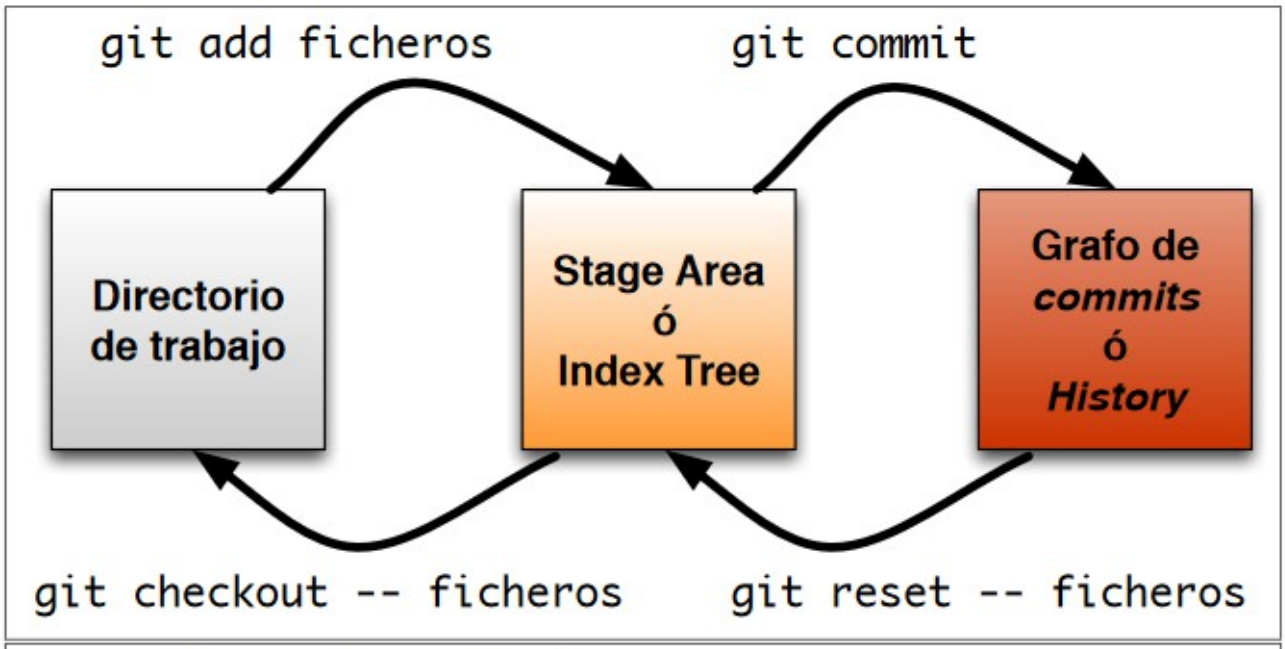
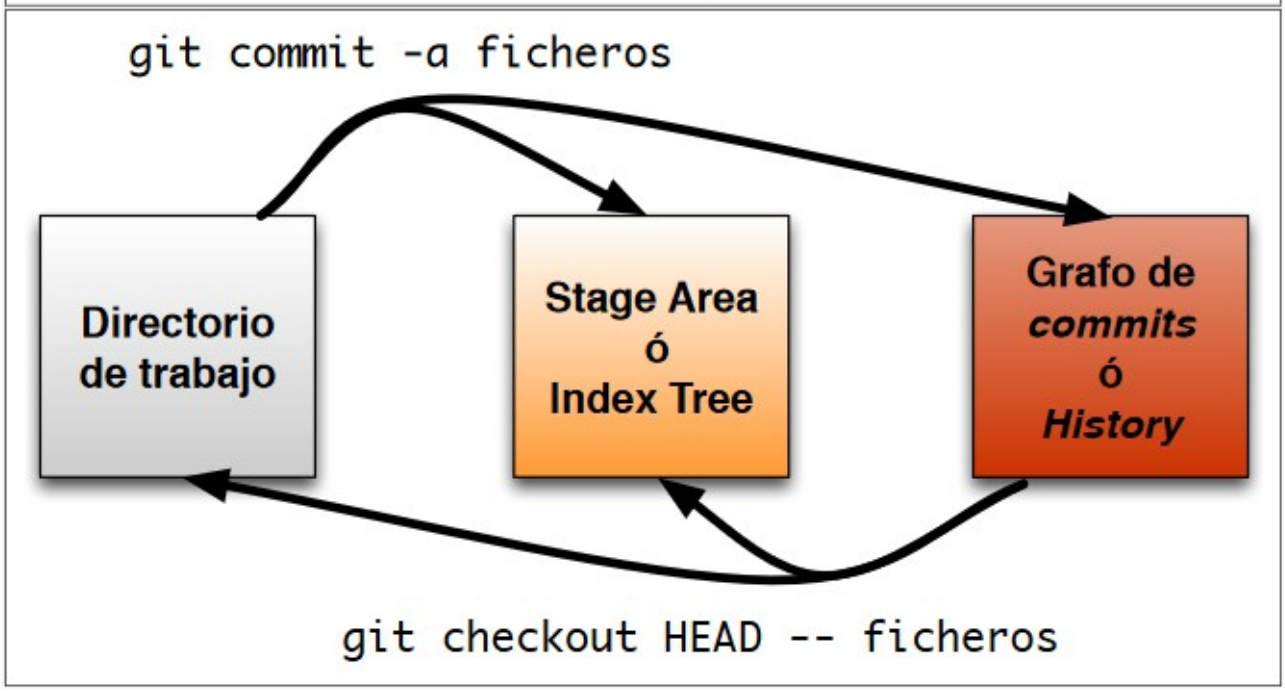
Raúl Gil López
53842198-A

Práctica: git

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

Resumen de comandos git que pueden ayudarte en la realización de los ejercicios 1, 2 y 3



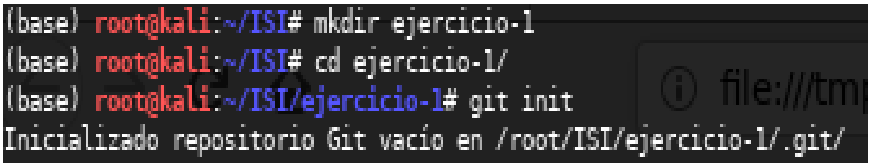


git init
git config --global user.name "name"
git config --global user.email "email"
git status
git add
git log
git diff
git diff --staged
git rm
git checkout -- filename
git reset HEAD filename
git checkout (commit hash) filename
git log --all --decorate --oneline --graph
alias gitgraph="git log --all --decorate --oneline --graph"

1. Ejercicio 1

Anota en un documento los comandos que vas utilizando para dar respuesta a cada apartado:

1. **Crea un repositorio en un directorio denominado ejercicio-1.**

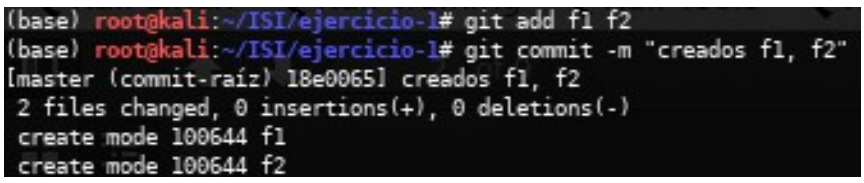


Creamos un directorio e iniciamos dentro de el el repositorio con " git init "

2. **Crea dos ficheros vacíos, de nombre f1, f2**

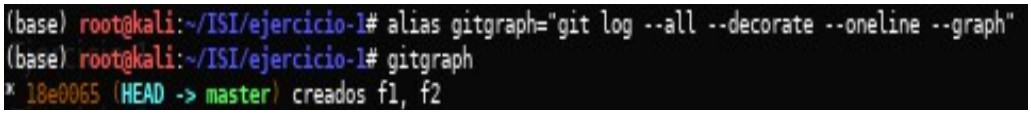
Utilizamos el comando "touch" para crear archivos

3. **Crea un primer commit que añada los dos ficheros, con el comentario C1: creados f1, f2.**



Con el comando git add <archivos> introducimos los ficheros en el stage area.
Y ejecutando el comando "git commit -m <mensaje>" lo introducimos al grafo de commits confirmando los ficheros.

4. **Utiliza, ahora y siempre que lo consideres oportuno, el comando gitgraph para observar el grafo de commits realizados hasta el momento, las ramas, y donde esta apuntando HEAD.**

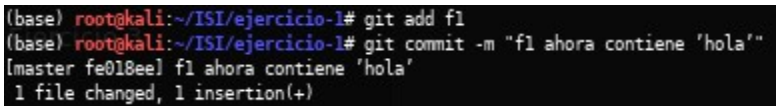


Utilizamos el alias para que nos reconozca el comando "gitgraph" y ver visualmente el grafo de commits

5. **Modifica el fichero f1 para que incluya una linea con este texto: hola.**

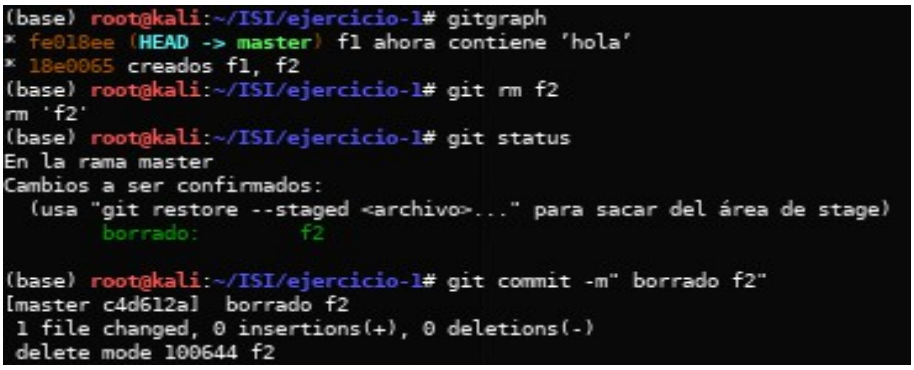
Hacemos "echo hola" y lo guardamos en f1

6. **Crea un segundo commit que refleje este cambio con el siguiente comentario: C2: f1 ahora contiene 'hola'**



Primero lo agregamos a stage area con "git add" y luego comprometemos con "git commit"

7. **Elimina del repositorio el fichero f2, creando un commit con el siguiente comentario: C3: borrado f2.**



Mediante el comando "git rm <fichero>" eliminamos de stage area y working area el fichero elegido.
Solo queda confirmarlo con "git commit -m <mensaje>"

8. **Recupera el contenido del fichero f2 para que aparezca en el área de trabajo (el directorio del repositorio).**

```
(base) root@kali:~/ISI/ejercicio-1# gitgraph
* c4d612a (HEAD -> master)  borrado f2
* fe018ee f1 ahora contiene 'hola'
* 18e0065 creados f1, f2
(base) root@kali:~/ISI/ejercicio-1# git checkout fe018ee f2
Actualizada 1 ruta para b3b170c
```

Usando el comando "git checkout <commit id> <fichero>" actualizamos el fichero elegido del directorio de trabajo a la versión seleccionada del commit id.

9. **Modifica el ultimo commit (ojo, no a~nadas un nuevo commit) para que contenga el fichero f2 recuperado en el paso anterior, y un fichero f1 que contenga dos lineas, en la primera hola y en la segunda adios. Haz que el nuevo comentario de este ultimo commit sea C3: modificado f1 y añadido de nuevo f2.**

```
(base) root@kali:~/ISI/ejercicio-1# echo adios >> f1
(base) root@kali:~/ISI/ejercicio-1# cat f1
hola
adios
(base) root@kali:~/ISI/ejercicio-1# ls
f1  f2
(base) root@kali:~/ISI/ejercicio-1# git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
        nuevo archivo:  f2

Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
        modificado:     f1

(base) root@kali:~/ISI/ejercicio-1# git add f1
(base) root@kali:~/ISI/ejercicio-1# git commit --amend -m "modificado f1 y a~nadido de nuevo f2."
[master 9919ca3] modificado f1 y a~nadido de nuevo f2.
 Date: Tue Nov 17 12:56:03 2020 +0100
 1 file changed, 1 insertion(+)
(base) root@kali:~/ISI/ejercicio-1# gitgraph
* 9919ca3 (HEAD -> master) modificado f1 y a~nadido de nuevo f2.
* fe018ee f1 ahora contiene 'hola'
* 18e0065 creados f1, f2
```

Primero modificamos f1 añadiendole una nueva linea (adios) con "echo adios >> f1"
Con "git status" vemos que nos falta agregar f1 a stage area. Por lo que hacemos "git add" y para modificar el commit anterior usamos "git commit –amend -m <mensaje>"

2. Ejercicio 2

Anota en un documento los comandos que vas utilizando para dar respuesta a cada apartado:

1. **Crea un repositorio de nombre ejercicio-2.**

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

Al igual que en el ejercicio anterior, creamos un directorio con "mkdir" y dentro de el arrancamos el repositorio con "git init"

2. **Añade al repositorio un fichero denominado s1 con las siguientes lineas:**

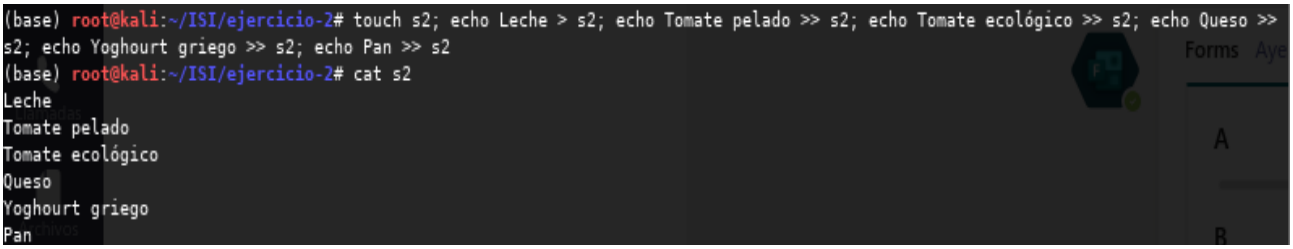
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego

Crea un primer commit con el siguiente comentario: C1: añadido s1

```
(base) root@kali:~/ISI/ejercicio-2# touch s1; echo Leche > s1; echo Tomate pelado >> s1; echo Tomate ecológico >> s1; echo Queso >>
s1; echo Yoghourt griego >> s1
(base) root@kali:~/ISI/ejercicio-2# cat s1
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
(base) root@kali:~/ISI/ejercicio-2# git add s1; git commit -m "añadido s1"
[master (commit-raiz) 3363367] añadido s1
 1 file changed, 5 insertions(+)
 create mode 100644 s1
(base) root@kali:~/ISI/ejercicio-2# gitgraph
* 3363367 (HEAD -> master) añadido s1
```

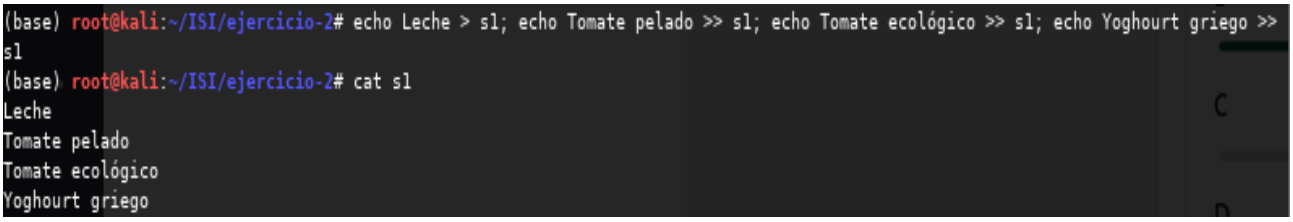
Lo primero que hacemos es un "touch" para crear el archivo s1 y le vamos metiendo echos para escribir en el. Finalmente lo añadimos al stage area con "git add" y al grafo de commits con "git commit"

3. **Crea un fichero s2 que contenga las mismas lineas que s1, y una nueva al final: Pan**



Igual que con el fichero anterior, lo creamos y vamos añadiendo echos al mismo.

4. **Modifica el fichero s1 para eliminar Queso**



Mediante echos actualizamos el fichero s1 para eliminar "Queso".

5. **Añade al stage area (tambien denominado index tree) los ficheros s1 y s2**

Usamos el comando "git add <archivos>"

6. **Utiliza el comando git diff para comparar las versiones de s1 y s2 que hay en el área de trabajo con las versiones del stage area. ¿Que diferencias hay?**

El comando "git diff" nos da la diferencia en los ficheros del directorio de trabajo con los archivos almacenados en el stage area. Como hemos añadido los archivos s1 y s2 a stage area, en ambos sitios van a estar sincronizados. Comprobamos que el comando no nos arroja ninguna diferencia.

7. **Utiliza el comando git diff para comparar las versiones de s1 y s2 que hay en el stage area con las versiones que hay en HEAD. >Que diferencias hay?**

```
(base) root@kali:~/ISI/ejercicio-2# git diff --staged s1 s2
diff --git a/s1 b/s1
index 5aad170..eb8dc85 100644
--- a/s1
+++ b/s1
@@ -1,5 +1,4 @@
 Leche
 Tomate pelado
 Tomate ecológico
-Queso
 Yoghourt griego
diff --git a/s2 b/s2
new file mode 100644
index 0000000..ae0c9c0
--- /dev/null
+++ b/s2
@@ -0,0 +1,6 @@
+Leche
+Tomate pelado
+Tomate ecológico
+Queso
+Yoghourt griego
+Pan
```

Con el comando "git diff –staged <archivos>" vemos las diferencias de los archivos almacenados en el stage area con el último commit (HEAD).

En stage area tenemos:

s1: Todo menos Queso

s2:Todo mas pan

En el último commit tenemos:

s1:Todo

s2:Nada, no existe el fichero

Ejecutando el comando nos arroja que en el stage area, en s1 no hay Queso y en el ultimo commit si, y en s2 que en el stage area hay Todo mas pan y en el ultimo commit nada, ya que no existía.

8. **Crea un nuevo commit con el siguiente comentario: C2: s1 no tiene queso, s2 tiene pan**

```
(base) root@kali:~/ISI/ejercicio-2# git commit -m "s1 no tiene queso, s2 tiene pan"
[master 6200f2a] s1 no tiene queso, s2 tiene pan
 2 files changed, 6 insertions(+), 1 deletion(-)
 create mode 100644 s2
(base) root@kali:~/ISI/ejercicio-2# gitgraph
* 6200f2a (HEAD -> master) s1 no tiene queso, s2 tiene pan
* 3363367 añadido s1
```

Agregamos al grafo de commits uno nuevo con "git commit -m"

9. **>Hay alguna diferencia entre las tres versiones de s1 y s2 en el area de trabajo, en staging area y en el grafo de commits?**

```
(base) root@kali:~/ISI/ejercicio-2# git diff
(base) root@kali:~/ISI/ejercicio-2# git diff --staged
```

No hay ninguna diferencia entre las versiones del directorio de trabajo, stage area y grafo de commits. Esto es lógico ya que hemos agregado a stage area los archivos del directorio de trabajo y luego los hemos comprometido a si que los archivos están sincronizados.

10. **Utiliza el comando git diff para comparar las versiones de s1 y s2 que hay en el último commit con las que habia en el primer commit.**

```
(base) root@kali:~/ISI/ejercicio-2# git diff 3363367 s1 s2
diff --git a/s1 b/s1
index 5aad170..eb8dc85 100644
--- a/s1
+++ b/s1
@@ -1,5 +1,4 @@
 Leche
 Tomate pelado
 Tomate ecológico
-Queso
 Yoghourt griego
diff --git a/s2 b/s2
new file mode 100644
index 0000000..ae0c9c0
--- /dev/null
+++ b/s2
@@ -0,0 +1,6 @@
+Leche
+Tomate pelado
+Tomate ecológico
+Queso
+Yoghourt griego
+Pan
```

Mediante el comando "git diff <commit id> <archivos> podemos ver las diferencias de los archivos actuales con los archivos de un commit especifico.

En el commit actual tenemos:

s1: todo menos queso

s2: Todo mas Pan

En el commit anterior teníamos:

s1: Todo

s2: Nada, no existía el archivo

Por lo que el comando nos arroja que actualmente en s1 Queso no está y en s2 actualmente está Todo mas pan y en el anterior commit no había nada, debido a que no existía el archivo.

11. **Elimina el fichero s2, creando un nuevo commit con el siguiente comentario: C3: borrado s2**

```
(base) root@kali:~/ISI/ejercicio-2# git rm s2
rm 's2'
(base) root@kali:~/ISI/ejercicio-2# git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
        borrado:        s2

(base) root@kali:~/ISI/ejercicio-2# git commit -m"borrado s2"
[master fff1c7e] borrado s2
 1 file changed, 6 deletions(-)
 delete mode 100644 s2
```

Usamos “git rm <archivo>” para eliminar un archivo y lo comprometemos con “git commit -m”

12. **¿Que diferencias hay entre las versiones de s1 de working area, staging area y HEAD? Utiliza un comando para comprobarlo.**

```
(base) root@kali:~/ISI/ejercicio-2# git diff s1
(base) root@kali:~/ISI/ejercicio-2# git diff --staged s1
```

No hay diferencias del archivo s1 en las versiones del directorio de trabajo, stage area y HEAD ya que no hemos hecho modificaciones en s1.

13. **¿Que diferencias hay entre las versiones de s2 de working area, staging area y HEAD? Utiliza un comando para comprobarlo.**

```
(base) root@kali:~/ISI/ejercicio-2# git diff s2
fatal: argumento ambiguo 's2': revisión desconocida o ruta fuera del árbol de trabajo.
Use '--' para separar las rutas de las revisiones, de esta manera:
'git <comando> [<revisión>...] -- [<archivo>...]'
(base) root@kali:~/ISI/ejercicio-2# git diff --staged s2
fatal: argumento ambiguo 's2': revisión desconocida o ruta fuera del árbol de trabajo.
Use '--' para separar las rutas de las revisiones, de esta manera:
'git <comando> [<revisión>...] -- [<archivo>...]'
```

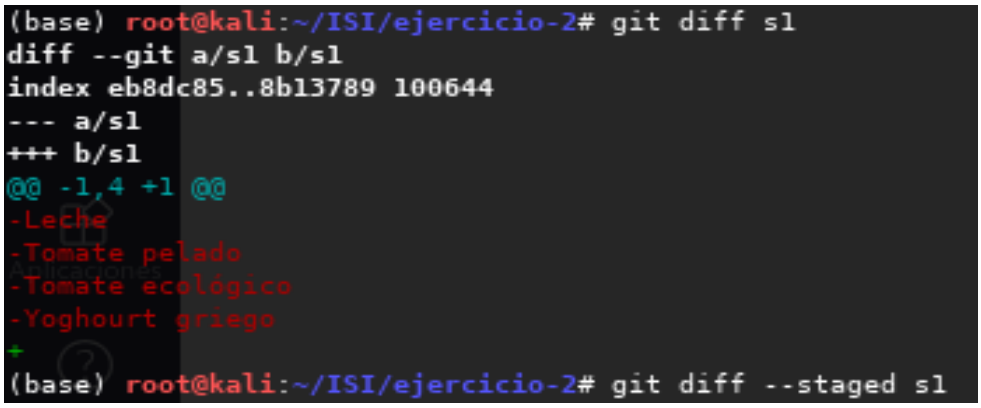
Como hemos eliminado el archivo s2 de todas las versiones y hemos comprometido, el terminal no reconoce s2.

14. **Edita el fichero s1, eliminando todas sus lineas, pero no borres el fichero.**

```
(base) root@kali:~/ISI/ejercicio-2# echo "" > s1
```

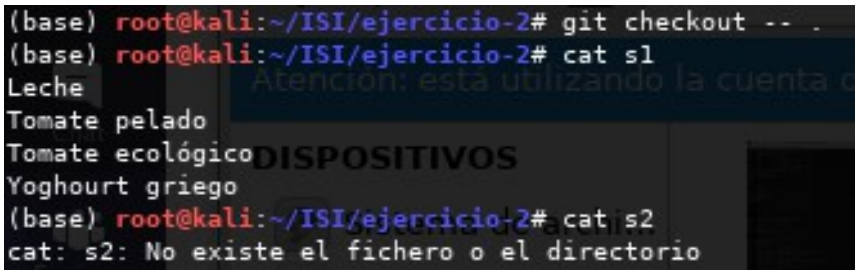
Borramos todas las lineas de s1 con echo “” > s1

15. ¿Que diferencias hay entre las versiones de s1 de working area, staging area y HEAD?



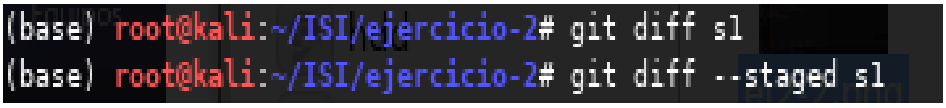
Como hemos modificado s1 eliminando todas las lineas pero no lo hemos actualizado en el stage area ni en el grafo de commits, el comando git diff nos arroja que en el directorio de trabajo falta Leche, Tomate pelado,etc.
En el directorio de trabajo, en s1 no hay nada. Sin embargo, en stage area y HEAD s1 sigue sincronizado porque no se ha vuelto a modificar.

16. Recupera al área de trabajo la versión de s1 que hay en el stage area



Para recuperar la version de stage area usamos "git checkout – .". Ponemos "." para que actualice todos los archivos del área de trabajo.
Usando "cat" vemos que el archivo s1 tiene Todo menos Queso como en stage area y s2 no existe.

17. ¿Que diferencias hay entre las versiones de s1 de working area, staging area y HEAD?



Hasta ahora, stage area y HEAD están sincronizados, por lo que si recupero s1 de stage area en el directorio de trabajo va a seguir estando sincronizado con todas las versiones.

18. **Edita s1 añadiendole una linea al final: Fregona**

```
(base) root@kali:~/ISI/ejercicio-2# echo Fregona >> s1
(base) root@kali:~/ISI/ejercicio-2# cat s1
Leche
Tomate pelado
Tomate ecológico
Yoghourt griego
Fregona
```

19. **Añade al stage area s1**

```
(base) root@kali:~/ISI/ejercicio-2# git add s1
```

Agregamos s1 a stage area

20. **>Que diferencias hay entre las versiones de s1 de working area, staging area y HEAD?**

```
(base) root@kali:~/ISI/ejercicio-2# git diff s1
(base) root@kali:~/ISI/ejercicio-2# git diff --staged s1
diff --git a/s1 b/s1
index eb8dc85..42d0d40 100644
--- a/s1
+++ b/s1
@@ -2,3 +2,4 @@ Leche
 Tomate pelado
 Tomate ecológico
 Yoghourt griego
+Fregona
```

Ahora el directorio de trabajo y el stage area están sincronizados pero no con HEAD.
A si que al ejecutar “git diff –staged s1” nos va a dar las diferencias del stage area con HEAD.
Concretamente en stage area hay Fregona, en HEAD no ya que no lo hemos comprometido en ningún momento.

21. **Edita s1 y borra todas sus lineas**

```
(base) root@kali:~/ISI/ejercicio-2# echo "" > s1
```

22. ¿Que diferencias hay entre las versiones de s1 de working area, staging area y HEAD?

```
(base) root@kali:~/ISI/ejercicio-2# git diff s1
diff --git a/s1 b/s1
index 42d0d40..8b13789 100644
--- a/s1
+++ b/s1
@@ -1,5 +1 @@
-Leche
-Tomate pelado
-Tomate ecológico
-Yoghourt griego
-Fregona
+
(base) root@kali:~/ISI/ejercicio-2# git diff --staged s1
diff --git a/s1 b/s1
index eb8dc85..42d0d40 100644
--- a/s1
+++ b/s1
@@ -2,3 +2,4 @@ Leche
 Tomate pelado
 Tomate ecológico
 Yoghourt griego
+Fregona
```

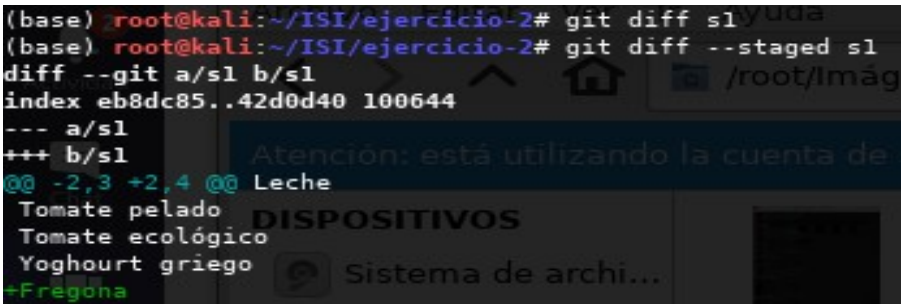
Ahora hemos borrado el contenido de s1 sin añadirlo a stage area por lo que el directorio de trabajo no está sincronizado con stage area. S1 en el directorio de trabajo le falta todo mas fregona.
Como en ningun momento hemos comprometido vamos a tener las mismas diferencias que antes en el stage area y HEAD.

23. Cambiamos de opinion. Queremos recuperar la version de staging area.

```
(base) root@kali:~/ISI/ejercicio-2# git checkout -- .
(base) root@kali:~/ISI/ejercicio-2# cat s1
Leche
Tomate pelado
Tomate ecológico
Yoghourt griego
Fregona
(base) root@kali:~/ISI/ejercicio-2# cat s2
cat: s2: No existe el fichero o el directorio
```

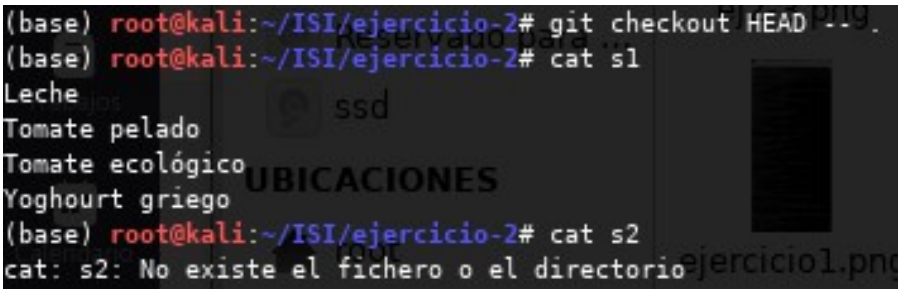
Mediante "git checkout – ." actualizamos todos los archivos de stage area en nuestro directorio de trabajo.

24. ¿Que diferencias hay entre las versiones de s1 de working area, staging area y HEAD?



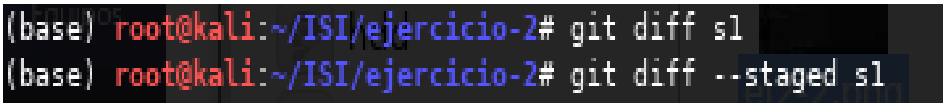
Como hemos sincronizado el directorio de trabajo con stage area, el comando "git diff s1" no arroja nada, no hay diferencias. Pero como antes, no hemos comprometido nada por lo que en stage area y HEAD nos darán las mismas diferencias.

25. Volvemos a cambiar de opinion: queremos recuperar la version de HEAD, tanto en staging area como en working area



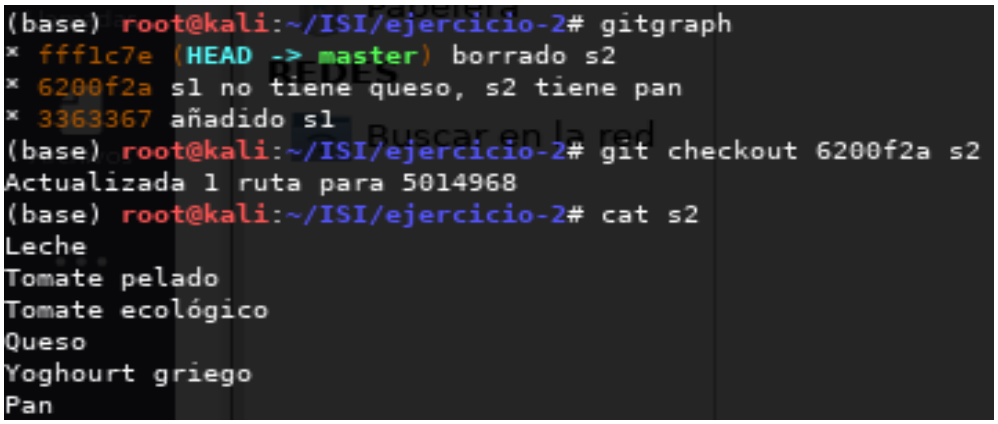
Para recuperar la versión de HEAD usamos "git checkout HEAD – ." para actualizar todos los archivos del area de trabajo y stage area.

26. Comprueba que vuelves a tener los mismos contenidos de s1 en el area de trabajo, en el stage area y en el grafo de commits (en HEAD).



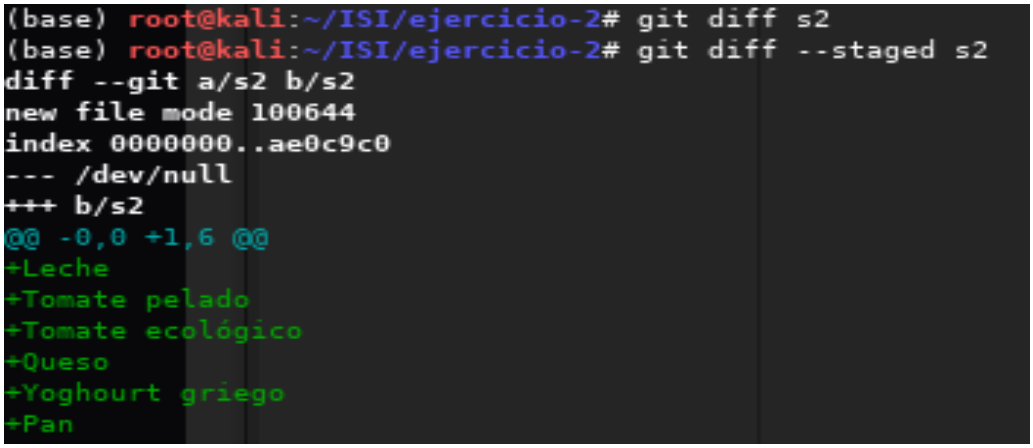
Como hemos sincronizado todas las versiones con la de HEAD, no nos saldrá ninguna diferencia.

27. **Ahora no tenemos s2 en el área de trabajo ya que HEAD esta en el commit en el que borramos s2. Recupera el contenido de s2 que había en el commit anterior (el etiquetado C2: s1 no tiene queso, s2 tiene pan).**



En mi caso, el commit anterior donde s1 no tiene queso y s2 tiene pan, empieza con el identificador 6200f2a por lo que actualizamos el archivo s2 de ese commit al directorio de trabajo.

28. **¿Que diferencias hay entre las versiones de s2 de working area, staging area y HEAD?**



S2 está sincronizado en el irectorio de trabajo y en stage area pero no en el stage area y en HEAD ya que en HEAD no existe el fichero s2.

29. **Crea un nuevo commit que contenga el fichero s2 recuperado y el comentario: C4: recuperado s2**

```
(base) root@kali:~/ISI/ejercicio-2# git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
        nuevo archivo:  s2

(base) root@kali:~/ISI/ejercicio-2# git commit -m "recuperado s2"
[master 62c037d] recuperado s2
 1 file changed, 6 insertions(+)
 create mode 100644 s2
```

30. **¿Que diferencias hay entre las versiones de s2 de working area, staging area y HEAD?**

```
(base) root@kali:~/ISI/ejercicio-2# git diff s2
(base) root@kali:~/ISI/ejercicio-2# git diff --staged s2
```

Ahora tenemos todas las versiones de s2 sincronizadas a si que no tendremos diferencias entre ellas.

31. **Crea los siguientes cheros: p1.java, p2.java**
32. **Crea los siguientes cheros: p1.class, p2.class**
33. **Crea los siguiente ficheros : p1.log**
34. **Crea un subdirectorio logs y dentro de el dos ficheros, log1.log, log2.log**

```
(base) root@kali:~/ISI/ejercicio-2# touch p1.java; touch p2.java
(base) root@kali:~/ISI/ejercicio-2# touch p1.class; touch p2.class
(base) root@kali:~/ISI/ejercicio-2# touch log1.log
(base) root@kali:~/ISI/ejercicio-2# mkdir logs
(base) root@kali:~/ISI/ejercicio-2# cd logs
(base) root@kali:~/ISI/ejercicio-2/logs# touch log1.log; touch log2.log
```

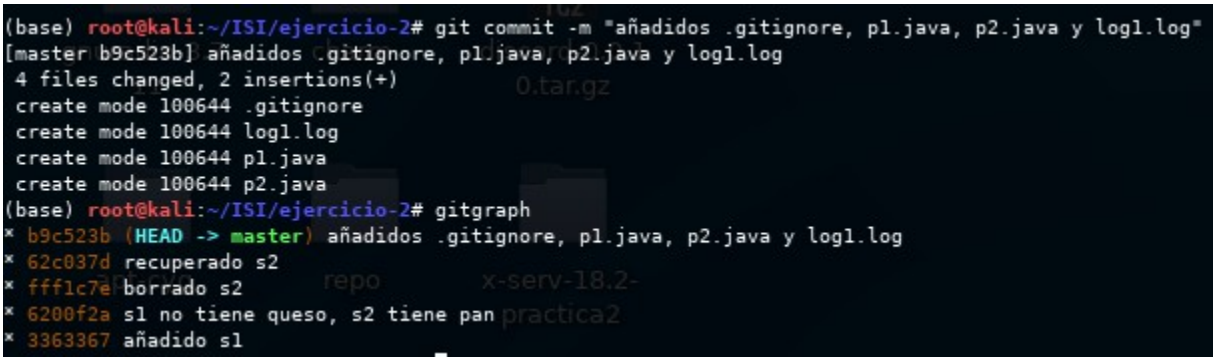
35. Crea un fichero .gitignore en el directorio raiz del repositorio que impida que se añadan al repositorio los ficheros terminados en .class y los ficheros terminados en .log que esten en el directorio logs.

36. Ejecuta el comando git add . que añade todo lo que ha cambiado en el directorio actual.

37. Comprueba que solo estan en stage area los cheros .gitignore, p1.java, p2.java y log1.log

Al crear el archivo .gitignore, vamos a introducir en el las restricciones para que no se agregen al stage area archivos que no queremos.
*Con *.class impediremos que los archivos acabados en .class del directorio raiz se añadan y con logs/*.log impediremos que se agregen los archivos del subdirectorio logs acabados en .log.*

38. Crea un ultimo commit con el comentario: C5: a~nadidos .gitignore, p1.java, p2.java y log1.log



Resumen de comandos git que pueden ayudarte en la realización de los ejercicios 3

git log
git log --all --decorate --oneline --graph
git branch (nombre-de-rama)
git checkout (nombre-de-rama)
git commit -a -m "mensaje del commit"
git diff master..rama
git merge (nombre-de-rama)
git branch --merged
git branch -d (nombre-de-rama)
git branch -D (nombre-de-rama)
git merge --abort
git checkout (hash-commit)
git stash
git stash list
git stash list -p
git stash apply
git stash pop
git stash apply (referencia-de-stash)
git stash save "descripcion"

3. Ejercicio 3

Anota en un documento los comandos que vas utilizando para dar respuesta a cada apartado:

1. **Crea un nuevo repositorio git en un directorio de nombre ejercicio-3.**
2. **Crea un fichero s1 con los siguientes contenidos:**

Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego

```
(base) root@kali:~/ISI# mkdir ejercicio-3
(base) root@kali:~/ISI# cd ejercicio-3
(base) root@kali:~/ISI/ejercicio-3# git init
Inicializado repositorio Git vacío en /root/ISI/ejercicio-3/.git/
(base) root@kali:~/ISI/ejercicio-3# touch s1; echo Leche > s1; echo Tomate pelado >> s1; echo Tomate ecológico >> s1; echo Queso >> s1; echo Yoghourt griego >> s1
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
```

3. **Añade a stage area s1 y crea un commit con el siguiente comentario: C1: creado s1**

```
(base) root@kali:~/ISI/ejercicio-3# git add s1; git commit -m "creado s1"
```

4. **Haz una copia de s1 con el nombre s2.**

```
(base) root@kali:~/ISI/ejercicio-3# cp s1 s2
(base) root@kali:~/ISI/ejercicio-3# cat s2
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
```

Hacemos uso del comando cp para copiar archivos.

5. **Añade a stage area s2 y crea un commit con el siguiente comentario: C2: creado s2**

```
(base) root@kali:~/ISI/ejercicio-3# git add s2; git commit -m "creado s2"
```

5. **Crea una rama devel**
6. **Crea una rama bugFix**
7. **Comprueba el estado del grafo de commits con gitgraph y con git branch**

```
(base) root@kali:~/ISI/ejercicio-3# git branch devel
(base) root@kali:~/ISI/ejercicio-3# git branch bugFix
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* 2bbd9c8 (HEAD -> master, devel, bugFix) creado s2
* 724e871 creado s1
(base) root@kali:~/ISI/ejercicio-3# git branch
  bugFix
  devel
* master
```

Usamos git branch <nombre> para crear una rama.

8. **Haz que HEAD apunte a devel**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout devel
Cambiado a rama 'devel'
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* 2bbd9c8 (HEAD -> devel, master, bugFix) creado s2
* 724e871 creado s1
(base) root@kali:~/ISI/ejercicio-3# git branch
  bugFix
* devel
  master
```

Usamos git checkout <rama> para que HEAD apunte a la rama indicada.

9. **Edita s1, añadiendo Fregona**
10. **Añade a stage area y crea un nuevo commit con el siguiente comentario: C3: añadida Fregona**
11. **Muestra con cat el contenido de s1**

```
(base) root@kali:~/ISI/ejercicio-3# echo Fregona >> s1
(base) root@kali:~/ISI/ejercicio-3# git add s1; git commit -m "añadida fregona"
[devel 0052dc3] añadida fregona
 1 file changed, 1 insertion(+)
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
Fregona
```

12. **Cambia HEAD a la rama bugFix**
13. **Muestra con cat el contenido de s1**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout bugFix
Cambiado a rama 'bugFix'
(base) root@kali:~/ISI/ejercicio-3# git branch
* bugFix
  devel
  master
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
```

14. **Vuelve a devel**
15. **Muestra con cat el contenido de s1**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout devel
Cambiado a rama 'devel'
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
Fregona
```

16. **Vuelve a bugFix**
17. **Edita s1: cambia Leche por Leche desnatada**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout bugFix
Cambiado a rama 'bugFix'
(base) root@kali:~/ISI/ejercicio-3# echo Leche desnatada > s1; echo Tomate pelado >> s1; echo Tomate ecológico >> s1; echo Queso >> s1; echo Yoghourt griego >> s1
(base) root@kali:~/ISI/ejercicio-3# git status
En la rama bugFix
Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
        modificado:     s1

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
```

18. **Comprueba que diferencias hay entre el área de trabajo, el stage area y HEAD. Recuerda que tienes a tu disposición los comandos git log, gitgraph, git status, git diff, git diff --staged**

```
(base) root@kali:~/ISI/ejercicio-3# git diff
diff --git a/s1 b/s1
index 5aad170..9384b0e 100644
--- a/s1
+++ b/s1
@@ -1,4 +1,4 @@
-Leche
+Leche desnatada
 Tomate pelado
 Tomate ecológico
 Queso
(base) root@kali:~/ISI/ejercicio-3# git diff --staged
```

En la rama bugFix hemos cambiado en el directorio de trabajo la linea Leche por Leche desnatada. En stage area tenemos Leche por lo que detecta esa diferencia.
Sin embargo, no hemos cambiado nada ni en stage area ni en HEAD por lo que no detecta diferencias en ambos.

19. **Añade s1 a stage area y crea un nuevo commit con el comentario: C4: Cambiada Leche por Leche desnatada**

```
(base) root@kali:~/ISI/ejercicio-3# git add s1; git commit -m "Cambiada Leche por Leche desnatada"
[bugFix c480515] Cambiada Leche por Leche desnatada
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) root@kali:~/ISI/ejercicio-3# git log
commit c480515bba436bf65d83ddad5ed98cb72b241d40 (HEAD -> bugFix)
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 11:41:29 2020 +0100

    Cambiada Leche por Leche desnatada

commit 2bbd9c8b979c5a9324f9038587f4e8b0e17f792d (master)
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 11:30:11 2020 +0100

    creado s2

commit 724e871b2158b94342301f23044a9e18de7710b9
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 11:27:34 2020 +0100

    creado s1
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* c480515 (HEAD -> bugFix) Cambiada Leche por Leche desnatada
| * 0052dc3 (devel) añadida fregona
|/
* 2bbd9c8 (master) creado s2
* 724e871 creado s1
```

Ahora añadimos s1 a stage area y lo comprometemos.

Merge con Fast Forward

1. **Comprueba el estado actual (rama en la que estas, ficheros en el área de trabajo, etc.)**
2. **Cambiate a la rama master (i.e. haz que HEAD apunte a master)**

```
(base) root@kali:~/ISI/ejercicio-3# git branch
* bugFix
  devel
  master
(base) root@kali:~/ISI/ejercicio-3# git checkout master
Cambiado a rama 'master'
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* 44e91cb (bugFix) Cambiada Leche por Leche desnatada
| * 6eb664d (devel) añadida fregona
|/
* 0f8e196 (HEAD -> master) creado s2
* f6d0bd7 creado s1
```

3. **Comprueba con este comando las diferencias entre las ramas master y devel: git diff master..devel**

```
(base) root@kali:~/ISI/ejercicio-3# git diff master..devel
diff --git a/s1 b/s1
index 5aad170..494122a 100644
--- a/s1
+++ b/s1
@@ -3,3 +3,4 @@ Tomate pelado
 Tomate ecológico
 Queso
 Yoghourt griego
+Fregona
```

En la rama master tenemos lo mismo que en la rama devel menos Fregona por lo que reporta esa diferencia.

4. **Comprueba con este comando las diferencias entre las ramas master y bugFix: git diff master..bugFix**

```
(base) root@kali:~/ISI/ejercicio-3# git diff master..bugFix
diff --git a/s1 b/s1
index 5aad170..9384b0e 100644
--- a/s1
+++ b/s1
@@ -1,4 +1,4 @@
-Leche
+Leche desnatada
 Tomate pelado
 Tomate ecológico
 Queso
```

En la rama master tenemos lo mismo que en la rama bugFix salvo en la primera linea que master tiene Leche y bugFix tiene Leche desnatada.

5. **Observa el grafo de commits con gitgraph**

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* 44e91cb (bugFix) Cambiada Leche por Leche desnatada
| * 6eb664d (devel) añadida fregona
|/
* 0f8e196 (HEAD -> master) creado s2
* f6d0bd7 creado s1
```

6. **Mezcla en la rama master los cambios de la rama devel. Antes de hacerlo, trata de predecir que tipo de merge realizar git. ¿Crear un nuevo commit o hará un fast-forward?**

En este escenario, como solo hemos hecho cambios en una de las ramas a fusionar, hará un fast-forward por lo que se copiará la nueva linea sobrescribiendose en la vieja.

7. **Observa el grafo de commits con gitgraph. Describe que ha ocurrido con las etiquetas de las ramas. Observa que no se ha creado un commit para la mezcla en esta ocasión. Se ha hecho un fast-forward**

```
(base) root@kali:~/ISI/ejercicio-3# git merge devel
Actualizando 0f8e196..6eb664d
Fast-forward
 s1 | 1 +
 1 file changed, 1 insertion(+)
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* 44e91cb (bugFix) Cambiada Leche por Leche desnatada
| * 6eb664d (HEAD -> master, devel) añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
```

Como en la rama master no se ha modificado s1 pero en la rama devel si, simplemente se copia las lineas nuevas de devel a master.

8. **Comprueba el contenido de s1**

```
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
Fregona
```

Vemos que se ha añadido solo Fregona de la rama devel.

Borrar ramas

1. **Ejecutando el siguiente comando puedes comprobar que ramas están mezcladas con la rama en la que estas (master en este caso): git branch –merged**

```
(base) root@kali:~/ISI/ejercicio-3# git branch
  bugFix
  devel
* master
(base) root@kali:~/ISI/ejercicio-3# git branch --merged
  devel
* master
```

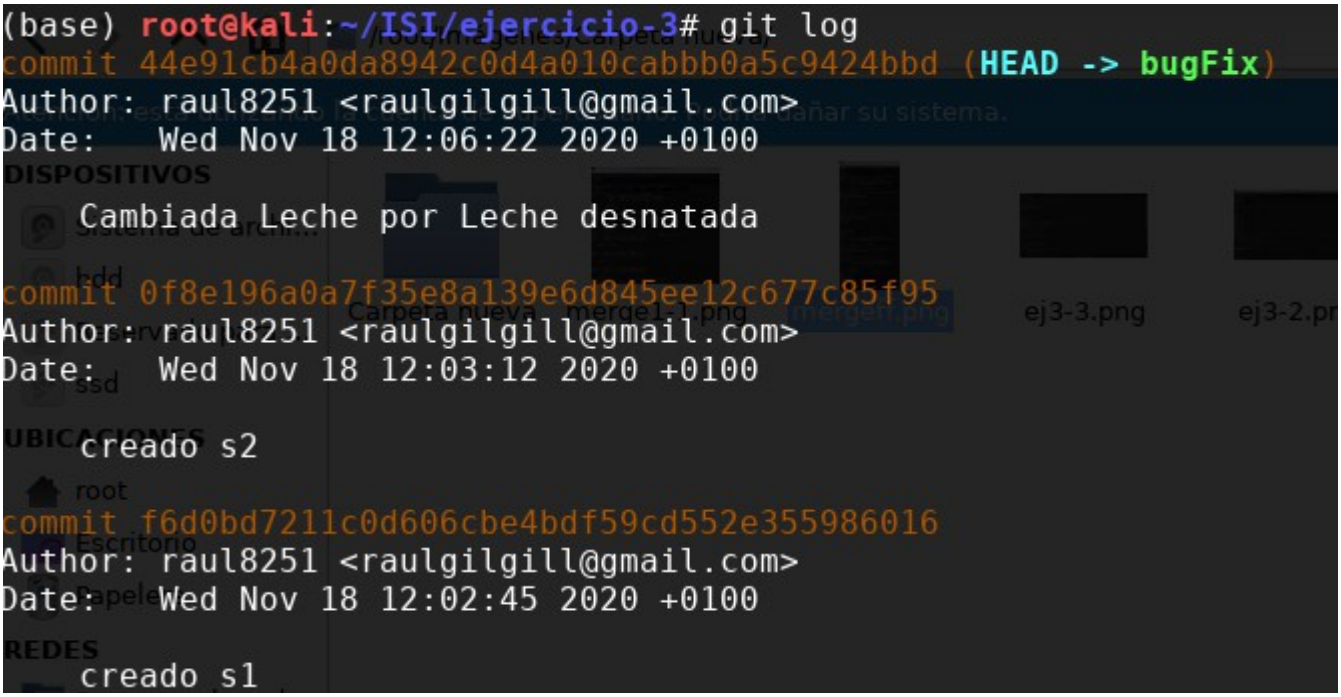
Vemos que la rama devel se ha fusionado a master.

2. **Ahora es seguro borrar la rama devel pues sus cambios están aplicados en la rama master. Antes de borrar inspecciona el grafo de commits. Ahora borra devel.**

3. **Comprueba que efectos ha tenido sobre el grafo de commits.**

Cuando eliminamos la rama devel se borra la etiqueta del grafo de commits.

4. **¿Que commits forman parte de la rama bugFix?**



Hacemos git checkout bugFix para entrar en la rama y luego git log para ver los commits de esa rama.

5. **¿Que commits forman parte de la rama master?**

```
(base) root@kali:~/ISI/ejercicio-3# git log
commit 6eb664df47576a24cab86f687c99fe6b2ddca4d9 (HEAD -> master)
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 12:04:42 2020 +0100

    añadida fregona

commit 0f8e196a0a7f35e8a139e6d845ee12c677c85f95
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 12:03:12 2020 +0100

    creado s2

commit f6d0bd7211c0d606cbe4bdf59cd552e355986016
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 12:02:45 2020 +0100

    creado s1
```

Hacemos git checkout master y posteriormente git log para ver los commits de la rama master.

6. **Intenta ahora borrar la rama bugFix. Describe y explica por que no puedes borrarla. Estudia para ello el contenido de los ficheros de cada rama.**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout master
Cambiado a rama 'master'
(base) root@kali:~/ISI/ejercicio-3# git branch -d bugFix
error: La rama 'bugFix' no ha sido fusionada completamente.
Si estás seguro de querer borrarla, ejecuta 'git branch -D bugFix'.
```

En la rama bugFix hemos modificado s1 pero tambien lo hemos hecho en la rama master al fusionarla con devel, por lo que habría que hacer un commit para poder fusionarlas

Merge 1

1. **Si te empeñas en borrar la rama bugFix puedes hacerlo con git branch -D bugFix. NO LO HAGAS. Recuerda, una rama no es mas que una etiqueta. Si la borrases perder as la posibilidad de acceder al commit C4 pues no estar a en ninguna rama. Solo sabiendo su commit id podrás hacer checkout a ese commit. En lugar de eso, vamos a mezclar la rama bugFix en master y luego ya sera seguro borrarla sin perder cambios.**

2. **¿Que tipo de merge crees que se hará? ¿Se puede hacer un fast forward, o por el contrario git creará un commit para el merge? ¿Crees que git podrá mezclar sin necesidad de requerir la ayuda manual del usuario?**

Como he explicado antes, al haberse modificado las dos ramas, git no puede hacer un fast forward porque se han modificado las mismas lineas en las dos versiones. Por lo que hay que hacer un commit.

3. **Mezcla en master la rama bugFix. Comprueba que ahora si se ha producido un commit nuevo en el que se han fusionado los cambios de ambas ramas. Añade al principio del comentario sugerido por git para el commit: C5:**

4. **Comprueba ahora con gitgraph y con git branch --merged los efectos de la mezcla.**

```
(base) root@kali:~/ISI/ejercicio-3# git commit -m "C5: Merge de bugFix"
[master be95a23] C5: Merge de bugFix
(base) root@kali:~/ISI/ejercicio-3# gitgraph
*   be95a23 (HEAD -> master) C5: Merge de bugFix
|\
| * 44e91cb (bugFix) Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git branch --merged
  bugFix
* master
```

5. **Comprueba el contenido de s1. Observa que ha incorporado el cambio de s1 de bugFix (cambiado Leche por Leche desnatada) y el de master (Fregona).**

```
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche desnatada
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
Fregona
```

Se ha comprometido s1 juntando las lineas de la rama master(Fregona) y bugFix(Leche desnatada)

6. **Comprueba las diferencias entre las ramas master y bugFix. ¿A que son debidas?**

```
(base) root@kali:~/ISI/ejercicio-3# git diff master..bugFix
diff --git a/s1 b/s1
index e51ea94..9384b0e 100644
--- a/s1
+++ b/s1
@@ -3,4 +3,3 @@ Tomate pelado
 Tomate ecológico
 Queso
 Yoghourt griego
-Fregona
```

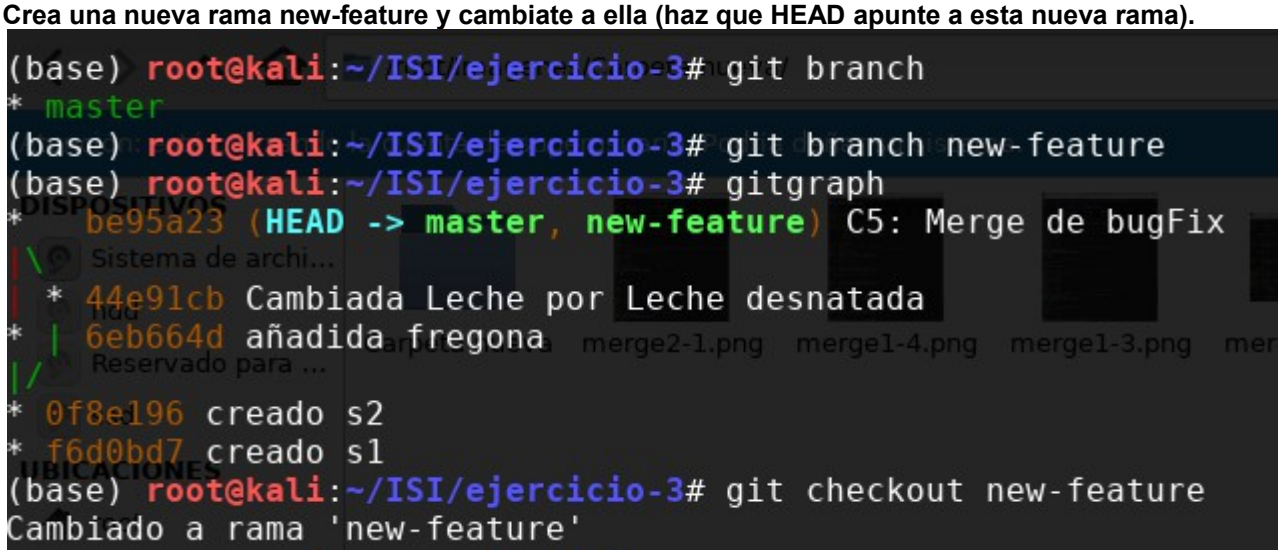
S1 en Master es una copia de bugFix pero incluye lineas que no estaban en bugFix (Fregona) por lo que reporta esa difetencia.

7. **Borra ahora la rama bugFix.**
8. **Observa el grafo de commits. La estructura creada debido a la ultima mezcla permanece, pero no las ramas borradas. La historia de commits de la rama master ha dejado de ser lineal, es un grafo. Hay usuarios de git a los que no les gusta esta estructura. Veremos mas adelante como mezclar ramas sin perder una estructura lineal de las ramas.**

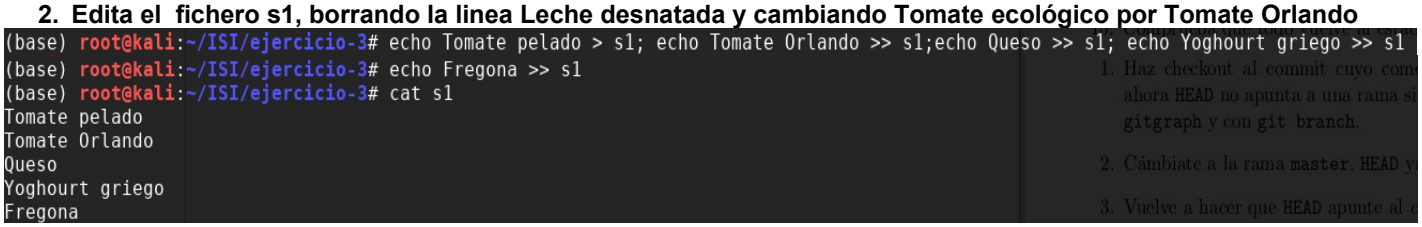
```
(base) root@kali:~/ISI/ejercicio-3# git branch -d bugFix
Eliminada la rama bugFix (era 44e91cb)..
(base) root@kali:~/ISI/ejercicio-3# gitgraph
*   be95a23 (HEAD -> master) C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git log
commit be95a2354a1d80aafbee58717ac1c6aba90bddc2 (HEAD -> master)
Merge: 6eb664d 44e91cb
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 12:34:37 2020 +0100

    C5: Merge de bugFix

commit 44e91cb4a0da8942c0d4a010cabbb0a5c9424bbd
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 12:06:22 2020 +0100

    Cambiada Leche por Leche desnatada

commit 6eb664df47576a24cab86f687c99fe6b2ddca4d9
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 12:04:42 2020 +0100

    añadida fregona

commit 0f8e196a0a7f35e8a139e6d845ee12c677c85f95
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 12:03:12 2020 +0100

    creado s2

commit f6d0bd7211c0d606cbe4bdf59cd552e355986016
Author: raul8251 <raulgilgill@gmail.com>
Date:   Wed Nov 18 12:02:45 2020 +0100

    creado s1
```

Merge 2

1. **Crea una nueva rama new-feature y cambiate a ella (haz que HEAD apunte a esta nueva rama).**



2. **Edita el fichero s1, borrando la linea Leche desnatada y cambiando Tomate ecológico por Tomate Orlando**



3. **Comprueba la diferencia entre el entorno de trabajo, el stage area y HEAD con variantes del comando git diff**

En stage area no hemos añadido nada, alli está leche desnatada y Tomate ecologico.
En el directorio de trabajo hemos eliminado leche desnatada y hemos modificado Tomate ecologico por Tomate Orlando, por lo que reporta esas diferencias

4. **Añade a stage area el fichero s1**
5. **Comprueba la diferencia entre el entorno de trabajo, el stage area y HEAD con variantes del comando git diff**

Ahora hemos añadido los cambios a stage area, pero como es lógico no lo hemos comprometido asi que nos reporta esas diferencias entre stage area y HEAD. Entre el directorio de trabajo y stage area no reporta diferencias porque están sincronizados con los mismos archivos.

5. **Crea un nuevo commit con el comentario:**

   C6: borrada Leche desnatada, cambiado Tomate ecologico por Tomate Orlando

```
(base) root@kali:~/ISI/ejercicio-3# git commit -m "borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando"
[new-feature 4144018] borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
 1 file changed, 1 insertion(+), 2 deletions(-)
```

6. **Cambiate a la rama master**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout master
Cambiado a rama 'master'
```

7. **Edita s1, cambiando Yoghourt griego por Yoghourt y eliminando Tomate ecologico.**

```
(base) root@kali:~/ISI/ejercicio-3# echo Leche desnatada > s1; echo Tomate pelado >> s1; echo Queso >> s1; echo Yoghourt >> s1; echo Fregona >> s1
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche desnatada
Tomate pelado
Queso
Yoghourt
Fregona
```

8. **Añade a stage area y compromete el cambio con el comentario:**

   C7: cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecologico

```
(base) root@kali:~/ISI/ejercicio-3# git commit -am "cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico"
[master a14ac57] cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
 1 file changed, 1 insertion(+), 2 deletions(-)
```

9. **Comprueba el estado del grafo de commits**

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* a14ac57 (HEAD -> master) cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
| * 4144018 (new-feature) borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
|/
* be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
```

10. **Vamos a mezclar en master la rama new-feature. Pero antes contesta: ¿Que tipo de merge crees que se hara? ¿Se puede hacer un fast forward, o por el contrario git crear un commit para el merge? ¿Crees que git podra mezclar sin necesidad de requerir la ayuda manual del usuario?**

11. **Ahora si , estando en master mezcla new-feature.**

```
(base) root@kali:~/ISI/ejercicio-3# git merge new-feature
Auto-fusionando s1
CONFLICTO (contenido): Conflicto de fusión en s1
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.
(base) root@kali:~/ISI/ejercicio-3# git status
En la rama master
Tienes rutas no fusionadas.
  (arregla los conflictos y corre "git commit"
  (usa "git merge --abort" para abortar la fusion)

Rutas no fusionadas:
  (usa "git add <archivo>..." para marcar una resolución)
        ambos modificados:     s1

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
```

12. **¿Que ha ocurrido? Comprueba los contenidos de s1, pero no modifiques nada.**

```
(base) root@kali:~/ISI/ejercicio-3# cat s1
Tomate pelado
<<<<<<< HEAD
=======
Tomate Orlando
>>>>>>> new-feature
Queso
Yoghourt
Fregona
```

13. **Tras hacerlo, deshaz los efectos de la mezcla de ramas usando el comando git merge --abort**
14. **Comprueba que todo vuelve al estado anterior a la mezcla. Comprueba que hay en s1.**

```
(base) root@kali:~/ISI/ejercicio-3# git merge --abort
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche desnatada
Tomate pelado
Queso
Yoghourt
Fregona
```

15. **Vuelve a intentar la mezcla de new-feature**

```
(base) root@kali:~/ISI/ejercicio-3# git merge new-feature
Auto-fusionando s1
CONFLICTO (contenido): Conflicto de fusión en s1
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.
(base) root@kali:~/ISI/ejercicio-3# git status
En la rama master
Tienes rutas no fusionadas.
  (arregla los conflictos y corre "git commit"
  (usa "git merge --abort" para abortar la fusion)

Rutas no fusionadas:
  (usa "git add <archivo>..." para marcar una resolución)
        ambos modificados:      s1

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
```

16. **Si ha habido conflicto que requiere accion manual, crea un commit para arreglarlo, de forma que en s1 se queden los cambios de new-feature y no los de master.**

    Cuando hayas dejado s1 en el estado deseado, añade s1 a stage area y escribe git commit, escribiendo un comentario que comience por C8: . F jate que este commit aparece como la mezcla de las ramas.

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* a14ac57 (HEAD -> master) cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
| * 4144018 (new-feature) borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# ^C
(base) root@kali:~/ISI/ejercicio-3# git checkout 4144018 s1
Actualizada 1 ruta para 3a14db7
(base) root@kali:~/ISI/ejercicio-3# cat s1
Tomate pelado
Tomate Orlando
Queso
Yoghourt griego
Fregona
(base) root@kali:~/ISI/ejercicio-3# git commit -am "C8"
[master a9e1ea4] C8
```

17. **Comprueba ahora el estado del grafo de commits**
18. **Comprueba si es seguro ahora borrar la rama new-feature**

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
*   a9e1ea4 (HEAD -> master) C8
|\
| * 4144018 (new-feature) borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
* | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git branch --merged
* master
  new-feature
```

19. **Borra la rama new-feature**
20. **Comprueba ahora el estado del grafo de commits**

```
(base) root@kali:~/ISI/ejercicio-3# git branch -d new-feature
Eliminada la rama new-feature (era 4144018)..
(base) root@kali:~/ISI/ejercicio-3# gitgraph
*   a9e1ea4 (HEAD -> master) C8
|\
| * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
* | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
```

HEAD desacoplado (detached)

1. **Haz checkout al commit cuyo comentario empieza por C2:... Como puedes ver, git nos avisa de que ahora HEAD no apunta a una rama sino a un commit. HEAD esta desacoplado (detached). Compruebalo con gitgraph y con git branch.**

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
*   a9e1ea4 (HEAD -> master) C8
|\
| * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
* | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git checkout ^C
(base) root@kali:~/ISI/ejercicio-3# git checkout 0f8e196
Nota: cambiando a '0f8e196'.

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.

Si quieres crear una nueva rama para mantener los commits que has creado,
puedes hacerlo (ahora o después) usando -c con el comando checkout. Ejemplo:

  git switch -c <nombre-de-nueva-rama>

O deshacer la operación con:

  git switch -

Desactiva este aviso poniendo la variable de config advice.detachedHead en false

HEAD está ahora en 0f8e196 creado s2
```

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
*   a9e1ea4 (master) C8
|\
| * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
* | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 (HEAD) creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git branch
* (HEAD desacoplada en 0f8e196)
  master
```

2. **Cambiate a la rama master. HEAD ya no esta detached. Observa con gitgraph.**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout master
La posición previa de HEAD era 0f8e196 creado s2
Cambiado a rama 'master'
(base) root@kali:~/ISI/ejercicio-3# gitgraph
*   a9e1ea4 (HEAD -> master) C8
|\
| * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
* | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git branch
* master
```

3. **Vuelve a hacer que HEAD apunte al commit cuyo comentario empieza por C2:...**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout 0f8e196
Nota: cambiando a '0f8e196'.

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.

Si quieres crear una nueva rama para mantener los commits que has creado,
puedes hacerlo (ahora o después) usando -c con el comando checkout. Ejemplo:

  git switch -c <nombre-de-nueva-rama>

O deshacer la operación con:

  git switch -

Desactiva este aviso poniendo la variable de config advice.detachedHead en false

HEAD está ahora en 0f8e196 creado s2
```

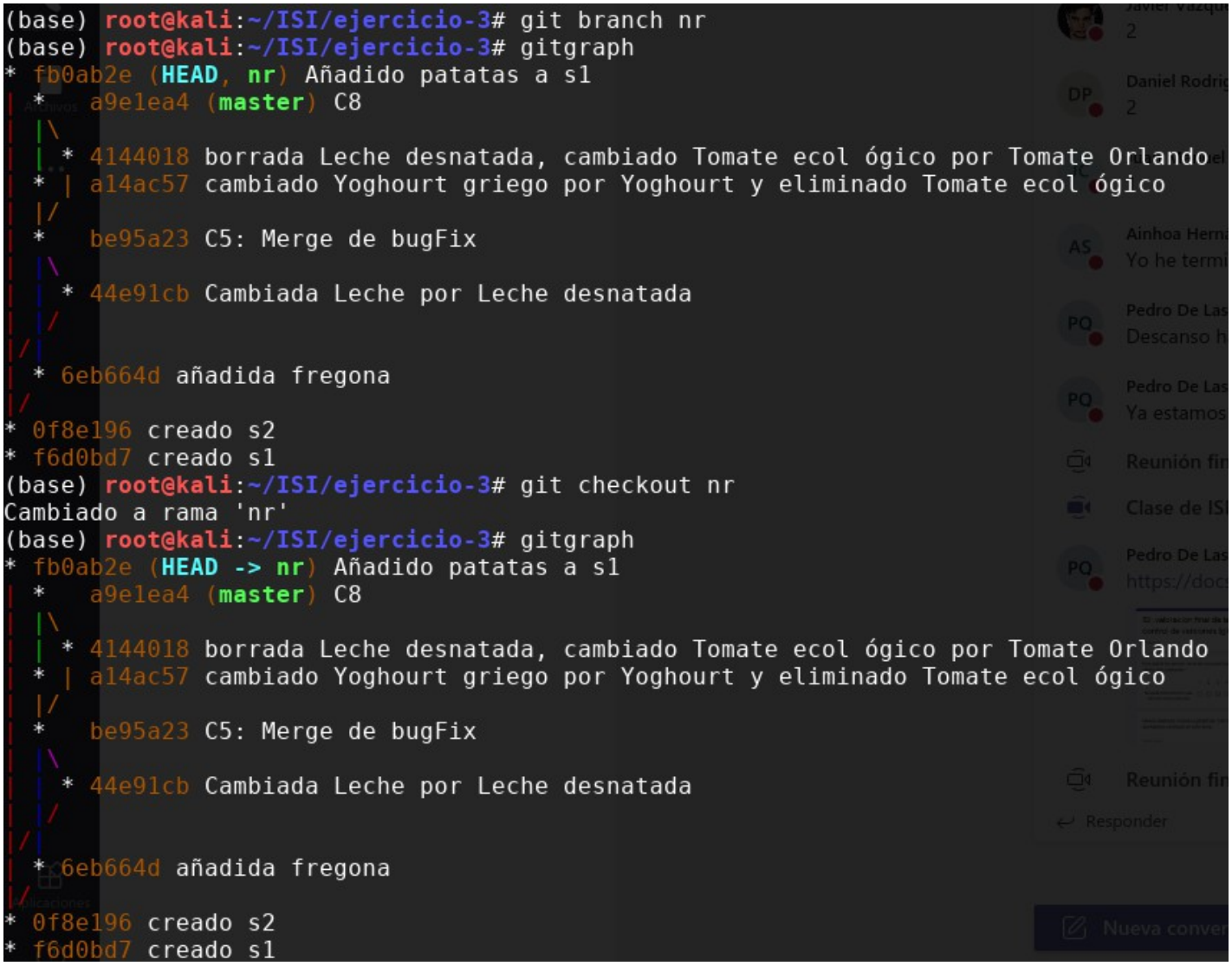
4. **Si creamos ahora un commit corremos el riesgo de que no haya forma de acceder a el al no estar en ninguna rama. Crea un commit con s1 modificado para que tenga al final una nueva linea con Patatas. Ponle como comentario C9: Añadido Patatas a s1.**

```
(base) root@kali:~/ISI/ejercicio-3# cat s1
Leche
Tomate pelado
Tomate ecológico
Queso
Yoghourt griego
(base) root@kali:~/ISI/ejercicio-3# echo Patatas >> s1
(base) root@kali:~/ISI/ejercicio-3# git commit -am "Añadido patatas a s1"
[HEAD desacoplado fb0ab2e] Añadido patatas a s1
 1 file changed, 1 insertion(+)
```

5. **Estamos en situacion de maximo peligro. Este commit no esta en ninguna rama. Compruebalo con gitgraph: solo HEAD apunta a este commit. Si ahora te cambiases a otra rama lo podriamos perder (git hace de vez en cuando recoleccion de basura de commits que no estan en ninguna rama). Por ello, NO TE CAMBIES a otra rama.**

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* fb0ab2e (HEAD) Añadido patatas a s1
|  *   a9e1ea4 (master) C8
|  |\
|  | * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
|  * | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|  |/
|  *   be95a23 C5: Merge de bugFix
|  |\
|  | * 44e91cb Cambiada Leche por Leche desnatada
|  |/
|  * 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
```

5. **Crea una nueva rama nr. Comprueba los efectos con gitgraph. Ahora el commit esta accesible a traves de la rama nr. Ha pasado el peligro. Pero HEAD aun no esta apuntando a la rama.**
6. **Cambiate a la rama nr. Comprueba los efectos con gitgraph. Ahora HEAD ya no esta desacoplado (detached): apunta a nr. Ahora si haces nuevos commits quedan en la rama nr.**



7. **Haz un nuevo commit en el que añadas a Manzanas al final de s1. Ponle como comentario: C10: añadidas Manzanas a s1**

git stash

1. **Comprueba el estado del repo con gitgraph, git status,...**

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* fe30372 (HEAD -> nr) Añadidas manzanas a s1
* fb0ab2e Añadido patatas a s1
|   a9e1ea4 (master) C8
|\
| * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
| | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
|   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
| /
|/
* 6eb664d añadida fregona
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git status
En la rama nr
nada para hacer commit, el árbol de trabajo está limpio
```

2. **Cambiate a la rama master**
3. **Edita s1 añadiendole al final Peras.**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout master
Cambiado a rama 'master'
(base) root@kali:~/ISI/ejercicio-3# cat s1
Tomate pelado
Tomate Orlando
Queso
Yoghourt griego
Fregona
(base) root@kali:~/ISI/ejercicio-3# echo Peras >> s1
```

4. **Cambiate a la rama nr. ¿Que ocurre? ¿Por que no puedes? Trata de entender por que git, para ayudarte, no te deja cambiar de rama en este caso.**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout nr
error: Los cambios locales de los siguientes archivos serán sobrescritos por checkout:
        s1
Por favor realiza un commit con los cambios o un stash antes de cambiar ramas.
Abortando
```

Como no hemos guardado los cambios nos advierte que podemos perder los cambios al cambiarnos de rama.

5. **Lo que ocurre es que en el área de trabajo esta la versión de s1 que acabas de modificar. Esta versión NO esta comprometida. Si ahora te cambiases a la rama nr, se copiar a la ultima versión comprometida de s1 que hay en nr al área de trabajo, por lo que perder as la versión actual que acabas de modificar.**

   Una solución ser a copiar el fichero s1 a otro con otro nombre, para no perder sus cambios, recuperar el estado de s1 comprometido, y entonces git s te dejar a cambiar de rama. Haciéndolo as se conservar a en el área de trabajo el nuevo fichero en el que has realizado la copia de s1. Pero no haremos eso sino que utilizaremos git stash para automatizar esta tarea.

6. **Guarda el estado del área de trabajo (s1 con Peras) de la rama master, en la que estas, ejecutando git stash. Comprueba que después de guardarlo, se recupera la ultima versión comprometida de s1, que no tiene Peras. La versión que tenias en el área de trabajo que incluía Peras ha quedado almacenada gracias a git stash en una nueva rama que se ha creado de nombre refs/stash. La veras con gitgraph pero la manejaras directamente a través de git stash.**

7. **Comprueba que ahora todo esta bien con git status**

```
(base) root@kali:~/ISI/ejercicio-3# git stash
Directorio de trabajo guardado y estado de índice WIP on master: a9e1ea4 C8
(base) root@kali:~/ISI/ejercicio-3# cat s1
Tomate pelado
Tomate Orlando
Queso
Yoghourt griego
Fregona
(base) root@kali:~/ISI/ejercicio-3# gitgraph
*   e1ccf8a (refs/stash) WIP on master: a9e1ea4 C8
|\
| * 947cea4 index on master: a9e1ea4 C8
|/
*   a9e1ea4 (HEAD -> master) C8
|\
| * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
* | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
| * fe30372 (nr) Añadidas manzanas a s1
| * fb0ab2e Añadido patatas a s1
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
```

8. **Prueba ahora a cambiarte a nr. Ahora si podrás.**
9. **Vuelve a la master**

```
(base) root@kali:~/ISI/ejercicio-3# git checkout nr
Cambiado a rama 'nr'
(base) root@kali:~/ISI/ejercicio-3# git checkout master
Cambiado a rama 'master'
```

10. **Recuperemos ahora los cambios que guardamos con git stash para que reaparezcan en el área de trabajo. Para ello ejecuta git stash apply. Comprueba que vuelves a tener en el área de trabajo la versión de s1 que contiene Peras.**

```
(base) root@kali:~/ISI/ejercicio-3# git stash apply
En la rama master
Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
        modificado:     s1

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
(base) root@kali:~/ISI/ejercicio-3# cat s1
Tomate pelado
Tomate Orlando
Queso
Yoghourt griego
Fregona
Peras
```

11. **Crea un commit con este cambio de s1 con el comentario: C11: añadidas Peras a s1**

```
(base) root@kali:~/ISI/ejercicio-3# git commit -am "añadidas Peras a s1"
[master 94c3c07] añadidas Peras a s1
 1 file changed, 1 insertion(+)
```

12. **Lo que se añade a stash no se borra. Con gitgraph veras que sigue estando ahi la rama refs/stash. Puedes ver todo lo que has ido guardando en stash con el comando git stash list.**

```
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* 94c3c07 (HEAD -> master) añadidas Peras a s1
| * e1ccf8a (refs/stash) WIP on master: a9e1ea4 C8
|/|
| * 947cea4 index on master: a9e1ea4 C8
|/
*   a9e1ea4 C8
|\
| * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
* | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
* | 6eb664d añadida fregona
|/
| * fe30372 (nr) Añadidas manzanas a s1
| * fb0ab2e Añadido patatas a s1
|/
* 0f8e196 creado s2
* f6d0bd7 creado s1
(base) root@kali:~/ISI/ejercicio-3# git stash list
stash@{0}: WIP on master: a9e1ea4 C8
```

13. **Si modificas de nuevo s1 puedes volver a añadir una entrada con git stash, actuando como una pila: git stash apply recupera lo ultimo guardado con git stash. Haz algún cambio a s1 y guardalo de nuevo en stash.**

```
(base) root@kali:~/ISI/ejercicio-3# echo Platanos >> s1
(base) root@kali:~/ISI/ejercicio-3# git stash
Directorio de trabajo guardado y estado de índice WIP on master: 94c3c07 añadidas Peras a s1
```

14. **Ahora tendras dos entradas que puedes mostrar con git stash list**

```
(base) root@kali:~/ISI/ejercicio-3# git stash list
stash@{0}: WIP on master: 94c3c07 añadidas Peras a s1
stash@{1}: WIP on master: a9e1ea4 C8
```

15. **Puedes eliminar todas las entradas de stash con (git stash clear), aplicar alguna de las versiones concreta con git stash apply stash@{NUMERO}, ... Ejecuta git help stash para saber mas.**

16. **Para terminar, borra con git stash clear. Comprueba con git stash list y gitgraph que ya no hay nada en stash.**

```
(base) root@kali:~/ISI/ejercicio-3# git stash clear
(base) root@kali:~/ISI/ejercicio-3# gitgraph
* 94c3c07 (HEAD -> master) añadidas Peras a s1
*   a9e1ea4 C8
|\
| * 4144018 borrada Leche desnatada, cambiado Tomate ecol ógico por Tomate Orlando
* | a14ac57 cambiado Yoghourt griego por Yoghourt y eliminado Tomate ecol ógico
|/
*   be95a23 C5: Merge de bugFix
|\
| * 44e91cb Cambiada Leche por Leche desnatada
```

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.**