25_conflictos_y_correcciones

May 6, 2020

1 Conflictos y correcciones.

1.1 Preliminares:

Para este capítulo se utilizará una versión creada previamente del directorio `demo` que incluye los ejercicios de los capítulos previos.

[1]:
```
rm -rf prueba
```

[2]:
```
unzip src/prueba.zip
```

```
Archive:  src/prueba.zip
   creating: prueba/
 extracting: prueba/archivo-2
 extracting: prueba/archivo-1
   creating: prueba/.git/
  inflating: prueba/.git/index
   creating: prueba/.git/refs/
   creating: prueba/.git/refs/heads/
  inflating: prueba/.git/refs/heads/master
 extracting: prueba/.git/refs/heads/restituida
 extracting: prueba/.git/refs/heads/nueva
   creating: prueba/.git/refs/tags/
  inflating: prueba/.git/description
   creating: prueba/.git/info/
  inflating: prueba/.git/info/exclude
 extracting: prueba/.git/ORIG_HEAD
 extracting: prueba/.git/COMMIT_EDITMSG
   creating: prueba/.git/objects/
   creating: prueba/.git/objects/90/
 extracting: prueba/.git/objects/90/a8b80bbb845cb1187852777769937a9574e3ab
   creating: prueba/.git/objects/ed/
 extracting: prueba/.git/objects/ed/7d117fc8970b297653c23ebc41936cc4ab4472
   creating: prueba/.git/objects/pack/
   creating: prueba/.git/objects/cb/
 extracting: prueba/.git/objects/cb/0b6bb3a84bc6fcd0f6a0f87909bf68f298e21a
   creating: prueba/.git/objects/2c/
 extracting: prueba/.git/objects/2c/b76ef80c41c0037c6a82092737cba6061c7083
   creating: prueba/.git/objects/info/
```

```
  creating: prueba/.git/objects/e6/
extracting: prueba/.git/objects/e6/9de29bb2d1d6434b8b29ae775ad8c2e48c5391
  creating: prueba/.git/objects/a1/
extracting: prueba/.git/objects/a1/9abfea0f29b668c91c58c834b8965e6c37804f
  creating: prueba/.git/objects/dc/
extracting: prueba/.git/objects/dc/d3ac8c60d63186964b45f87cc4416e6caefa5f
  creating: prueba/.git/objects/78/
extracting: prueba/.git/objects/78/8564428c6076630b22efc89ad20c5f63ee9bf1
  creating: prueba/.git/objects/ba/
extracting: prueba/.git/objects/ba/397e8f7c51f8dfdefa12250933811042df91a2
  creating: prueba/.git/objects/ff/
extracting: prueba/.git/objects/ff/7ede1f3ce0201ae532684bbee157247c3a88b7
  creating: prueba/.git/objects/ac/
extracting: prueba/.git/objects/ac/9103394fb02ee49adb4c05e2125174707fbb7d
  creating: prueba/.git/objects/1f/
extracting: prueba/.git/objects/1f/1f5adbb5e6ff2e2ff651e352f21df0417defb2
  creating: prueba/.git/objects/08/
extracting: prueba/.git/objects/08/34a0b252e02bc28bae34588e01f410ad771884
  creating: prueba/.git/objects/d7/
extracting: prueba/.git/objects/d7/f0e7ba5cf079cdea1d7c8792e1432161f082c1
  creating: prueba/.git/objects/46/
extracting: prueba/.git/objects/46/c8014e29a665bd0e883c9008ad4412f8933809
  creating: prueba/.git/objects/df/
extracting: prueba/.git/objects/df/8a330f084618bdb823d99a6997d7be4b9efd15
  creating: prueba/.git/objects/b3/
extracting: prueba/.git/objects/b3/890082fec489a5ecff76db4fd36cc10d4bc8d6
  creating: prueba/.git/branches/
  creating: prueba/.git/logs/
  creating: prueba/.git/logs/refs/
  creating: prueba/.git/logs/refs/heads/
 inflating: prueba/.git/logs/refs/heads/master
 inflating: prueba/.git/logs/refs/heads/restituida
 inflating: prueba/.git/logs/refs/heads/nueva
 inflating: prueba/.git/logs/HEAD
extracting: prueba/.git/HEAD
 inflating: prueba/.git/config
  creating: prueba/.git/hooks/
 inflating: prueba/.git/hooks/post-update.sample
 inflating: prueba/.git/hooks/commit-msg.sample
 inflating: prueba/.git/hooks/pre-receive.sample
 inflating: prueba/.git/hooks/update.sample
 inflating: prueba/.git/hooks/pre-push.sample
 inflating: prueba/.git/hooks/fsmonitor-watchman.sample
 inflating: prueba/.git/hooks/prepare-commit-msg.sample
 inflating: prueba/.git/hooks/pre-applypatch.sample
 inflating: prueba/.git/hooks/pre-rebase.sample
 inflating: prueba/.git/hooks/applypatch-msg.sample
 inflating: prueba/.git/hooks/pre-commit.sample
```

```
 extracting: prueba/archivo_nuevo
 extracting: prueba/invisible
 extracting: prueba/.gitignore
```

[3]:
```
cd prueba
```

[4]:
```
git branch
```

```
* master
  nueva
  restituida
```

[5]:
```
git log --oneline
```

```
ff7ede1 (HEAD -> master) commit
fusionado
0834a0b quinto commit
46c8014 (nueva) primer commit de la rama nueva
ba397e8 (restituida) cuarto commit
cb0b6bb segundo commit
ed7d117 primer commit
```

1.2 El comando `git clean`.

El comando `git clean` permite eliminar del directorio de trabajo a aquellos archivos que no están en estado de seguimiento.

`git clean <opciones> <ruta>`

Donde:

- `<opciones>` son las opciones aplicables.
- `<ruta>` es la ruta o patrón de los archivos a afectar con este comando.

La documentación de referencia del comando `git clean` está disponible en:

https://git-scm.com/docs/git-clean

En caso de ejecutar el comando `git clean` sin opciones, se producirá un error.

1.2.1 La opción `-f`.

La opción `-f` le indica al comando `git clean`que ejecute la acción forzosamente. En realidad, se utuliza como un "seguro", ya que en caso de uno ingresar dicha opción, se produciría un error.

Ejemplo:

- La siguiente celda modificará al archivo `archivo-5`.

[6]:
```
echo "Texto ilustrativo" > archivo-5
```

- La siguiente celda creará al archivo `indeseable`.

[7]:
```
touch indeseable
```

- La siguiente celda añadirá a `archivo-5` al área de preparación.

[8]:
```
git add archivo-5
```

[9]:
```
git status
```

```
En la rama master
Cambios a ser confirmados:
  (usa "git reset HEAD <archivo>…" para sacar del área de stage)

        nuevo archivo:  archivo-5

Archivos sin seguimiento:
  (usa "git add <archivo>…" para incluirlo a lo que se será confirmado)

        indeseable
```

- Al ejecutar la siguiente celda se generará un mensaje de error debido a que no es posible usar el comando `git clean` sin opciones.

[10]:
```
git clean
```

```
fatal: clean.requireForce default en true y ninguno -i, -n, ni -f entregados;
rehusando el clean
```

- La siguiente celda eliminará a los archivos que no estén en el área de seguimiento ni en el índice.

[11]:
```
git clean -f
```

```
Borrando indeseable
```

[12]:
```
ls
```

```
archivo-1  archivo-2  archivo-5  archivo_nuevo  invisible
```

[13]:
```
git status
```

```
En la rama master
Cambios a ser confirmados:
  (usa "git reset HEAD <archivo>…" para sacar del área de stage)

        nuevo archivo:  archivo-5
```

- Se realizará un commit.

[14]:
```
git commit -m  "septimo commit"
```

```
[master e206da6] septimo commit
 1 file changed, 1 insertion(+)
 create mode 100644 archivo-5
```

[15]:
```
git log --oneline
```

```
e206da6 (HEAD -> master) septimo
commit
ff7ede1 commit fusionado
0834a0b quinto commit
46c8014 (nueva) primer commit de la rama nueva
ba397e8 (restituida) cuarto commit
cb0b6bb segundo commit
ed7d117 primer commit
```

[16]:
```
ls
```

```
archivo-1  archivo-2  archivo-5  archivo_nuevo  invisible
```

1.3 El comando `git revert`.

El comando `git revert` permite regresar al estado de un commit previo, realizando las operaciones en sentido inverso que lleven al estado del commit en cuestión.

Al ejecutar este comando se realiza un commit nuevo.

`git revert <referencia>`

La documentación de referencia del comando `git revert` está disponible en:

https://git-scm.com/docs/git-revert

Ejemplo:

- La siguiente línea desplegaráel contenido de `archivo-1` el cual es:

```
Hola
Otra linea
```

[17]:
```
cat archivo-1
```

```
Hola
Otra linea
```

- La siguiente celda mostrará las diferencias entre `HEAD~2` y `HEAD`.

[18]:
```
git diff HEAD~2 HEAD
```

```
diff --git a/archivo-5 b/archivo-5
new file mode 100644
index 0000000..7d8a316
```

```
--- /dev/null
+++ b/archivo-5
@@ -0,0 +1 @@
+Texto ilustrativo
diff --git a/archivo_nuevo b/archivo_nuevo
new file mode 100644
index 0000000..df8a330
--- /dev/null
+++ b/archivo_nuevo
@@ -0,0 +1 @@
+Archivo de la rama nueva.
```

- La siguiente celda utilizará al comando `git revert` para restaurar el repositorio al estado de `HEAD~2`.
- La opción `--no-edit` evita que se abra un editor y se asigna de forma automática el comentario `Revert "quinto commit"` al commit resultante.

[19]:
```
git revert HEAD~2 --no-edit
```

```
[master 3cd265e] Revert "quinto commit"
 Date: Wed May 6 13:02:32 2020 -0500
 1 file changed, 1 deletion(-)
```

[20]:
```
ls
```

```
archivo-1  archivo-2  archivo-5  archivo_nuevo  invisible
```

[21]:
```
git log --oneline
```

```
3cd265e (HEAD -> master) Revert
"quinto commit"
e206da6 septimo commit
ff7ede1 commit fusionado
0834a0b quinto commit
46c8014 (nueva) primer commit de la rama nueva
ba397e8 (restituida) cuarto commit
cb0b6bb segundo commit
ed7d117 primer commit
```

- La siguiente celda desplegará el contenido actual de `archivo-1` el cual es:

```
Hola
```

[22]:
```
cat archivo-1
```

```
Hola
```

- La siguiente celda desplegará las diferencias entre `HEAD~1`y `HEAD`.

[23]:
```
git diff HEAD~1 HEAD
```

```
diff --git a/archivo-1 b/archivo-1
index d7f0e7b..a19abfe 100644
--- a/archivo-1
+++ b/archivo-1
@@ -1,2 +1 @@
 Hola
-Otra linea
```

- La siguiente celda desplegará las diferencias entre `HEAD`y `HEAD~3`.

[24]:
```
git diff HEAD HEAD~3
```

```
diff --git a/archivo-1 b/archivo-1
index a19abfe..d7f0e7b 100644
--- a/archivo-1
+++ b/archivo-1
@@ -1 +1,2 @@
 Hola
+Otra linea
diff --git a/archivo-5 b/archivo-5
deleted file mode 100644
index 7d8a316..0000000
--- a/archivo-5
+++ /dev/null
@@ -1 +0,0 @@
-Texto ilustrativo
diff --git a/archivo_nuevo b/archivo_nuevo
deleted file mode 100644
index df8a330..0000000
--- a/archivo_nuevo
+++ /dev/null
@@ -1 +0,0 @@
-Archivo de la rama nueva.
```

1.4 El comando `git reset`.

Este comando permite regresar `HEAD` a un estado previo, eliminando ciertos elementos de los commits intermedios.

‘`git reset <referencia> <opción>`

La documentación de referencia del comando `git reset` está disponible en:

https://git-scm.com/docs/git-reset

1.4.1 La opción `--mixed`.

Esta opción regresa a `HEAD` y al índice al estado indicado, pero no a los archivos del directorio de trabajo. Esta es la opción por defecto.

1.4.2 La opción `--hard`.

Esta opción regresa a `HEAD`, al índice y al directorio de trabajo al estado indicado.

1.4.3 La opción `--soft`

Esta opción regresa a `HEAD` al estado indicado, pero no a los archivos del directorio de trabajo ni al índice.

Ejemplo:

- La siguiente celda desplegará el estado actual del directorio de trabajo.

[25]:
```
ls
```

```
archivo-1  archivo-2  archivo-5  archivo_nuevo  invisible
```

- La siguiente celda desplegará el contenido de `archivo-1`, el cual es:

```
Hola
```

[26]:
```
cat archivo-1
```

```
Hola
```

- La siguiente celda muesta las diferencias entre `HEAD~2` y `HEAD`.

[27]:
```
git diff HEAD~2 HEAD
```

```
diff --git a/archivo-1 b/archivo-1
index d7f0e7b..a19abfe 100644
--- a/archivo-1
+++ b/archivo-1
@@ -1,2 +1 @@
 Hola
-Otra linea
diff --git a/archivo-5 b/archivo-5
new file mode 100644
index 0000000..7d8a316
--- /dev/null
+++ b/archivo-5
@@ -0,0 +1 @@
+Texto ilustrativo
```

- La siguiente celda regresará al índice y al directorio de trabajo al estado de `HEAD~2` la historia de los commits posteriores a este estado será eliminada.

[28]:
```
git reset HEAD~2 --hard
```

```
HEAD está ahora en ff7ede1 commit fusionado
```

[29]:
```
git status
```

```
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
```

[30]:
```
git log --oneline
```

```
ff7ede1 (HEAD -> master) commit
fusionado
0834a0b quinto commit
46c8014 (nueva) primer commit de la rama nueva
ba397e8 (restituida) cuarto commit
cb0b6bb segundo commit
ed7d117 primer commit
```

[31]:
```
ls
```

```
archivo-1  archivo-2  archivo_nuevo  invisible
```

[32]:
```
cat archivo-1
```

```
Hola
Otra linea
```

- La siguiente celda creará al archivo `archivo-6`.

[33]:
```
touch archivo-6
```

- Las siguientes celdas crearán un commit con las últimas modificaciones al directorio de trabajo.

[34]:
```
git add --all
```

[35]:
```
git commit -a -m "commit posterior a un hard reset"
```

```
[master 82cd602] commit posterior a un hard reset
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 archivo-6
```

[36]:
```
git log --oneline
```

```
82cd602 (HEAD -> master) commit
posterior a un hard reset
ff7ede1 commit fusionado
0834a0b quinto commit
46c8014 (nueva) primer commit de la rama nueva
ba397e8 (restituida) cuarto commit
cb0b6bb segundo commit
ed7d117 primer commit
```

- La siguiente celda mostrará las diferencias entre `HEAD~3` y `HEAD`.

[37]:
```
git diff HEAD~3 HEAD
```

```
diff --git a/archivo-1 b/archivo-1
index a19abfe..d7f0e7b 100644
--- a/archivo-1
+++ b/archivo-1
@@ -1 +1,2 @@
 Hola
+Otra linea
diff --git a/archivo-6 b/archivo-6
new file mode 100644
index 0000000..e69de29
diff --git a/archivo_nuevo b/archivo_nuevo
new file mode 100644
index 0000000..df8a330
--- /dev/null
+++ b/archivo_nuevo
@@ -0,0 +1 @@
+Archivo de la rama nueva.
```

- La siguiente celda traerá al índice del repositorio a `HEAD~3`, pero dejará intacto al directorio de trabajo.

[38]:
```
git reset HEAD~3 --mixed
```

```
Cambios fuera del área de stage tras el reset:
M       archivo-1
```

- Esto permite regresar el índice al estado previo, pero da la oportunidad de añadir las últimas modificaciones al directorio de trabajo en un nuevo commit.

[39]:
```
git status
```

```
En la rama master
Cambios no rastreados para el commit:
  (usa "git add <archivo>…" para actualizar lo que será confirmado)
  (usa "git checkout -- <archivo>…" para descartar los cambios en el
directorio de trabajo)

        modificado:     archivo-1

Archivos sin seguimiento:
  (usa "git add <archivo>…" para incluirlo a lo que se será confirmado)

        archivo-6
        archivo_nuevo

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

[40]:
```
git log
```

```
commit ba397e8f7c51f8dfdefa12250933811042df91a2 (HEAD ->
master, restituida)
Author: Jose Luis Chiquete <josech@gmail.com>
Date:   Tue Dec 17 12:39:12 2019 -0600

    cuarto commit

commit cb0b6bb3a84bc6fcd0f6a0f87909bf68f298e21a
Author: Jose Luis Chiquete <josech@gmail.com>
Date:   Tue Dec 17 12:38:22 2019 -0600

    segundo commit

commit ed7d117fc8970b297653c23ebc41936cc4ab4472
Author: Jose Luis Chiquete <josech@gmail.com>
Date:   Tue Dec 17 12:38:17 2019 -0600

    primer commit
```

- Las siguientes celdas crearán un commit con las últimas modificaciones al directorio de trabajo.

[41]:
```
git add --all
```

[42]:
```
git commit -m "commit posterior a un mixed reset"
```

```
[master 6a7a74b] commit posterior a un mixed reset
 3 files changed, 2 insertions(+)
 create mode 100644 archivo-6
 create mode 100644 archivo_nuevo
```

[43]:
```
git log --oneline
```

```
6a7a74b (HEAD -> master) commit
posterior a un mixed reset
ba397e8 (restituida) cuarto commit
cb0b6bb segundo commit
ed7d117 primer commit
```

- La siguiente celda traerá al repositorio a `HEAD~3`, pero dejará intacto al directorio de trabajo y al índice.

[44]:
```
git reset HEAD~3 --soft
```

[45]:
```
ls
```

```
archivo-1  archivo-2  archivo-6  archivo_nuevo  invisible
```

- Se puede apreciar que los objetos del índice siguen en seguimiento.

[46]:
```
git status
```

```
En la rama master
Cambios a ser confirmados:
  (usa "git reset HEAD <archivo>…" para sacar del área de stage)

        modificado:     archivo-1
        renombrado:     archivo-3 -> archivo-6
        nuevo archivo:  archivo_nuevo
```

[47]:
```
git log
```

```
commit ed7d117fc8970b297653c23ebc41936cc4ab4472 (HEAD ->
master)
Author: Jose Luis Chiquete <josech@gmail.com>
Date:   Tue Dec 17 12:38:17 2019 -0600

    primer commit
```

- Las siguientes celdas crearán un commit con las últimas modificaciones al directorio de trabajo.

[48]:
```
git add --all
```

[49]:
```
git commit -m "commit posterior a un soft reset"
```

```
[master 4613cc2] commit posterior a un soft reset
 3 files changed, 3 insertions(+)
 rename archivo-3 => archivo-6 (100%)
 create mode 100644 archivo_nuevo
```

[50]:
```
git log --oneline
```

```
4613cc2 (HEAD -> master) commit
posterior a un soft reset
ed7d117 primer commit
```

1.5 El comando `git rebase`.

Este comando permite recombinar dos ramas a partir del estado en que fueron separadas.

`git rebase <rama 1> <rama 2>`

Ejemplo:

- La siguiente celda creará la rama `rama_alterna` y moverá el repositorio a dicha rama.

[51]:
```
git checkout -b rama_alterna
```

```
Cambiado a nueva rama 'rama_alterna'
```

- Se crearán los archivos `archivo-a1`y `archivo-a2`.

[52]:
```
touch archivo-a1 archivo-a2
```

- Las siguientes celdas crearán un commit con las últimas modificaciones al directorio de trabajo.

[53]:
```
git add --all
```

[54]:
```
git commit -m "primer commit rama alterna"
```

```
[rama_alterna d34a84c] primer commit rama alterna
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 archivo-a1
 create mode 100644 archivo-a2
```

[55]:
```
git log
```

```
commit d34a84c3951b140b289550769b4761a6829edd26 (HEAD ->
rama_alterna)
Author: Jose Luis Chiquete <josech@gmail.com>
Date:   Wed May 6 13:03:03 2020 -0500

    primer commit rama alterna

commit 4613cc2ddfd68c8b28189eb998e188603f1c3a25
(master)
Author: Jose Luis Chiquete <josech@gmail.com>
Date:   Wed May 6 13:02:51 2020 -0500

    commit posterior a un soft reset

commit ed7d117fc8970b297653c23ebc41936cc4ab4472
Author: Jose Luis Chiquete <josech@gmail.com>
Date:   Tue Dec 17 12:38:17 2019 -0600

    primer commit
```

- La siguiente celda moverá al repositorio a la rama `master`.

[56]:
```
git checkout master
```

```
Cambiado a rama 'master'
```

[57]:
```
ls
```

```
archivo-1  archivo-2  archivo-6  archivo_nuevo  invisible
```

- La siguiente celda creará a `archivo-6` y eliminará a `archivo-2` en el directorio de trabajo.

[58]:
```
touch archivo-6
rm archivo-2
```

- Las siguientes celdas crearán un commit con las últimas modificaciones al directorio de trabajo.

[59]:
```
git add --all
```

[60]:
```
git commit -m "commit de rebase"
```

```
[master 8491192] commit de rebase
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 archivo-2
```

[61]:
```
git log --oneline
```

```
8491192 (HEAD -> master) commit de
rebase
4613cc2 commit posterior a un soft reset
ed7d117 primer commit
```

- La siguiente celda moverá el repositorio a la rama `rama_alterna`.

[62]:
```
git checkout rama_alterna
```

```
Cambiado a rama 'rama_alterna'
```

- La siguiente celda mostrará las diferencias entre las ramas `rama_alternay` `master`.

[63]:
```
git diff rama_alterna master
```

```
diff --git a/archivo-2 b/archivo-2
deleted file mode 100644
index e69de29..0000000
diff --git a/archivo-a1 b/archivo-a1
deleted file mode 100644
index e69de29..0000000
diff --git a/archivo-a2 b/archivo-a2
deleted file mode 100644
index e69de29..0000000
```

- La siguiente celda reconstruirá las acciones de `master` sobre `rama_alterna` a partir de que se creó la rama.

[64]:
```
git rebase master
```

```
En primer lugar, rebobinando HEAD para después reproducir tus cambios encima de
ésta…
Aplicando: primer commit rama alterna
```

[65]:
```
git branch
```

```
  master
  nueva
* rama_alterna
  restituida
```

[66]:
```
git log --oneline
```

```
44e7b81 (HEAD -> rama_alterna)
primer commit rama alterna
8491192 (master) commit de rebase
4613cc2 commit posterior a un soft reset
ed7d117 primer commit
```

1.6 Uso de `git checkout` con archivos.

En ciertas ocasiones es necesario traer archivos de versiones previas del repositorio. La manera de hacer esto es mediante:

`git checkout <referencia> <archivo>`

Ejemplo:

- La siguiente celda eliminará a `archivo-6` del directorio de trabajo.

[67]:
```
rm  archivo-6
```

- Las siguientes celdas crearán un commit con las últimas modificaciones al directorio de trabajo.

[68]:
```
git add --all
```

[69]:
```
git commit -m "eliminacion en rama alterna"
```

```
[rama_alterna 95bc1bf] eliminacion en rama alterna
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 archivo-6
```

[70]:
```
ls
```

```
archivo-1  archivo-a1  archivo-a2  archivo_nuevo  invisible
```

- La siguiente celda traerá a `archivo-6` desde `HEAD~`.

[71]:
```
git checkout HEAD~ archivo-6
```

[72]:
```
ls
```

```
archivo-1  archivo-6  archivo-a1  archivo-a2  archivo_nuevo  invisible
```

[73]:
```
git status
```

```
En la rama rama_alterna
Cambios a ser confirmados:
  (usa "git reset HEAD <archivo>…" para sacar del área de stage)

        nuevo archivo:  archivo-6
```