Soluciones: Ejercicios nivel avanzado de Git

0. *Directorio de trabajo*

```
$ cd GIT
$ git clone https://github.com/lorgut/EvolSoft Avanzado
$ cd Avanzado
```

1. *Preparación del entorno*

```
$ echo "Grupo de prácticas formado por el equipo NombreEquipo1:
alumno1 y alumno2 y por el equipo NombreEquipo2: alumno3 y
alumno4" > READMENombreEquipo1_NombreEquipo2
$ git init
$ git add .
$ git commit -m "Entorno de trabajo"
$ git checkout -b NombreEquipo1vsNombreEquipo2
```

Alumno4

```
$ git add . //Tras añadir los ficheros pb.c y makefile en el directorio
$ git commit -m "Ficheros pb.c y makefile añadidos"
```

2. *No queremos tantos commits (alumno 4)*

```
$ git rebase -i HEAD~2
```

Salta un editor y lo modificamos para que quede así:

```
pick 617c7d1 Entorno de trabajo
squash 8859208 Ficheros pb.c y makefile añadidos

# Rebase 2a07dc8..8859208 onto 2a07dc8
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log
message
#  x, exec = run command (the rest of the line) using shell
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
```

Guardamos los cambios (con el nombre del fichero que viene por defecto) y salimos de ese editor. Directamente sale otro editor en el que realizamos los siguientes cambios:

```
# This is a combination of 2 commits.
```

```
# The first commit's message is:

~~Entorno de trabajo~~
Rama del equipo NombreEquipo1, NombreEquipo2 versión inicial

~~# This is the 2nd commit message:~~

~~Ficheros pb.c y makefile añadidos~~

# Please enter the commit message for your changes. Lines
starting
# with '#' will be ignored, and an empty message aborts the
commit.
# Not currently on any branch.
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   Reame
#       new file:   makefile
#       new file:   pb.c
#
```

Guardamos los cambios en el fichero que viene por defecto, y salimos del editor.
Obtenemos el siguiente resultado:

```
~/avanzado$ git rebase -i HEAD~2
[detached HEAD 57d9b27] Rama del equipo NombreEquipo1 y
NombreEquipo2 versión inicial
 3 files changed, 124 insertions(+)
 create mode 100644 Reame
 create mode 100644 makefile
 create mode 100644 pb.c
Successfully rebased and updated
refs/heads/NombreEquipo1vsNombreEquipo2.
```

Comprobamos los cambios

```
~/avanzado$ git hist
* 57d9b27 Wed Apr 10 19:42:23 2013 +0200 | Rama del equipo
NombreEquipo1 y NombreEquipo2 versión inicial (HEAD,
NombreEquipo1vsNombreEquipo2) [Lorena Gutiérrez]
$ git push https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
```

3. *Creando la rama de tu equipo*

```
$ git pull https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2 //Situados en NombreEquipo1vsNombreEquipo2
$ git checkout –b NombreEquipo1
```

```
$ git branch
$ echo "Este es el entorno de trabajo del equipo NombreEquipo,
donde trabajan los alumnos: alumno1 y alumno2, en el que
almacenarán los ficheros de la clase de prácticas de la
asignatura Evolución del Software" > READMENombreEquipo1
$ git add READMENombreEquipo1
$ git commit –m "Readme añadido"
```

Con la orden *`ls` vemos el contenido de la rama y nos cambiamos de ramas con la orden de git* `"git checkout NombreRama"`

4. *Trabajando con un repositorio remoto*

Alumno1

```
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo1
```

Alumno3

```
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo2

$ git ls-remote --heads https://github.com/lorgut/EvolSoft.git
```

5. *Clave pública SSH*

Seguimos los pasos que nos indican en la web.

6. *Generando números*

Alumno3

```
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo2
$ emacs pb.c //Realizamos los cambios
$ git add pb.c
$ git commit –m "Generando números aleatorios para las bolas
blancas y la powerball"
```

Alumno4

```
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo2
$ emacs pb.c //Realizamos los cambios
$ git add pb.c
$ git commit –m "Función para controlar los números repetidos"
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo2
```

7. *Merge sin conflictos*

Alumno2

```
$ git ls-remote --heads https://github.com/lorgut/EvolSoft.git
$ git checkout -b NombreEquipo2
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo2
$ git diff NombreEquipo1vsNombreEquipo2 NombreEquipo2
$ git merge NombreEquipo2 //Situados en la rama
NombreEquipo1vsNombreEquipo2
```

```
$ git mergetool
$ git commit –m "Fusión con los cambios realizados por el equipo NombreEquipo2" //Si hubo conflictos en la fusión
$ emacs pb.c //Realizamos los cambios en el pb.c de su equipo
$ git add pb.c
$ git commit –m "Imprimiendo los números tras ordenarlos"
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo1vsNombreEquipo2
$ git checkout NombreEquipo1
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo1
```

Alumno1

```
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1vsNombreEquipo2
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1
$ git diff NombreEquipo1 NombreEquipo1vsNombreEquipo2
$ emacs pb.c & //Realiza los cambios en el pb.c de su equipo
$ git add pb.c
$ git commit –m "Función showing_results"
$ git merge NombreEquipo1 //Situados en la rama NombreEquipo1vsNombreEquipo2
$ git mergetool
$ git commit –m "Fusión con rama NombreEquipo1vsNombreEquipo2" //Si hubo conflictos en la fusión
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo1vsNombreEquipo2
```

8. *Merge*

```
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1vsNombreEquipo2
$ gid diff NombreEquipo1vsNombreEquipo2 NombreEquipo2
$ git merge NombreEquipo2 //Situados en la rama NombreEquipo1vsNombreEquipo2
$ git mergetool
$ git commit –m "Función checkwhiteballs fusionada" //Si hubo conflictos en la fusión
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo1vsNombreEquipo2
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1vsNombreEquipo2 //Situados en la rama NombreEquipo2
$ emacs pb.c //Realiza los cambios
$ git add pb.c
$ git commit –m "Usando defines"
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo1vsNombreEquipo2
$ git checkout NombreEquipo2
$ emacs pb.c //Realiza los cambios
$ git add pb.c
```

```
$ git commit –m "Usando defines"
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo2
```

9. *Alumno1 vs alumno3*

```
$ git pull https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1
```
//Para el alumno1
```
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo2
```
//Para el alumno3
`$ git merge NombreEquipo1vsNombreEquipo2` //Situados en la rama NombreEquipo2 para el alumno3
`$ git merge NombreEquipo1vsNombreEquipo2` //Situados en la rama NombreEquipo1 para el alumno1
`$ git emacs pb.c &` //Realiza los cambios
```
$ git add pb.c
$ git commit –m "Simulación de números premiados"
```
`$ git merge NombreEquipo2` //Situados en la rama NombreEquipo1vsNombreEquipo2 para el alumno3
`$ git merge NombreEquipo1` //Situados en la rama NombreEquipo1vsNombreEquipo2 para el alumno1
```
$ git add pb.c
$ git commit –m "Simulación de los números premiados por
NombreEquipo"
$ git push https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
```
Dependiendo del alumno que fuera el último en realizar el push, resolverá los cambios con la herramienta `tkdiff`, que aparece con la orden: `mergetool`. Los cambios realizados para la simulación de números premiados, se deja en la decisión del alumno en conservar el que quiera.

10. *Ficheros a ignorar (alumnos 2 y 4)*

```
$ git pull https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1
```
//Para el alumno2
```
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo2
```
//Para el alumno4
`$ git diff NombreEquipo1vsNombreEquipo2 NombreEquipo1` //Para el alumno2
`$ git diff NombreEquipo1vsNombreEquipo2 NombreEquipo2` //Para el alumno4
`$ git merge NombreEquipo1vsNombreEquipo2` //Situados en la rama NombreEquipo2 para el alumno4
`$ git merge NombreEquipo1vsNombreEquipo2` //Situados en la rama NombreEquipo1 para el alumno2

Alumno2

```
$ git checkout NombreEquipo1vsNombreEquipo2
$ emacs .gitignore //Realiza los cambios
$ git add .gitignore
$ git commit –m "Fichero .gitignore para ambos equipos"
$ git push https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
$ git status
```

Alumno4

```
$ touch pruebas
$ emacs info/exclude & //Realiza los cambios
```

Realiza varias pruebas y comprueba luego que no contempla el fichero "pruebas" con la orden `git status`.

11. *Fetch + merge (alumnos 1 y 3)*

Alumno1

```
$ git pull https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1
$ git diff NombreEquipo1vsNombreEquipo2 NombreEquipo1
$ git merge NombreEquipo1vsNombreEquipo2 //Situados en la rama
NombreEquipo1
$ emacs pb.c & //Realiza los cambios
$ git add pb.c
$ git commit –m "Función showing_results"
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo1
```

Alumno3

```
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo2
$ git fetch https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
$ git diff FETCH_HEAD NombreEquipo1vsNombreEquipo2
$ git merge FETCH_HEAD
$ git checkout -b NombreEquipo1
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1
$ git diff NombreEquipo1 NombreEquipo1vsNombreEquipo2
$ git checkout NombreEquipo1vsNombreEquipo2
$ git merge NombreEquipo1
$ git push https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
```

12. *Alumno2 vs alumno4 (conflictos)*

```
$ git pull https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo1
```

//Para el alumno2

```
$ git pull https://github.com/lorgut/EvolSoft.git NombreEquipo2
```
//Para el alumno4
```
$ git diff NombreEquipo1vsNombreEquipo2 NombreEquipo1
```
//Para el alumno2
```
$ git diff NombreEquipo1vsNombreEquipo2 NombreEquipo2
```
//Para el alumno4
```
$ git merge NombreEquipo1vsNombreEquipo2
```
//Situados en la rama NombreEquipo2 para el alumno4
```
$ git merge NombreEquipo1vsNombreEquipo2
```
//Situados en la rama NombreEquipo1 para el alumno2
```
$ emacs pb.c
```
//Realiza los cambios
```
$ git add pb.c
$ git commit –m "Modificaciones en la entrada de la función lottery_numbers_simulation"
$ emacs pb.c
```
//Realiza los cambios
```
$ git add pb.c
$ git commit –m "Adaptando la función calculate_result para calcular probabilidades"
$ emacs pb.c
```
//Realiza los cambios
```
$ git add pb.c
$ git commit –m "Cálculos para las bolas blancas"
$ emacs pb.c
```
//Realiza los cambios
```
$ git add pb.c
$ git commit –m "Cálculos para la power ball"
```
// Se realizarán tantos commits como el alumno considere necesario.
```
Alumno que finaliza primero
$ git bundle create <nombrefichero.bundle> <NombreEquipo> <HEAD>
```
//Nombre que desee el alumno para el fichero

Se enviará un correo, o se le pasará en un pen-drive al alumno del otro equipo el fichero generado tras usar la orden anterior.

Alumno que recibe el correo
```
$ git bundle verify <nombrefichero.bundle>
$ git clone <nombrefichero.bundle> -b master <Nombrerepo>
$ cd Nombrerepo
$ git hist
$ git push https://github.com/lorgut/EvolSoft.git NombreEquipo1vsNombreEquipo2
```

13. Dividiendo commits

```
$ git add -i
```
Utilizaremos la opción 5 (patch) para dividir los cambios. Añadiremos los cambios con `y`, no añadiremos con `n` y dividiremos con `s`. Cada vez que terminemos de añadir los cambios que queremos que pertenezcan a un mismo commit, realizaremos el commit del mismo y seguiremos luego con el siguiente conjunto de cambios que queramos que pertenezcan al commit. Así hasta finalizar el ejercicio.

14. Rebase manejo de ramas

```
$ gitk NombreEquipo1vsNombreEquipo2 NombreEquipo1
$ gitk NombreEquipo1vsNombreEquipo2 NombreEquipo2
$ gitweb NombreEquipo1vsNombreEquipo2 NombreEquipo2
$ gitweb NombreEquipo1vsNombreEquipo2 NombreEquipo1
$ git rebase NombreEquipo1 NombreEquipo1vsNombreEquipo2
$ git rebase NombreEquipo2 NombreEquipo1vsNombreEquipo2
```

15. ¿¡Quién hizo eso!?

```
$ git pull https://github.com/lorgut/EvolSoft.git
NombreEquipo1vsNombreEquipo2
$ git blame pb.c
```

En la salida se puede observar quién hizo los cambios, o bien en github, seleccionando el fichero correspondiente, en el botón “blame” también podemos observarlo.