Desplegament d'aplicacions web

TEMA 04

Control de Versions

Git – Part III

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

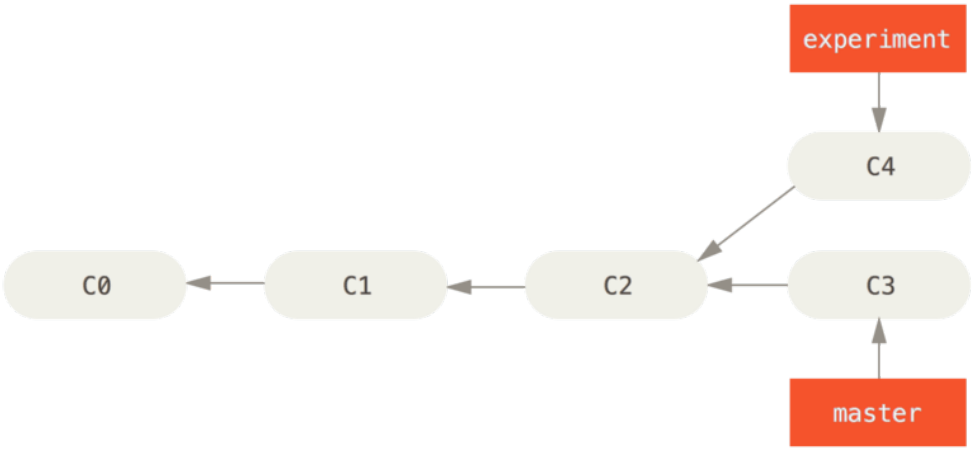
IES DR. LLUÍS SIMARRO

Git

Rebase

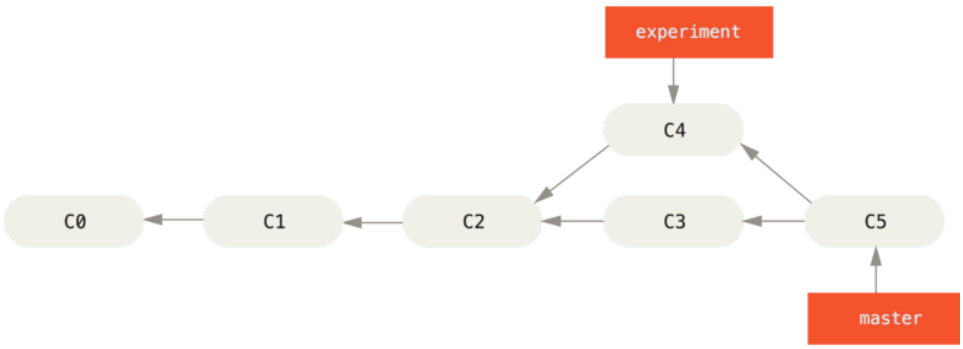
Una altra manera d'**integrar canvis fets en diferents branques** és l'opció **`rebase`**.

Per **comparar el seu funcionament amb el `merge`**, veiem un exemple en el que s'ha creat una branca i s'han fet **`commits`** en les dues després de la separació:



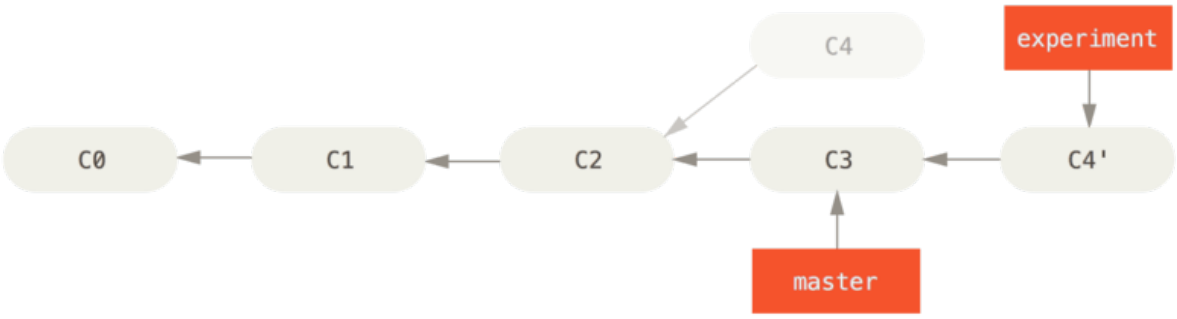
La forma més fàcil d'integrar les branques, ja que ja hem vist, és l'ordre **`merge`**. Porta a terme una fusió a tres bandes, entre els dos últims **`commits`** en cada branca (C3 i C4) i el més recent ancestre comú de les dues (C2), creant un nou **`commit`** (C5):

```
git checkout master
git merge experiment
```



L'altra manera més comuna d'unir les branques és un **rebase**. En el nostre exemple el que es fa és prendre el **commit** que es va introduir en C4 i tornar a aplicar-lo a C3. Fixem-nos en el fet que en aquest exemple, per executar l'ordre **rebase**, hem d'estar en la branca que volem fusionar, a diferència del **merge**, on havíem d'estar en la branca a la que volíem fusionar:

```
git checkout experiment
git rebase master
```



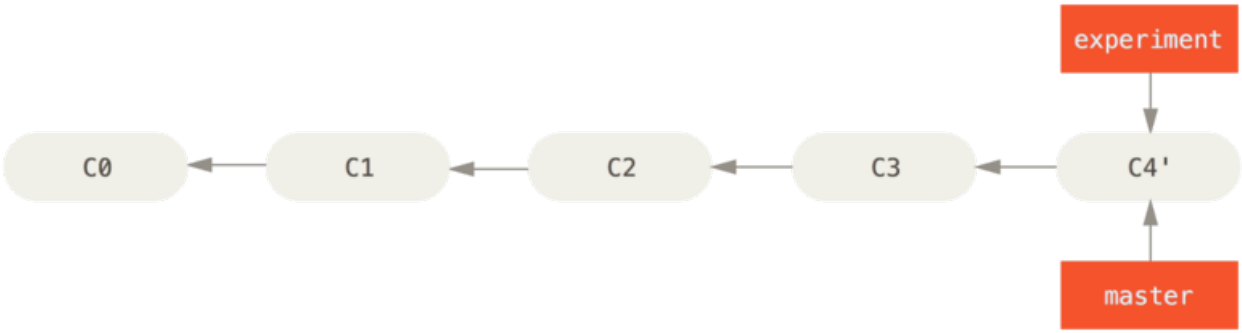
En general, amb l'ordre **rebase**, es prenen tots els **commits** que s'han fet en una branca i es tornen a reproduir en una altra.

En aquest exemple, l'acció consisteix a anar a l'ancestre comú de les dues branques (la que estem i aquella a la que volem fer el **rebase**), obtenint la diferència introduïda per cada **commit** de la branca en la que estem, es guarden aquestes

diferencies en arxius temporals, es reinicia la branca actual al **commit** de la branca a la qual es vol fer el **rebase**, i finalment es crea un nou i únic **commit** (perquè en la branca només n'hem fet un) en el que s'apliquen de manera incremental les diferències trobades anteriorment.

En aquest punt, es pot tornar a la branca principal (**master**) i fer un **merge**:

```
git checkout master
git merge experiment
```



Ara, el resultat en **C4' és exactament el mateix que l'obtingut en C5** en l'exemple del **merge**. **No hi ha diferència en el producte final de la integració**, però **rebase** deixa un historial més net. Si s'examina el log d'una branca on s'ha utilitzat **rebase**, es veu com una història lineal: sembla que tota la feina ha ocorregut en sèrie, encara que va succeir originalment en paral·lel.

Es pot fer açò per assegurar-se que els **commits** s'apliquen netament en una branca remota, per exemple en un projecte al qual estem tractant de contribuir, però que no mantenim. En aquest cas, faríem el nostre treball en una branca i després faríem el **rebase** del treball en **origin/master** quan considerem adequat pujar els canvis al projecte principal. D'aquesta manera, el mantenidor no ha de fer cap tasca complexa d'integració.

rebase torna a aplicar els canvis d'una línia de treball sobre una altra en l'ordre en què es van introduir, mentre que **merge** agafa els punts finals i els fusiona.

Altre exemple

En realitat, **no fa falta estar necessàriament en la branca a la que volem fer el `rebase`**, en la mateixa instrucció podem indicar-ho amb un altre paràmetre. Per exemple, si tenim aquesta situació, en la que **la branca actual és experiment**:

```
      A---B---C *experiment
     /
D---E---F---G master
```

El resultat de les següents 2 ordres seria el mateix:

(amb branca experiment activa): `git rebase master`

(amb branca master activa): `git rebase master experiment`

```
                A'--B'--C' *experiment
               /
D---E---F---G master
```

L'última forma és simplement una abreviatura de:

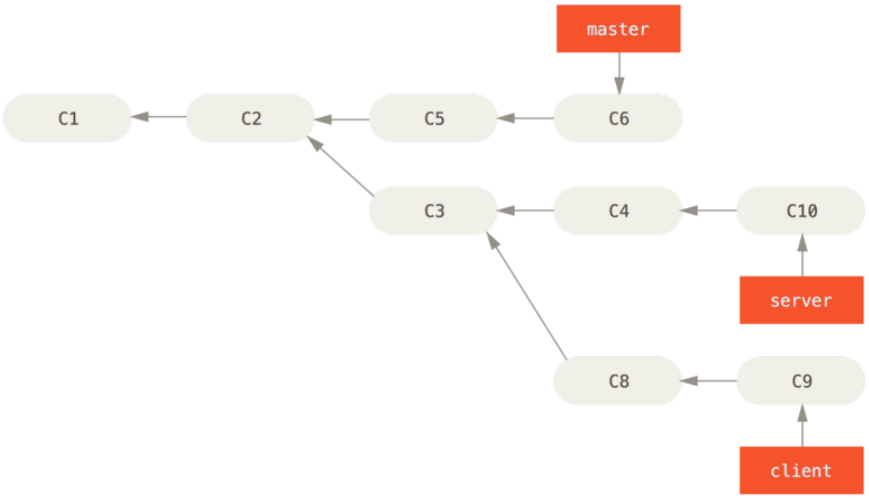
```
git checkout experiment
git rebase master
```

Quan acabe el `rebase`, **experiment seguirà sent la branca activa**.

Altres usos de rebase

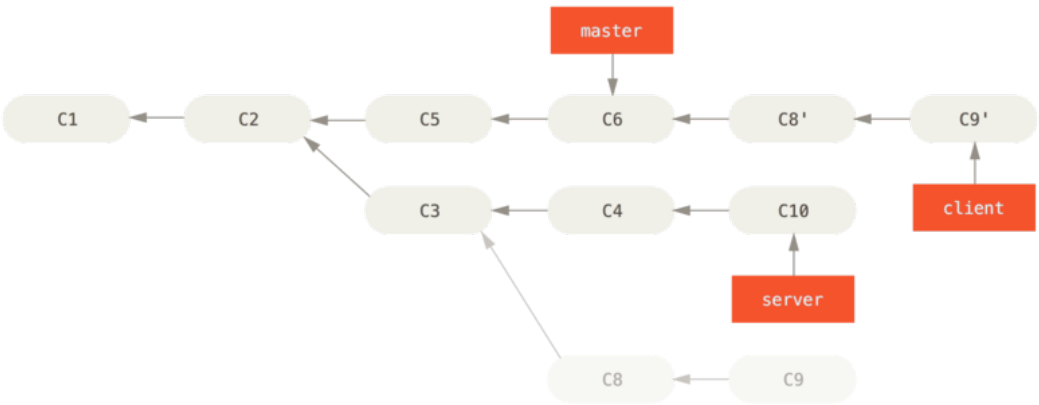
Rebase de branques no directament connectades

Veiem un altre exemple, partint d'una situació com la de la següent figura:



Suposem que decidim fusionar els **canvis de la branca del client en la línia master** per a un llançament de codi, però volem mantindre **sense aplicar els canvis del server** fins que es facen més proves. Es pot prendre els canvis en el client que no estan al servidor (C8 i C9) i reproduir-los en la branca principal mitjançant l'opció **`--onto`**:

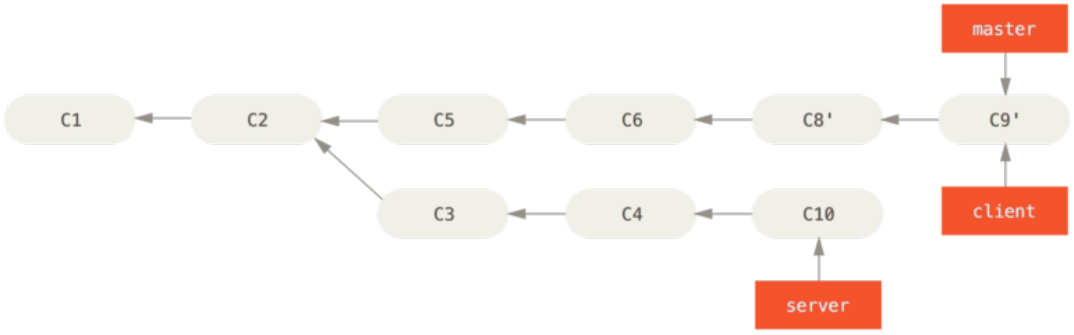
```
git rebase --onto master server client
```



Això bàsicament diu: "Agafa la branca client, esbrina els canvis des que es va separar de la branca del servidor, i reprodueix aquests canvis en la branca master."

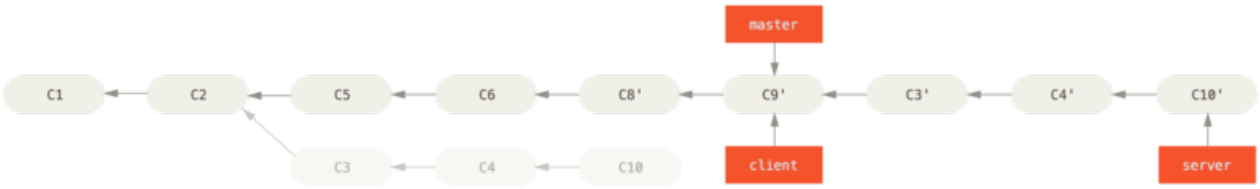
Podem continuar el procés fent un **`merge`** de la branca master i la client:

```
git checkout master
git merge client
```



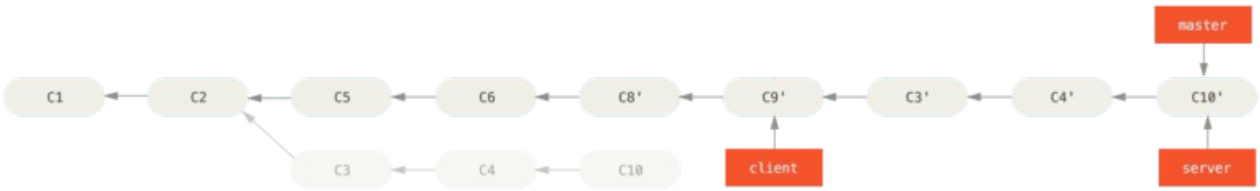
Tot seguit decidim **integrar en la branca master els canvis fets en server**. Com hem vist abans, podem fer el **`rebase`** sense estar en la branca master, amb l'ordre:

```
git rebase master server
```



Després podem **unir la branca master amb la server**, perquè el HEAD del master estiga en el **`commit`** més avançat (fem un **fast forward**):

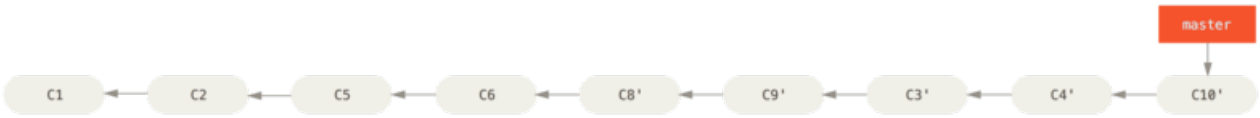
```
git checkout master
git merge server
```



Finalment, podem **eliminar les branques client i server**, perquè el seu treball ja està integrat en la línia principal i ja no es necessiten més, deixant la història final com un

seguit lineal de commits:

```
git branch -d client
git branch -d server
```



Quina tècnica és millor utilitzar, **`rebase`** o **`merge`**? Tot dependrà dels casos particulars, però, en general:

- Si volem que la història reflectisca **tot el procés realitzat per arribar al resultat final**, encara que s'hagen realitzat moltes proves, canvis i errades, s'ha d'utilitzar el **`merge`**.
- Si el que volem és deixar una **història neta i fàcil de seguir per comprendre tots els canvis que s'han fet**, tal vegada siga millor utilitzar **`rebase`**.

Un ús comú sol ser utilitzar **`rebase`** en canvis locals que encara no han sigut compartits amb un **`push`** per deixar més neta la història, però no fer el **`rebase`** d'alguna cosa que ja s'ha pujat.

Utilitzant rebase per a combinar diversos commits en un mateix branch

L'ordre **`rebase`** permet combinar diversos **`commits`** en un d'únic. Això és útil ja que permet a l'usuari **reescriure una mica la història dels `commits`** (fent una mica de neteja) **abans de fer un `push`** dels canvis a un repositori remot.

A continuació crearem diversos **`commits`** que seran combinats més endavant.

Creem un nou arxiu

```
touch rebase.txt
```

Afegim a l'índex i fem el **`commit`**

```
git add . && git commit -m "rebase.txt afegit a l'índex"
```

```
professor@pcprofessor:~/repo01$ touch rebase.txt
professor@pcprofessor:~/repo01$ git add . && git commit -m "rebase.txt afegit a
l'índex"
[master cdf42c9] rebase.txt afegit a l'índex
 2 files changed, 5 insertions(+)
 create mode 100644 conflictiu.txt.orig
 create mode 100644 rebase.txt
professor@pcprofessor:~/repo01$ git log
commit cdf42c9995998080e1c47ffdfb141e4d75214c66 (HEAD -> master)
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 07:30:55 2019 +0200

    rebase.txt afegit a l'índex
```

A continuació realitzem alguns **xicotets canvis** al fitxer, fem un **commit** en cada cas:

```
echo "contingut" >> rebase.txt
git add . && git commit -m "agregue contingut"
echo "més contingut" >> rebase.txt
git add . && git commit -m "agregue més contingut"
echo "més contingut" >> rebase.txt
git add . && git commit -m "agregue més contingut"
echo "més contingut" >> rebase.txt
git add . && git commit -m "agregue més contingut"
echo "més contingut" >> rebase.txt
git add . && git commit -m "agregue més contingut"
```

```
professor@pcprofessor:~/repo01$ echo "més contingut" >> rebase.txt
professor@pcprofessor:~/repo01$ git add . && git commit -m "agregue més contingut"
[master 8168dcf] agregue més contingut
 1 file changed, 1 insertion(+)
professor@pcprofessor:~/repo01$ echo "més contingut" >> rebase.txt
professor@pcprofessor:~/repo01$ git add . && git commit -m "agregue més contingut"
[master 300f840] agregue més contingut
 1 file changed, 1 insertion(+)
```

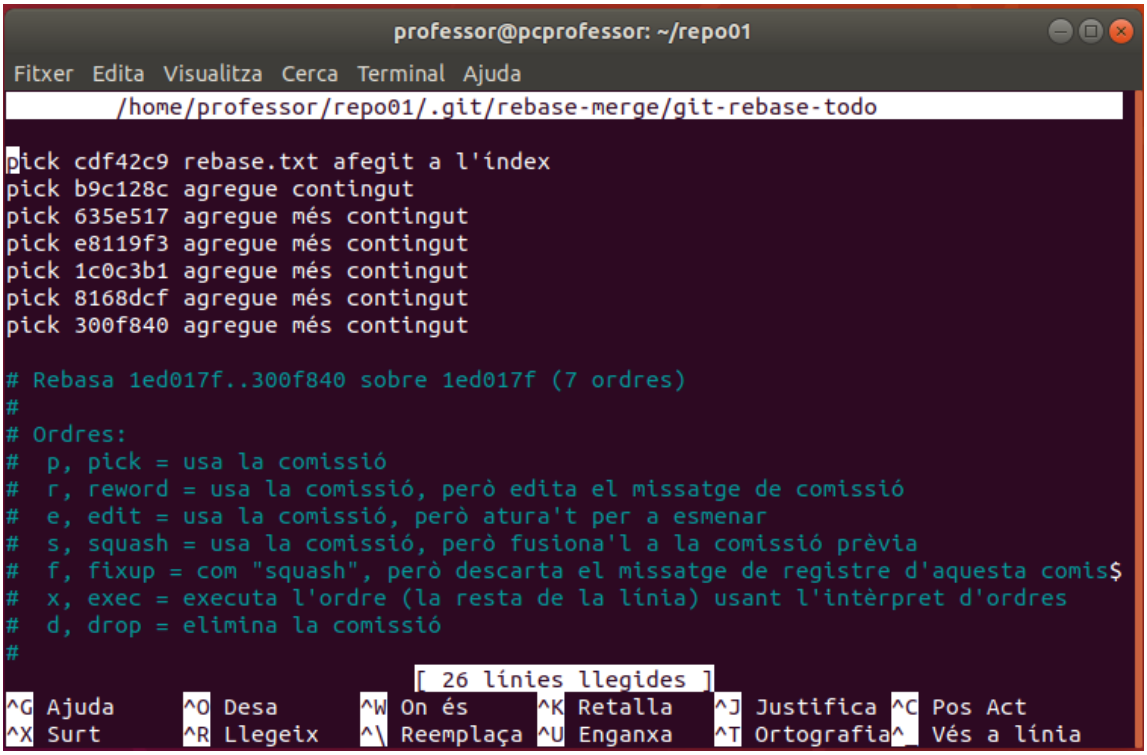
Si mirem el log, s'ha fet molt llarg, amb canvis xicotets.

git log

```
professor@pcprofessor:~/repo01$ git log
commit 300f84071ea55a733aaa9b14b25f154c50169082 (HEAD -> master)
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 07:33:52 2019 +0200

    agregue més contingut

commit 8168dcfe587120fd4fb55791b9476ccbb3bc987e
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 07:33:51 2019 +0200

    agregue més contingut

commit 1c0c3b141ab9d3aa69e1d6593699fc264e22e52e
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 07:33:51 2019 +0200

    agregue més contingut
```

Ara **combinarem els últims set `commits`** mitjançant la següent ordre:

git rebase -i HEAD~7

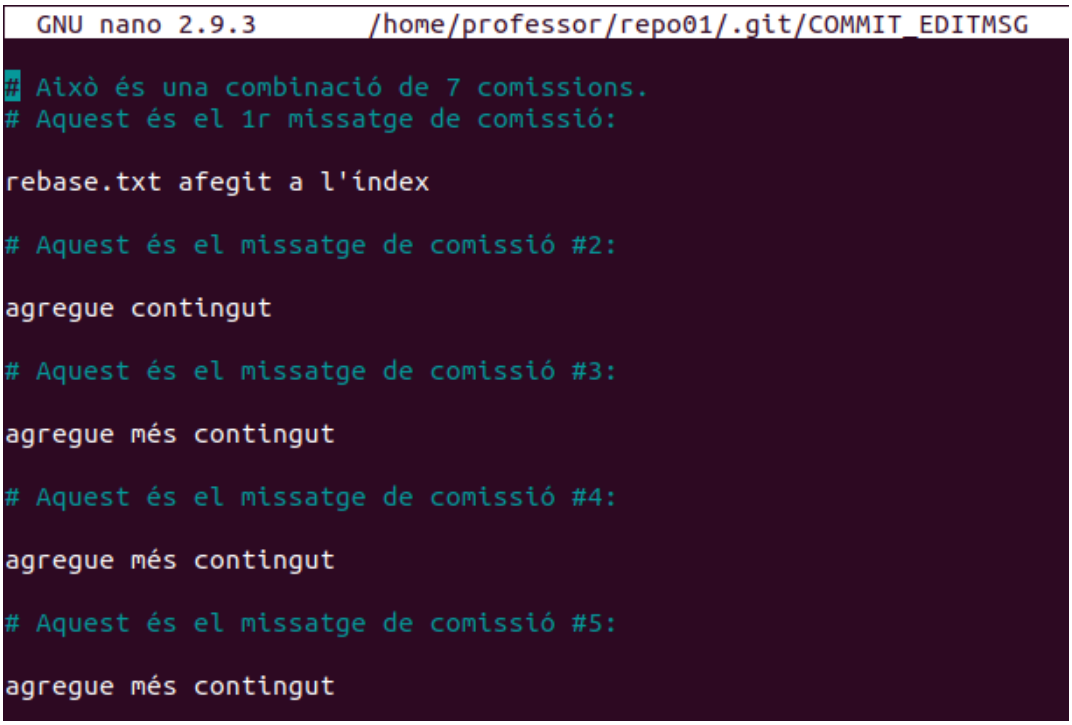


Amb el paràmetre -i (interactive) s'obre un editor de text (nano, per exemple) i deixa

editar el missatge de **commit** per utilitzar en cada **commit** qualsevol de les opcions que es mostren: mantenir (**pick**), mantenir i canviar el missatge (**reword**), compactar amb el **commit** anterior (**squash**) / arreglar (**fixup**) o eliminar (**drop**).

Per exemple, si volguérem fusionar tots els **commits**, utilitzaríem en totes les branques l'opció **squash**, excepte en la primera, que utilitzarem com a **commit** per a reunir-les.

Tot seguit, després de guardar i tancar aquest fitxer, s'obre de nou un editor de text per modificar el missatge corresponent al **commit** resultant:



Deixem només un missatge indicant que s'ha fet un **rebase**:

```
 GNU nano 2.9.3        /home/professor/repo01/.git/COMMIT_EDITMSG        Modificat

# Això és una combinació de 7 comissions.
# Aquest és el 1r missatge de comissió:

rebase.txt afegit a l'índex, CANVIS UNITS AMB rebase

# Introduïu el missatge de comissió dels vostres canvis.
# S'ignoraran les línies començant amb '#', i un missatge de
# comissió buit avorta la comissió.
```

Finalment té lloc la unió dels diferents **commits** en un de sol, canviant el missatge:

```
professor@pcprofessor:~/repo01$ git rebase -i HEAD~7
[HEAD separat f52ef20] rebase.txt afegit a l'índex, CANVIS UNITS AMB rebase
 Date: Tue Apr 9 07:30:55 2019 +0200
 2 files changed, 11 insertions(+)
 create mode 100644 conflictiu.txt.orig
 create mode 100644 rebase.txt
Successfully rebased and updated refs/heads/master.
professor@pcprofessor:~/repo01$ git log
commit f52ef203cc7212e549d2fb72bf9fe942aa9e1e31 (HEAD -> master)
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 07:30:55 2019 +0200

    rebase.txt afegit a l'índex, CANVIS UNITS AMB rebase
```

Podem veure com els continguts introduïts en **rebase.txt**, també es mantenen:

```
professor@pcprofessor:~/repo01$ cat rebase.txt
contingut
més contingut
més contingut
més contingut
més contingut
més contingut
```

rebase entre branches

Com ja hem descrit gràficament abans, l'ordre **merge** combina els canvis de dues branques. **rebase** pren els canvis d'una branca, crea un **patch** i l'aplica a l'altra branca. Com ja hem indicat també prèviament, el resultat final del codi font és el mateix que amb **merge** però la història de **commit** és més neta. La història sembla ser lineal.

Creem una **nova branca** i ens hi canviem:

```
git branch provarebase
git checkout provarebase
git branch
```

```
professor@pcprofessor:~/repo01$ git branch
* master
professor@pcprofessor:~/repo01$ git branch provarebase
professor@pcprofessor:~/repo01$ git checkout provarebase
S'ha canviat a la branca «provarebase»
professor@pcprofessor:~/repo01$ git branch
  master
* provarebase
```

Fem alguns **canvis** i el **commit** corresponent, comprovant la història després:

```
echo "Farem un rebase amb el màster"> test01
git commit -a -m "Alguna cosa nova en el branch"
git log
```

```
professor@pcprofessor:~/repo01$ echo "Farem un rebase amb el màster"> test01
professor@pcprofessor:~/repo01$ git commit -a -m "Alguna cosa nova en el branch"
[provarebase 4b3a36f] Alguna cosa nova en el branch
 1 file changed, 1 insertion(+), 1 deletion(-)
professor@pcprofessor:~/repo01$ git log
commit 4b3a36ff1ca0234c9b4230bfe697f9ab403d702c (HEAD -> provarebase)
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 08:10:29 2019 +0200

    Alguna cosa nova en el branch
```

Ara tornem a la branca **master** i realitzem algun **commit**:

```
git checkout master
echo "Farem un canvi amb el màster"> test02
git commit -a -m "Alguna cosa nova en el master"
```

```
professor@pcprofessor:~/repo01$ git checkout master
S'ha canviat a la branca «master»
professor@pcprofessor:~/repo01$ echo "Farem un canvi amb el màster"> test02
professor@pcprofessor:~/repo01$ git commit -a -m "Alguna cosa nova en el master"
[master 408520d] Alguna cosa nova en el master
 1 file changed, 1 insertion(+), 1 deletion(-)
professor@pcprofessor:~/repo01$ git log
commit 408520d29c9302c7a33f3b6cae9776c68a3b5a39 (HEAD -> master)
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 08:12:45 2019 +0200

    Alguna cosa nova en el master
```

Tornem a la branca **provarebase**, fem un **rebase** amb el màster i comprovem el resultat amb el log:

```
git checkout provarebase
git rebase master
git log
```

```
professor@pcprofessor:~/repo01$ git checkout provarebase
S'ha canviat a la branca «provarebase»
professor@pcprofessor:~/repo01$ git rebase master
Primer, s'està rebobinant HEAD per a reproduir el vostre treball al damunt...
S'està aplicant: Alguna cosa nova en el branch
professor@pcprofessor:~/repo01$ git log
commit df2fd46ea0dd78aac013ff083271e861145038ef (HEAD -> provarebase)
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 08:10:29 2019 +0200

    Alguna cosa nova en el branch

commit 408520d29c9302c7a33f3b6cae9776c68a3b5a39 (master)
Author: Professor Entorns <professor@entorns.com>
Date:   Tue Apr 9 08:12:45 2019 +0200

    Alguna cosa nova en el master
```