Raúl Gil López
53842198A

Practica: git rebase, cherry-pick y remotes

Departamento de Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación, URJC

16 de noviembre de 2020

En estos ejercicios utilizaremos una variante del comando git log que permite visualizar el arbol de commits, los nombres de las ramas y el puntero HEAD:

git log --all --decorate --oneline --graph

Para mayor comodidad, crea un alias en la shell que le de el nombre gitgraph:

alias gitgraph="git log --all --decorate --oneline --graph"

1. Recursos gráficos para aprender git

Estos recursos pueden resultarte utiles para visualizar el grafo de commits: https://git-school.github.io/visualizing-git https://learngitbranching.js.org/

Recuerda que git log --all --decorate --oneline --graph muestra esta misma informacion para el grafo de tus repos.

2. Ejercicio 1

Responde a las siguientes preguntas:

1. **Crea un nuevo repositorio git de nombre alpha**

```
(base) root@kali:~/ISI/Practica 2# mkdir alpha
(base) root@kali:~/ISI/Practica 2# cd alpha/
(base) root@kali:~/ISI/Practica 2/alpha# git init
```

2. **Crea un directorio llamado data dentro del repositorio**

```
(base) root@kali:~/ISI/Practica 2/alpha# mkdir data
(base) root@kali:~/ISI/Practica 2/alpha# cd data/
```

3. **Añade un nuevo fichero data/letra.txt que contenga una línea con el siguiente texto: a**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# touch letra.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# echo a > letra.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# cat letra.txt
a
```

4. **Añade un nuevo fichero data/numero.txt que contenga una línea con el siguiente texto: 1**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# touch numero.txt; echo 1>numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# cat numero.txt

(base) root@kali:~/ISI/Practica 2/alpha/data# echo 1 > numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# cat numero.txt
1
```

5. **Crea un commit que añada ambos ficheros, utilizando el siguiente comentario para el commit: a 1**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git add letra.txt numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "a 1"
[master (commit-raíz) a353ecc] a 1
 2 files changed, 2 insertions(+)
 create mode 100644 data/letra.txt
 create mode 100644 data/numero.txt
```

6. **Substituye el contenido de data/numero.txt por este otro: 2**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# echo 2 > numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# cat numero.txt
2
```

7. **Crea un nuevo commit que refleje este cambio, utilizando el siguiente comentario para el commit: a 2**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "a 2"
```

8. **Haz ahora que HEAD entre en modo desacoplado (detached), haciendo que apunte al commit a 2, en lugar de apuntar a la rama master. Comprueba que lo has hecho bien con gitgraph o con git log.**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git checkout 53c1035
Nota: cambiando a '53c1035'.

Te encuentras en estado 'detached HEAD'. Puedes revisar por aquí, hacer
cambios experimentales y hacer commits, y puedes descartar cualquier
commit que hayas hecho en este estado sin impactar a tu rama realizando
otro checkout.
```

9. **Substituye el contenido de data/numero.txt por este otro: 3**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# echo 3 > numero.txt
```

10. **Crea un nuevo commit que re eje este cambio, utilizando el siguiente comentario para el commit: a 3**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "a 3"
```

11. **Crea una nueva rama de nombre new-feature**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git branch new-feature
```

12. **Comprueba que HEAD sigue sin apuntar a ninguna rama. Pero este ultimo commit esta a salvo porque la rama new-feature ya apunta a ese commit por lo que ahora es seguro mover HEAD a cualquier otro sitio.**

Podemos ver que HEAD apunta al commit al igual que la nueva rama new-feature

```
(base) root@kali:~/ISI/Practica 2/alpha/data# gitgraph
* 0be7668 (HEAD, new-feature) a 3
* 53c1035 (master) a 2
* a353ecc a 1
```

13. **Cambiate a la rama master**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git checkout master
```

14. **Comprueba las diferencias entre la rama master y la rama new-feature con git diff master..new-feature**

En la rama master está comprometido "a 2" y en la nueva rama "a 3"

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git diff master..new-feature
diff --git a/data/numero.txt b/data/numero.txt
index 0cfbf08..00750ed 100644
--- a/data/numero.txt
+++ b/data/numero.txt
@@ -1 +1 @@
-2
+3
```

15. **Estando en la rama master, substituye el contenido de data/numero.txt por este otro: 789**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# echo 789 > numero.txt
```

16. **Cambiate a la rama new-feature. Como podrás comprobar, no es posible. Describe al menos 3 soluciones para conseguir cambiarte a la rama new-feature. Pruébalos volviendo a repetir los 3 pasos anteriores**

Para solventar este conflicto podemos usar "git stash", modificar el fichero manualmente o comprometerlo haciendo commit.

17. **Observa el estado del grafo de commits (o history)**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# gitgraph
*   af6485b (refs/stash) WIP on master: 53c1035 a 2
|\
| * 2fa27a6 index on master: 53c1035 a 2
|/
| * 0be7668 (HEAD -> new-feature) a 3
|/
* 53c1035 (master) a 2
* a353ecc a 1
```

18. **Estando en la rama new-feature mezcla en ella la rama master. Describe que ha ocurrido y por que: si se ha generado un nuevo commit explica por que, si no también, explica los cambios de las etiquetas de las ramas, explica los cambios en los ficheros del directorio data, etc.**

Observamos que no se crea ningún commit ya que la rama new-features tiene todos los commit que tiene master

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git checkout new-feature
Cambiado a rama 'new-feature'
(base) root@kali:~/ISI/Practica 2/alpha/data# git merge master
Ya está actualizado.
```

19. **Cambiate a la rama master y mezcla en ella la rama new-feature. Describe que ha ocurrido y por que: si se ha generado un nuevo commit explica por que, si no también, explica los cambios de las etiquetas de las ramas, explica los cambios en los ficheros del directorio data, etc.**

Se ha hecho un fast-forward apuntando master al commit con “a 3”

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git checkout master
Cambiado a rama 'master'
(base) root@kali:~/ISI/Practica 2/alpha/data# git merge new-feature
Actualizando 53c1035..0be7668
Fast-forward
 data/numero.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) root@kali:~/ISI/Practica 2/alpha/data# gitgraph
*   af6485b (refs/stash) WIP on master: 53c1035 a 2
|\
| * 2fa27a6 index on master: 53c1035 a 2
|/
| * 0be7668 (HEAD -> master, new-feature) a 3
|/
* 53c1035 a 2
* a353ecc a 1
```

20. **Estando en la rama master substituye el contenido de data/numero.txt por este otro: 4**
21. **Crea un nuevo commit que re eje este cambio, utilizando el siguiente comentario para el commit: a 4**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# echo 4 > numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "a 4"
[master 373b2a4] a 4
 1 file changed, 1 insertion(+), 1 deletion(-)
```

22. **Cambiate a la rama new-feature**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git checkout new-feature
Cambiado a rama 'new-feature'
```

23. **Estando en la rama new-feature substituye el contenido de data/letra.txt por este otro: b. Comprueba el contenido de los dos ficheros de data**
24. **Crea un nuevo commit que re eje este cambio, utilizando el siguiente comentario para el commit: b 3**

```
b
(base) root@kali:~/ISI/Practica 2/alpha/data# cat numero.txt
3
(base) root@kali:~/ISI/Practica 2/alpha/data# git add letra.txt numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "b 3"
[new-feature 20d4c57] b 3
 1 file changed, 1 insertion(+), 1 deletion(-)
```

25. **Estando en la rama new-feature mezcla en ella la rama master. Si se crea un nuevo commit, ponle el siguiente comentario: b 4. Describe que ha ocurrido y por que: si se ha generado un nuevo commit explica por que, si no también, explica los cambios de las etiquetas de las ramas, explica los cambios en los ficheros del directorio data, etc.**

Se crea un nuevo commit ya que en la rama new-feature no tiene el commit “a 4”

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git merge master
ayuda: Esperando que tu editor cierre el archivo ... error: cannot run ‘vi’: No exist
e el fichero o el directorio
error: unable to start editor ''vi''
No se realiza commit de la fusión; use 'git commit' para completar la fusión.
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "b 4"
[new-feature 5209539] b 4
(base) root@kali:~/ISI/Practica 2/alpha/data# gitgraph
*   5209539 (HEAD -> new-feature) b 4
|\
| * 373b2a4 (master) a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
```

26. **Cambiate a la rama master**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git checkout master
Cambiado a rama 'master'
```

27. **Mezcla los contenidos de la rama new-feature. Describe lo ocurrido al mezclar esta rama. Describe que ha ocurrido y por que: si se ha generado un nuevo commit explica por que, si no también, explica los cambios de las etiquetas de las ramas, explica los cambios en los ficheros del directorio data, etc.**

No se realiza ningún commit. Se hace un fast-forward ya que no aporta nada a new-features

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git merge new-feature
Actualizando 373b2a4..5209539
Fast-forward
 data/letra.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) root@kali:~/ISI/Practica 2/alpha/data# gitgraph
*   5209539 (HEAD -> master, new-feature) b 4
|\
| * 373b2a4 a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
```

28. **Cambiate a la rama new-feature**
29. **Substituye el contenido de data/numero.txt por este otro: 5. Comprueba el contenido de los ficheros del directorio data**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git checkout new-feature
Cambiado a rama 'new-feature'
(base) root@kali:~/ISI/Practica 2/alpha/data# echo 5 > numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# cat letra.txt
b
(base) root@kali:~/ISI/Practica 2/alpha/data# cat numero.txt
5
```

30. **Crea un nuevo commit que re eje este cambio, utilizando el siguiente comentario para el commit: b 5**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "b 5"
[new-feature c9e359d] b 5
 1 file changed, 1 insertion(+), 1 deletion(-)
```

31. **Cambiate a la rama master**
32. **Substituye el contenido de data/numero.txt por este otro: 6. Comprueba el contenido de los ficheros del directorio data**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git checkout master
Cambiado a rama 'master'
(base) root@kali:~/ISI/Practica 2/alpha/data# echo 6 > numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# cat letra.txt
b
(base) root@kali:~/ISI/Practica 2/alpha/data# cat numero.txt
6
```

33. **Crea un nuevo commit que re eje este cambio, utilizando el siguiente comentario para el commit: b 6**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "b 6"
[master 3a6915f] b 6
 1 file changed, 1 insertion(+), 1 deletion(-)
```

34. **Compara las diferencias entre las ramas master y new-feature con git diff master..new-feature**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git diff master..new-feature
diff --git a/data/numero.txt b/data/numero.txt
index 1e8b314..7ed6ff8 100644
--- a/data/numero.txt
+++ b/data/numero.txt
@@ -1 +1 @@
-6
+5
```

35. **Estando en la rama master, mezcla los contenidos de la rama new-feature. Describe lo ocurrido al mezclar esta rama. Describe que ha ocurrido y por que: si se ha generado un nuevo commit explica por que, si no también, explica los cambios de las etiquetas de las ramas, explica los cambios en los ficheros del directorio data, etc. En caso de conflicto en el fichero numero.txt resuélvelo sumando los valores de los ficheros y creando un commit con el comentario adecuado siguiendo el criterio de los comentarios de esta practica**

Tenemos un conflicto ya que tenemos diferentes el mismo fichero(no me he dado cuenta y no he hecho commit (b 11) y no me sale en mi gitgraph, pero se que va aquí)

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git merge new-feature
Auto-fusionando data/numero.txt
CONFLICTO (contenido): Conflicto de fusión en data/numero.txt
Fusión automática falló; arregle los conflictos y luego realice un commit con el resu
ltado.
(base) root@kali:~/ISI/Practica 2/alpha/data# cat numero.txt
<<<<<<< HEAD
6
=======
5
>>>>>>> new-feature
(base) root@kali:~/ISI/Practica 2/alpha/data# echo 11 > numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "b 11"
[master 4267a02] b 11
```

36. **Estando en la rama master, borra el fichero data/letra.txt y crea un commit que re eje este cambio, poniéndole el siguiente comentario al commit: 11**

```
(base) root@kali:~/ISI/Practica 2/alpha/data# git rm letra.txt
rm 'data/letra.txt'
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "11"
[master 8b2a2cc] 11
 1 file changed, 1 deletion(-)
 delete mode 100644 data/letra.txt
```

Al terminar el ultimo apartado deber as tener un grafo idéntico al de la siguiente gura, aunque con hashes de los commits distintos claro:

```
(base) root@kali:~/ISI/Practica 2/alpha/data# gitgraph
*   4267a02 (HEAD -> master) b 11
|\
| * c9e359d (new-feature) b 5
* | 3a6915f b 6
|/
*   5209539 b 4
|\
| * 373b2a4 a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
```

```
* 53fb3fc (HEAD -> master) 11
*   5c21020 b 11
|\
| * d936b7d (new-feature) b 5
* | 7ca7bd3 b 6
|/
*   b3a98ca b 4
|\
| * dfa91af a 4
* | d9cdbc0 b 3
|/
* 2358478 a 3
* d9dadf9 a 2
* faae5e0 a 1
```

3. Ejercicio 2

En este ejercicio se repiten las acciones del anterior ejercicio, pero en lugar de utilizar git merge para mezclar ramas, se utiliza git rebase.

Cuando se mezclan dos ramas, git merge genera en ocasiones commits que reflejan la mezcla, lo que hace que no aparezca una historia lineal de commits.

Utilizando git rebase en lugar de git merge se copian los commits de una rama origen encima de los commits de una rama destino. De esta forma en la rama destino se acaban teniendo todos los cambios de la rama origen, pero sin que aparezcan commits por mezcla, quedando en la rama destino una historia lineal de commits.

Ejemplo de uso: Si estando en la rama new-feature se ejecuta git rebase master, los commits de new-feature se copian encima de los de master.

1. **Crea un nuevo repositorio git de nombre beta**
2. **Crea un directorio llamado data dentro del repositorio**

```
(base) root@kali:~/ISI/Practica 2# mkdir beta
(base) root@kali:~/ISI/Practica 2# cd beta/
(base) root@kali:~/ISI/Practica 2/beta# git init
Inicializado repositorio Git vacío en /root/ISI/Practica 2/beta/.git/
(base) root@kali:~/ISI/Practica 2/beta# mkdir data
(base) root@kali:~/ISI/Practica 2/beta# cd data/
```

3. **Añade un nuevo fichero data/letra.txt que contenga una l nea con el siguiente texto: a**
4. **Añade un nuevo fichero data/numero.txt que contenga una l nea con el siguiente texto: 1**

```
(base) root@kali:~/ISI/Practica 2/beta/data# touch letra.txt
(base) root@kali:~/ISI/Practica 2/beta/data# touch numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# echo a > letra.txt
(base) root@kali:~/ISI/Practica 2/beta/data# echo 1 > numero.txt
```

5. **Crea un commit que añada ambos ficheros, utilizando el siguiente comentario para el commit: a 1**

```
(base) root@kali:~/ISI/Practica 2/beta/data# git add letra.txt numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git commit -m "a 1"
[master (commit-raíz) 9e9fbed] a 1
 2 files changed, 2 insertions(+)
 create mode 100644 data/letra.txt
 create mode 100644 data/numero.txt
```

6. **Substituye el contenido de data/numero.txt por este otro: 2**
7. **Crea un nuevo commit que re eje este cambio, utilizando el siguiente comentario para el commit: a 2**

```
(base) root@kali:~/ISI/Practica 2/beta/data# echo 2 > numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git commit -m "a 2"
[master d4c952e] a 2
 1 file changed, 1 insertion(+), 1 deletion(-)
```

8. **Haz ahora que HEAD entre en modo desacoplado (detached), haciendo que apunte al commit a 2, en lugar de apuntar a la rama master. Comprueba que lo has hecho bien con gitgraph o con git log.**

```
(base) root@kali:~/ISI/Practica 2/beta/data# gitgraph
* d4c952e (HEAD -> master) a 2
* 9e9fbed a 1
(base) root@kali:~/ISI/Practica 2/beta/data# ^C
(base) root@kali:~/ISI/Practica 2/beta/data# git checkout d4c952e
Nota: cambiando a 'd4c952e'.

Te encuentras en estado 'detached HEAD'. Puedes revisar por aquí, hacer
cambios experimentales y hacer commits, y puedes descartar cualquier
commit que hayas hecho en este estado sin impactar a tu rama realizando
otro checkout.
```

9. **Substituye el contenido de data/numero.txt por este otro: 3**
10. **Crea un nuevo commit que re eje este cambio, utilizando el siguiente comentario para el commit: a 3**

```
(base) root@kali:~/ISI/Practica 2/beta/data# echo 3 > numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git commit -m "a 3"
[HEAD desacoplado 43b2fe1] a 3
 1 file changed, 1 insertion(+), 1 deletion(-)
```

11. **Crea una nueva rama de nombre new-feature**
12. **Comprueba que HEAD sigue sin apuntar a ninguna rama. Pero este ultimo commit esta a salvo porque la rama new-feature ya apunta a ese commit por lo que ahora es seguro mover HEAD a cualquier otro sitio.**

```
(base) root@kali:~/ISI/Practica 2/beta/data# git branch new-feature
(base) root@kali:~/ISI/Practica 2/beta/data# gitgraph
* 43b2fe1 (HEAD, new-feature) a 3
* d4c952e (master) a 2
* 9e9fbed a 1
```

13. **Cambiate a la rama master**
14. **Comprueba las diferencias entre la rama master y la rama new-feature con git diff master..new-feature**

```
(base) root@kali:~/ISI/Practica 2/beta/data# git checkout master
La posición previa de HEAD era 43b2fe1 a 3
Cambiado a rama 'master'
(base) root@kali:~/ISI/Practica 2/beta/data# git diff master..new-feature
diff --git a/data/numero.txt b/data/numero.txt
index 0cfbf08..00750ed 100644
--- a/data/numero.txt
+++ b/data/numero.txt
@@ -1 +1 @@
-2
+3
```

New-feature tiene a 3 y master a 2

15. **Estando en la rama master, substituye el contenido de data/numero.txt por este otro: 789**
16. **Cambiate a la rama new-feature. Como podras comprobar, no es posible. Describe al menos 3 soluciones para conseguir cambiarte a la rama new-feature. Pruebalos volviendo a repetir los 3 pasos anteriores**

```
(base) root@kali:~/ISI/Practica 2/beta/data# echo 789 > numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git checkout new-feature
error: Los cambios locales de los siguientes archivos serán sobrescritos por checkout
:
        data/numero.txt
Por favor realiza un commit con los cambios o un stash antes de cambiar ramas.
Abortando
```

Para solventar este conflicto podemos usar "git stash", modificar el fichero manualmente o comprometerlo haciendo commit.

17. **Observa el estado del grafo de commits (o history)**

```
(base) root@kali:~/ISI/Practica 2/beta/data# gitgraph
*   437db4a (refs/stash) WIP on master: d4c952e a 2
|\
| * 7f5fef8 index on master: d4c952e a 2
|/
| * 43b2fe1 (new-feature) a 3
|/
* d4c952e (HEAD -> master) a 2
* 9e9fbed a 1
```

18. **En el ejercicio anterior, llegados a este punto, utilizabamos git merge para mezclar en la rama new-feature los cambios de la rama master. Ahora lo vamos a hacer copiando los commits de la rama master en la rama new-feature usando git rebase. Para ello, estando en la rama master, ejecuta: git rebase new-feature.**
Comprueba los resultados con gitgraph. Son identicos a los obtenidos en el ejercicio anterior. Como puedes comprobar las dos ramas apuntan al ultimo commit.

```
(base) root@kali:~/ISI/Practica 2/beta/data# git rebase new-feature
Rebase aplicado satisfactoriamente y actualizado refs/heads/master.
(base) root@kali:~/ISI/Practica 2/beta/data# gitgraph
*   437db4a (refs/stash) WIP on master: d4c952e a 2
|\
| * 7f5fef8 index on master: d4c952e a 2
|/
| * 43b2fe1 (HEAD -> master, new-feature) a 3
|/
* d4c952e a 2
* 9e9fbed a 1
```

19. **Estando en la rama master substituye el contenido de data/numero.txt por este otro: 4**

```
(base) root@kali:~/ISI/Practica 2/beta/data# echo 4 > numero.txt
```

20. **Crea un nuevo commit que refleje este cambio, utilizando el siguiente comentario para el commit: a 4**

```
(base) root@kali:~/ISI/Practica 2/beta/data# git commit -m "a 4"
[master bfcafb4] a 4
 1 file changed, 1 insertion(+), 1 deletion(-)
```

21. Cambiate a la rama new-feature

```
(base) root@kali:~/ISI/Practica 2/beta/data# git checkout new-feature
Cambiado a rama 'new-feature'
```

22. Estando en la rama new-feature substituye el contenido de data/letra.txt por este otro: b. Comprueba el contenido de los dos ficheros de data

```
(base) root@kali:~/ISI/Practica 2/beta/data# echo b > letra.txt
```

23. Crea un nuevo commit que re eje este cambio, utilizando el siguiente comentario para el commit: b 3

```
(base) root@kali:~/ISI/Practica 2/beta/data# git add letra.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git commit -m "b 3"
[new-feature 27001cc] b 3
 1 file changed, 1 insertion(+), 1 deletion(-)
```

24. En el ejercicio anterior, llegados a este punto, utilizabamos git merge para mezclar en la rama new-feature los cambios de la rama master. Ahora lo vamos a hacer copiando los commits de la rama master en la rama new-feature usando git rebase. Para ello, estando en la rama master, ejecuta: git rebase new-feature.Comprueba lo que ha ocurrido con gitgraph. Los cambios de la rama master se han aplicado encima de los de la rama new-feature. Mira el contenido de los dos cheros, data/numero.txt y data/letra.txt. El comentario del ultimo commit no es quiza el mas adecuado ya que no re eja el contenido de los cheros. Actual zalo con git commit --amend -m"b 4".

```
(base) root@kali:~/ISI/Practica 2/beta/data# git checkout master
Cambiado a rama 'master'
(base) root@kali:~/ISI/Practica 2/beta/data# git rebase new-feature
Rebase aplicado satisfactoriamente y actualizado refs/heads/master.
(base) root@kali:~/ISI/Practica 2/beta/data# gitgraph
* 8062207 (HEAD -> master) a 4
* 27001cc (new-feature) b 3
* 43b2fe1 a 3
| * 437db4a (refs/stash) WIP on master: d4c952e a 2
|/|
| * 7f5fef8 index on master: d4c952e a 2
|/
* d4c952e a 2
* 9e9fbed a 1
(base) root@kali:~/ISI/Practica 2/beta/data# cat letra.txt
b
(base) root@kali:~/ISI/Practica 2/beta/data# cat numero.txt
4
(base) root@kali:~/ISI/Practica 2/beta/data# git commit --amend -m"b 4"
[master 5f3ba58] b 4
 Date: Wed Nov 25 13:25:46 2020 +0100
 1 file changed, 1 insertion(+), 1 deletion(-)
```

25. Vamos a realizar el siguiente cambio siguiendo la pauta del anterior ejercicio. Vamos a usar de nuevo la rama new-feature. Podr amos borrarla y crearla de nuevo estando en master. Tambien podemos moverla con el siguiente comando: git branch -f new-feature master.

```
(base) root@kali:~/ISI/Practica 2/beta/data# git branch -f new-feature master
```

26. Cambiate a la rama new-feature

```
(base) root@kali:~/ISI/Practica 2/beta/data# git checkout new-feature
Cambiado a rama 'new-feature'
```

27. Substituye el contenido de data/numero.txt por este otro: 5. Comprueba el contenido de los cheros del directorio data

```
(base) root@kali:~/ISI/Practica 2/beta/data# echo 5 > numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# cat letra.txt
b
(base) root@kali:~/ISI/Practica 2/beta/data# cat numero.txt
5
```

28. Crea un nuevo commit que refleje este cambio, utilizando el siguiente comentario para el commit: b 5

```
(base) root@kali:~/ISI/Practica 2/beta/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git commit -m "b 5"
[new-feature 14925f2] b 5
 1 file changed, 1 insertion(+), 1 deletion(-)
```

29. Cambiate a la rama master

```
(base) root@kali:~/ISI/Practica 2/beta/data# git checkout master
Cambiado a rama 'master'
```

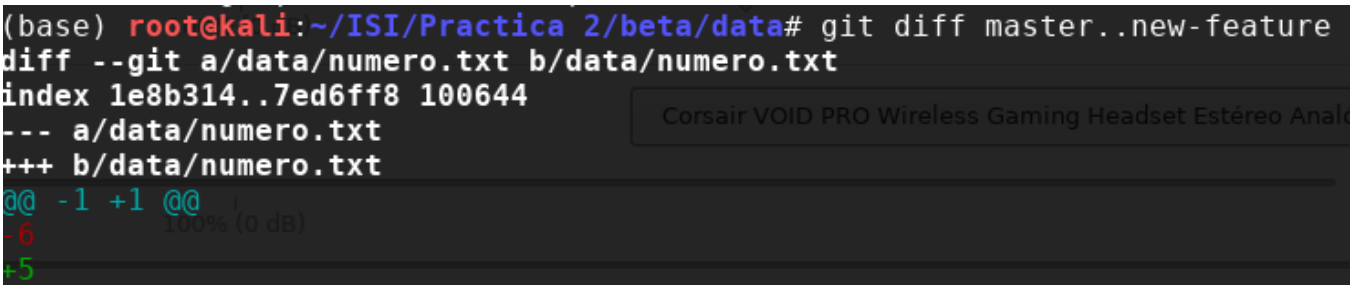
30. Substituye el contenido de data/numero.txt por este otro: 6. Comprueba el contenido de los ficheros del directorio data

```
(base) root@kali:~/ISI/Practica 2/beta/data# echo 6 > numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# cat letra.txt
b
(base) root@kali:~/ISI/Practica 2/beta/data# cat numero.txt
6
```

31. Crea un nuevo commit que refleje este cambio, utilizando el siguiente comentario para el commit: b 6

```
(base) root@kali:~/ISI/Practica 2/beta/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git commit -m "b 6"
[master a62a9e9] b 6
 1 file changed, 1 insertion(+), 1 deletion(-)
```

32. **Compara las diferencias entre las ramas master y new-feature con git diff master..new-feature**



33. **En el ejercicio anterior, llegados a este punto, utilizabamos git merge para mezclar en la rama master los cambios de la rama new-feature. Ahora lo vamos a hacer copiando los commits de la rama new-feature en la rama master usando git rebase. Para ello, estando en la rama new-feature, ejecuta: git rebase master.**

    Al igual que ocurrió en el ejercicio anterior cuando utilizamos git merge, git rebase ahora también ha detectado un conflicto. Para resolverlo, editamos el fichero data/number.txt y dejamos el contenido 11. Despues lo añadimos a staging area y ejecutamos git rebase --continue.

    Comprueba lo que ha ocurrido con gitgraph. Los cambios de la rama new-feature se han aplicado encima de los de la rama master. Mira el contenido de los dos cheros, data/numero.txt y data/letra.txt. El comentario del ultimo commit no es quizá el mas adecuado ya que no refleja el contenido de los cheros. Actualízalo con git commit --amend -m"b 11".

```
(base) root@kali:~/ISI/Practica 2/beta/data# git checkout new-feature
Cambiado a rama 'new-feature'
(base) root@kali:~/ISI/Practica 2/beta/data# git rebase master
Auto-fusionando data/numero.txt
CONFLICTO (contenido): Conflicto de fusión en data/numero.txt
error: no se pudo aplicar 14925f2... b 5
Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort".
No se pudo aplicar 14925f2... b 5
(base) root@kali:~/ISI/Practica 2/beta/data# cat numero.txt
<<<<<<< HEAD
6
=======
5
>>>>>>> 14925f2... b 5
(base) root@kali:~/ISI/Practica 2/beta/data# echo 11 > numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/beta/data# git rebase --continue
(base) root@kali:~/ISI/Practica 2/beta/data# cat numero.txt
11
(base) root@kali:~/ISI/Practica 2/beta/data# git commit -m "b 11"
[HEAD desacoplado e7fd63a] b 11
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) root@kali:~/ISI/Practica 2/beta/data# gitgraph
* e7fd63a (HEAD) b 11
* a62a9e9 (master) b 6
| * 14925f2 (new-feature) b 5
|/
* 5f3ba58 b 4
* 27001cc b 3
* 43b2fe1 a 3
| * 437db4a (refs/stash) WIP on master: d4c952e a 2
|/|
| * 7f5fef8 index on master: d4c952e a 2
|/
* d4c952e a 2
* 9e9fbed a 1
```

34. **Fíjate donde esta la etiqueta de la rama master. Vamos a moverla al mismo commit en el que esta la rama new-feature. Podríamos moverla con git branch -f master new-feature. Vamos a hacerlo sin embargo con git rebase. Trata de pensar el comando git rebase que deber amos aplicar.**

35. **Efectivamente, basta con hacer un git rebase new-feature estando en la rama master para que se produzca un fast forward, es decir, que se copie la etiqueta, ya que todos los commits de master ya estaban en new-feature.**

36. **Finalmente, estando en la rama master, borra el fichero data/letra.txt y crea un commit que re eje este cambio, poniéndole el siguiente comentario al commit: 11**

Al terminar el ultimo apartado deber as tener un grafo de commits idéntico al de la siguiente gura, aunque con hashes de los commits distintos claro:

```
* d303cb3 (HEAD -> master) 11
* beb2e00 (new-feature) b 11
* 13857a6 b 6
* 1fa04e7 b 4
* 7294c99 b 3
* 92af053 a 3
* 6528d66 a 2
* 23ba58a a 1
```

4. Ejercicio 3

En este ejercicio vas a hacer uso de git cherry-pick para incorporar selectivamente otros commits, sin realizar merge o rebase.

1. **Copia el repositorio resultado del ejercicio 1, dandole el nombre gamma: cp -r alpha gamma. El resto de este ejercicio real zalo en el repositorio gamma**

```
(base) root@kali:~/ISI/Practica 2# cp -r alpha gamma
(base) root@kali:~/ISI/Practica 2# cd gamma
(base) root@kali:~/ISI/Practica 2/gamma# git init
Reinicializado el repositorio Git existente en /root/ISI/Practica 2/gamma/.git/
```

2. **Haz que la rama new-feature esté en el mismo commit que master**

   git checkout new-feature; git rebase master

3. **Estando en la rama new-feature crea un chero data/Complex.java con los siguientes contenidos:**

```
public class Complex { private final
    double re; private final double
    im;

    public Complex(double real, double imag) {
        re = real;
        im = imag;
    }
}
```

4. **Crea un nuevo commit que re eje este cambio, poniendole el siguiente comentario: 11 Complex**

```
(base) root@kali:~/ISI/Practica 2/gamma# cd data/
(base) root@kali:~/ISI/Practica 2/gamma/data# touch Complex.java
(base) root@kali:~/ISI/Practica 2/gamma/data# git add Complex.java
(base) root@kali:~/ISI/Practica 2/gamma/data# git commit -m "11 Complex"
[new-feature b34393b] 11 Complex
 1 file changed, 8 insertions(+)
 create mode 100644 data/Complex.java
```

5. **Vuelve a la rama master y crea una nueva rama new-feature-2**

```
(base) root@kali:~/ISI/Practica 2/gamma/data# git checkout master
Cambiado a rama 'master'
(base) root@kali:~/ISI/Practica 2/gamma/data# git branch new-feature-2
```

6. **Ve a la rama new-feature-2 y crea en ella un chero data/Point.java con los siguientes contenidos:**

```
public class Point { private
    double x; private double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}
```

```
(base) root@kali:~/ISI/Practica 2/gamma/data# git checkout new-feature-2
Cambiado a rama 'new-feature-2'
(base) root@kali:~/ISI/Practica 2/gamma/data# touch Point.java
(base) root@kali:~/ISI/Practica 2/gamma/data# git checkout new-feature
Cambiado a rama 'new-feature'
(base) root@kali:~/ISI/Practica 2/gamma/data# cat Complex.java
public class Complex {
  private final double re;
  private final double im;
  public Complex(double real, double imag) {
    re = real;
    im = imag;
  }
}
```

7. **Crea un nuevo commit que re eje este cambio, poniendole el siguiente comentario: 11 Point**

```
(base) root@kali:~/ISI/Practica 2/gamma/data# git add Point.java
(base) root@kali:~/ISI/Practica 2/gamma/data# git commit -m "11 Point"
[new-feature-2 db26ab4] 11 Point
 1 file changed, 8 insertions(+)
 create mode 100644 data/Point.java
```

8. **Ve a la rama master y crea en ella un chero data/Main.java con los siguientes contenidos:**

```
public class Main {
    public static void main(String[] args) { Point p1, p2;
    }
}
```

```
(base) root@kali:~/ISI/Practica 2/gamma/data# git checkout master
Cambiado a rama 'master'
(base) root@kali:~/ISI/Practica 2/gamma/data# touch Main.java
(base) root@kali:~/ISI/Practica 2/gamma/data# cat Main.java
public class Main {
  public static void main(String[] args) {
    Point p1, p2;
  }
}
```

9. **Crea un nuevo commit que re eje este cambio, poniendole el siguiente comentario: 11 Main**

```
(base) root@kali:~/ISI/Practica 2/gamma/data# git add Main.java
(base) root@kali:~/ISI/Practica 2/gamma/data# git commit -m"11 Main"
[master 71fee1b] 11 Main
 1 file changed, 5 insertions(+)
 create mode 100644 data/Main.java
```

10. **Utiliza cherry-pick para que en la rama master aparezca el chero Point.java que se ha creado en el ultimo commit de la rama new-feature-2. El formato del comando es git cherry-pick commit-id**

```
(base) root@kali:~/ISI/Practica 2/gamma/data# gitgra
* 71fee1b (HEAD -> master) 11 Main
| * db26ab4 (new-feature-2) 11 Point
|/
| * b34393b (new-feature) 11 Complex
|/
* 8b2a2cc 11
*   4267a02 b 11
|\
| * c9e359d b 5
* | 3a6915f b 6
|/
*   5209539 b 4
|\
| * 373b2a4 a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
(base) root@kali:~/ISI/Practica 2/gamma/data# ^C
(base) root@kali:~/ISI/Practica 2/gamma/data# git che
checkout      cherry        cherry-pick
(base) root@kali:~/ISI/Practica 2/gamma/data# git cherry-pick db26ab4
[master 6ae4745] 11 Point
 Date: Wed Nov 25 15:02:25 2020 +0100
 1 file changed, 8 insertions(+)
 create mode 100644 data/Point.java
```

11. Actualiza el comentario del ultimo commit para sea este: 11 Main-Point

```
(base) root@kali:~/ISI/Practica 2/gamma/data# git commit --amend -m "11 Main-Point"
[master e72fb0a] 11 Main-Point
 Date: Wed Nov 25 15:02:25 2020 +0100
 1 file changed, 8 insertions(+)
 create mode 100644 data/Point.java
```

12. **Tras haber elegido lo que quer amos de una de las ramas, queremos descartar lo hecho en new-feature y new-feature-2. Trata de eliminar ambas ramas. ¿Que ocurre? Fuerza el borrado de ambas ramas.**

No nos deja borrarla ya que no las hemos fusionado completamente. Forzamos el borrado.

```
(base) root@kali:~/ISI/Practica 2/gamma/data# git branch -d new-feature
error: La rama 'new-feature' no ha sido fusionada completamente.
Si estás seguro de querer borrarla, ejecuta 'git branch -D new-feature'.
(base) root@kali:~/ISI/Practica 2/gamma/data# git branch -D new-feature
Eliminada la rama new-feature (era b34393b)..
(base) root@kali:~/ISI/Practica 2/gamma/data# git branch -D new-feature-2
Eliminada la rama new-feature-2 (era db26ab4)..
(base) root@kali:~/ISI/Practica 2/gamma/data#
(base) root@kali:~/ISI/Practica 2/gamma/data# gitgraph
* e72fb0a (HEAD -> master) 11 Main-Point
* 71fee1b 11 Main
* 8b2a2cc 11
*   4267a02 b 11
|\
| * c9e359d b 5
* | 3a6915f b 6
|/
*   5209539 b 4
|\
| * 373b2a4 a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
```

Al terminar el ultimo apartado deber as tener un grafo de commits identico al de la siguiente gura, aunque con hashes de los commits distintos claro:

```
* 38addc7 (HEAD -> master) 11 Main-Point
* 230403b 11 Main
* 53fb3fc 11
*   5c21020 b 11
|\
| * d936b7d b 5
* | 7ca7bd3 b 6
|/
*   b3a98ca b 4
|\
| * dfa91af a 4
* | d9cdbc0 b 3
|/
* 2358478 a 3
* d9dadf9 a 2
* faae5e0 a 1
```

5. Ejercicio 4

En este ejercicio vas a crear y sincronizar varios repositorios con git pull y git remote.

1. **Haz una copia del repo que generaste al nal del Ejercicio 1: cp -r alpha remoteRepo. El repositorio remoteRepo puede funcionar como un repositorio remoto desde el punto de vista del repositorio alpha, y viceversa. A la hora de referenciar desde el respositorio alpha el respositorio remoto remoteRepo puedes utilizar su path en vez de utilizar una url como haces cuando utilizas un respositorio remoto de GitHub.**

```
(base) root@kali:~/ISI/Practica 2# cp -r alpha remoteRepo
(base) root@kali:~/ISI/Practica 2# cd remoteRepo/
(base) root@kali:~/ISI/Practica 2/remoteRepo# git init
Reinicializado el repositorio Git existente en /root/ISI/Practica 2/remoteRepo/
.git/
```

2. **Observa el grafo de commits en el repositorio remoteRepo**

```
(base) root@kali:~/ISI/Practica 2/remoteRepo# gitgraph
* 8b2a2cc (HEAD -> master) 11
*   4267a02 b 11
|\
| * c9e359d (new-feature) b 5
* | 3a6915f b 6
|/
* 5209539 b 4
|\
| * 373b2a4 a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
```

3. **Estando en el repo alpha añade el repositorio remoteRepo como repositorio remoto:**

   git remote add remoteRepo ../remoteRepo

   Comprueba con git remote -v que el anterior comando ha tenido efecto.

```
(base) root@kali:~/ISI/Practica 2/alpha# git remote add remoteRepo ../remoteRep
o/
(base) root@kali:~/ISI/Practica 2/alpha# git remote -v
remoteRepo      ../remoteRepo/ (fetch)
remoteRepo      ../remoteRepo/ (push)
```

4. **Situate en la rama master del repositorio remoteRepo, y modifica el fichero data/numero.txt para que contenga el valor 12**
5. **Comprometa los cambios poniendole el comenatario 12 al commit**

```
(base) root@kali:~/ISI/Practica 2/remoteRepo/data# git checkout master
Ya en 'master'
(base) root@kali:~/ISI/Practica 2/remoteRepo/data# echo 12 > numero.txt
(base) root@kali:~/ISI/Practica 2/remoteRepo/data# cat numero.txt
12
(base) root@kali:~/ISI/Practica 2/remoteRepo/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/remoteRepo/data# git commit -m "12"
[master 1c8bc85] 12
 1 file changed, 1 insertion(+), 1 deletion(-)
```

6. **Ve al repositorio alpha**

```
(base) root@kali:~/ISI/Practica 2/remoteRepo# cd ..
(base) root@kali:~/ISI/Practica 2# cd alpha/
(base) root@kali:~/ISI/Practica 2/alpha# gitgraph
*   8b2a2cc (HEAD -> master) 11
*   4267a02 b 11
|\
| * c9e359d (new-feature) b 5
* | 3a6915f b 6
|/
*   5209539 b 4
|\
| * 373b2a4 a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
```

7. **Obten con git fetch remoteRepo master el nuevo commit realizado en la rama master del repositorio remoteRepo. Tras ejecutar git fetch la etiqueta FETCH_HEAD apunta a la rama local en el repo alpha en la que se copian los commits remotos de la rama a la que se le ha hecho un fetch: remoteRepo/master**

```
(base) root@kali:~/ISI/Practica 2/alpha# git fetch remoteRepo master
remote: Enumerando objetos: 7, listo.
remote: Contando objetos: 100% (7/7), listo.
remote: Total 4 (delta 0), reusado 0 (delta 0), pack-reusado 0
Desempaquetando objetos: 100% (4/4), 258 bytes | 258.00 KiB/s, listo.
Desde ../remoteRepo
 * branch            master     -> FETCH_HEAD
 * [nueva rama]      master     -> remoteRepo/master
```

8.

8. **Observa el grafo de commits. F jate en la rama local en la que git fetch ha dejado los cambios remotos, remoteRepo/master.**

```
(base) root@kali:~/ISI/Practica 2/alpha# gitgraph
* 1c8bc85 (remoteRepo/master) 12
* 8b2a2cc (HEAD -> master) 11
*   4267a02 b 11
|\
| * c9e359d (new-feature) b 5
* | 3a6915f b 6
|/
*   5209539 b 4
|\
| * 373b2a4 a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
```

9. Mezcla en la rama master los commits obtenidos con git fetch. Puedes referirte a la rama local en la que estan los commits del repo remoto a traves de su nombre remoteRepo/master o a traves de la etiqueta FETCH_HEAD. Observa los cambios producidos en el grafo de commits tras realizarse la mezcla y explícalos. ¿Que tipo de merge se ha realizado?

se realiza un fastforward ya que los nuevos commits se han hecho en la rama local

```
(base) root@kali:~/ISI/Practica 2/alpha# git merge remoteRepo/master
Actualizando 8b2a2cc..1c8bc85
Fast-forward
 data/numero.txt | 2 +
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) root@kali:~/ISI/Practica 2/alpha# gitgraph
* 1c8bc85 (HEAD -> master, remoteRepo/master) 12
* 8b2a2cc 11
* 4267a02 b 11
|\
| * c9e359d (new-feature) b 5
* | 3a6915f b 6
|/
* 5209539 b 4
|\
| * 373b2a4 a 4
* | 20d4c57 b 3
|/
* 0be7668 a 3
| * af6485b (refs/stash) WIP on master: 53c1035 a 2
|/|
| * 2fa27a6 index on master: 53c1035 a 2
|/
* 53c1035 a 2
* a353ecc a 1
```

10. git pull es equivalente a git fetch + git merge[1].

Haz ahora un git pull remoteRepo master y explica los cambios observados.

observamos que ya no esta la rama local remoteRepo/master

11. Tambien desde remoteRepo podemos hacer git fetch + git merge, o directamente git pull, del repo alpha. Modi ca en el repo alpha el chero data/numero.txt para que contenga el numero 13. Haz un commit con comentario 13. A~nade en el repo remoteRepo un remote llamado alpha que correponda al repositorio en ../alpha. Ahora haz un git pull desde el repo remoteRepo para incorporar los cambios de la rama master de alpha a la rama master de remoteRepo. Comprueba que ambos repos tienen los mismos commit en su rama master. Observa en los repos alpha y en remoteRepo los nombres de las ramas que contienen los commits obtenidos del otro repo: respectivamente, remoteRepo/master y alpha/master.

Ejecuta git pull en ambos repos para que los grafos sean identicos, salvo por los nombres de las ramas que se utilizan para seguir los respectivos repos remotos.

```
(base) root@kali:~/ISI/Practica 2/alpha# git pull remoteRepo master
```

```
(base) root@kali:~/ISI/Practica 2# cd remoteRepo/
(base) root@kali:~/ISI/Practica 2/remoteRepo# git pull alpha master
```

12. Clonar un repositorio remoto con git clone repoRemoto repoLocal es equivalente a ejecutar la siguiente secuencia de comandos: mkdir repoLocal; cd repoLocal; git init; git add remote origin repoRemoto; git pull origin.

Estando en el directorio padre de alpha, clona el repositorio alpha en uno nuevo denominado remoteRepoNObare:
git clone alpha remoteRepoNObare. Recuerda que el nombre que se le da tras git clone al repositorio remoto es origin. Estando en el repo remoteRepoNOBare, ejecuta gitgraph y observa los nombres de las ramas locales que se han creado para seguir las ramas del repositorio remoto. Observa tambien don de se ha situado origin/HEAD, para seguir donde esta HEAD en el repo remoto. Comprueba con git remote -v la relación entre este repo y el repo alpha. Ahora desde remoteRepoNObare podríamos hacer pull de la rama master del repo alpha con git pull origin master. Hazlo. ¿Ha cambiado algo? ¿Por que?

```
(base) root@kali:~/ISI/Practica 2# git clone alpha remoteRepoNObare
Clonando en 'remoteRepoNObare'...
hecho.
```

13. **Ve al repositorio alpha y modifica el contenido del chero data/numero.txt para que contenga 14**
14. **Crea un commit que re eje dichos cambios, con comentario de commit 14**

```
(base) root@kali:~/ISI/Practica 2# cd alpha/
(base) root@kali:~/ISI/Practica 2/alpha# cd data/
(base) root@kali:~/ISI/Practica 2/alpha/data# echo 14 > numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git add numero.txt
(base) root@kali:~/ISI/Practica 2/alpha/data# git commit -m "14"
[master 4855ba8] 14
 1 file changed, 1 insertion(+), 1 deletion(-)
```

15. **Añade remoteRepoNObare como un nuevo remote de alpha. Comprueba los remotes con git remote -v. Ahora desde el repo alpha podemos también hacer pull del repo remoteRepoNObare.**

```
(base) root@kali:~/ISI/Practica 2/remoteRepoNObare# git remote add alpha ../alp
ha/
(base) root@kali:~/ISI/Practica 2/remoteRepoNObare# git remote -v
alpha   ../alpha/ (fetch)
alpha   ../alpha/ (push)
origin  /root/ISI/Practica 2/alpha (fetch)
origin  /root/ISI/Practica 2/alpha (push)
```

16. **Trata de hacer un push del nuevo commit de alpha a remoteRepoNObare: git push remoteRepoNObare master. Comprueba si esta el nuevo commit de alpha en remoteRepoNObare.**

17. **El commit no se ha propagado al hacer git push. Hacerlo modi car a en remoteRepoNObare la etiqueta HEAD, lo que podría crear confusión en el usuario del repo remoteRepoNObare, por lo que git no lo permite. El repositorio destino de un git push ha de haber sido creado como un repositorio bare. Los repos que creas en GitHub son repos bare**

18. **Clona el repositorio remoteRepoNOBare en uno nuevo denominado remoteRepoBare, pero esta vez añadiendo la opcion --bare: git clone remoteRepoNOBare remoteRepoBare --bare. Comprueba con gitgraph que en remoteRepoBare no esta el commit que tiene el repo alpha con comentario 14.**

```
(base) root@kali:~/ISI/Practica 2# git clone remoteRepoNObare/ remoteRepoBare -
-bare
Clonando en un repositorio vacío 'remoteRepoBare'...
hecho.
```

19. **Añade como repositorio remoto en alpha el nuevo repo remoteRepoBare.**

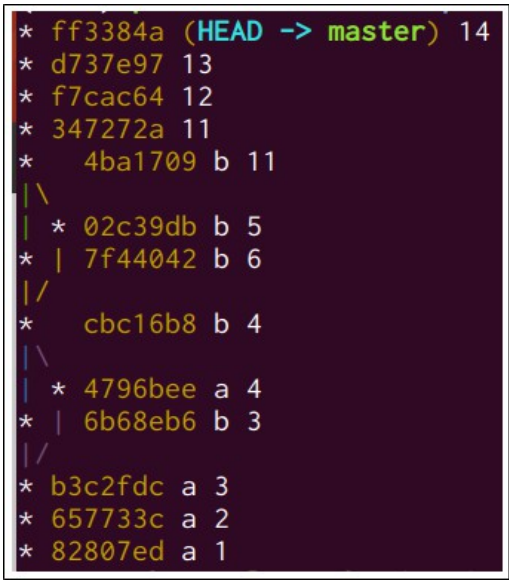
```
(base) root@kali:~/ISI/Practica 2# cd alpha/
(base) root@kali:~/ISI/Practica 2/alpha# git remote add remoteRepoBare ../remot
eRepoBare/
```

20. **Haz ahora un push de alpha a remoteRepoBare. Ahora si podrás hacerlo.**

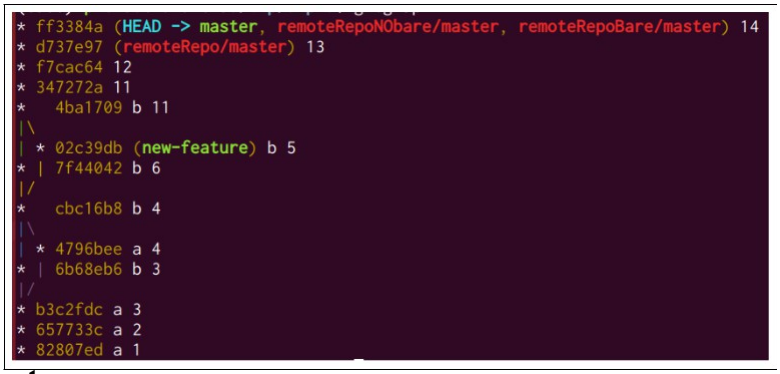
```
(base) root@kali:~/ISI/Practica 2/alpha# git push remoteRepoBare master
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Escribiendo objetos: 100% (4/4), 277 bytes | 277.00 KiB/s, listo.
Total 4 (delta 0), reusado 0 (delta 0), pack-reusado 0
To ../remoteRepoBare/
   a094e1d..4855ba8  master -> master
```

21. **Observa el grafo de commits en alpha y remoteRepoBare: contienen los mismos commits.**

Al terminar el ultimo apartado deberás tener en remoteRepoBare un grafo de commits idéntico al de la siguiente gura, aunque con hashes de los commits distintos claro:



Y en alpha el grafo de commits deber a ser como este otro, coincidiendo los hashes con los de remoteRepoBare:



Normas de entrega

1. **Pon tu nombre y apellidos al principio de un fichero (de texto o pdf) que tienes que entregar con los comandos que has utilizado para dar respuesta a cada apartado. Responde a todos los ejercicios en el fichero, dejando claramente identificado a que pregunta se responde.**

2. **Una vez hayas realizado todos los ejercicios de esta prueba entrega el fichero y los directorios de los repositorios creados en el enlace del Aula Virtual.**