rama 'devel'
# Haz cambios en AUTHORS aqui y guardalos (commit)
# cambiate de nuevo a 'master'
git checkout master
Switched to branch 'master'

# Diferencias entre 'devel' y 'master' (estamos en master)
git diff devel
# o tambien
git diff master devel
git diff devel master
```

Ramas: Diferencias a nivel de commits.

Lo hacemos con '`git log [-p]`'

```
# Podemos verlas de varias maneras
git log master..devel # rama origen es master, rama destino es devel
git log devel..master
git log master..      # rama origen es master, rama destino la actual
git log ..master      # rama origen la actual, rama destino master

# tambien podemos emplear show-branch, p.e.:
git show-branch master devel
! [master] Renombrado archivo.
 * [devel] Cambios en README.
--
 * [devel] Cambios en README.
+* [master] Renombrado archivo.
```

Ejercicio con ramas

- 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
   git merge devel
3  Updating ddeebdc..646eba9
   Fast-forward
5   README |    1 +
    1 files changed, 1 insertions(+), 0 deletions(-)
7
   # Opcion Rebase
9  git rebase devel
   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
   # lo hacemos en un directorio llamado practica1b
3  git clone Juego Juegob
   Cloning into 'Juegob'...
5  done.

7  cd Juegob; ls  # debe estar todo copia-trabajo + repositorio (.git)

9  # Si intentamos hacer un push obtendremos un error. El repositorio origen
   # 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.

Objetivos...

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.

Entrega...

- 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`'.