

Introducció a Git

Què és Git?

- És un sistema de control de versions
 - Permet a diferents persones treballar en paral·lel en el mateix projecte
 - Es guarda un històric (*repositori*) de tots els canvis fets al projecte
- És totalment distribuït
 - No hi ha cap servidor central que guardi versions de referència
 - Per exemple, un mateix repositori es pot replicar intacte des de GitHub cap a Gitlab
- Internament un repositori és un graf amb una llista de punters
 - Els nodes són conjunts de canvis (*commits*)
 - Els punters poden apuntar a l'últim canvi d'una branca (*HEAD*) o a versions o releases importants (*tags*)
- Ara mateix és l'estàndard de facto

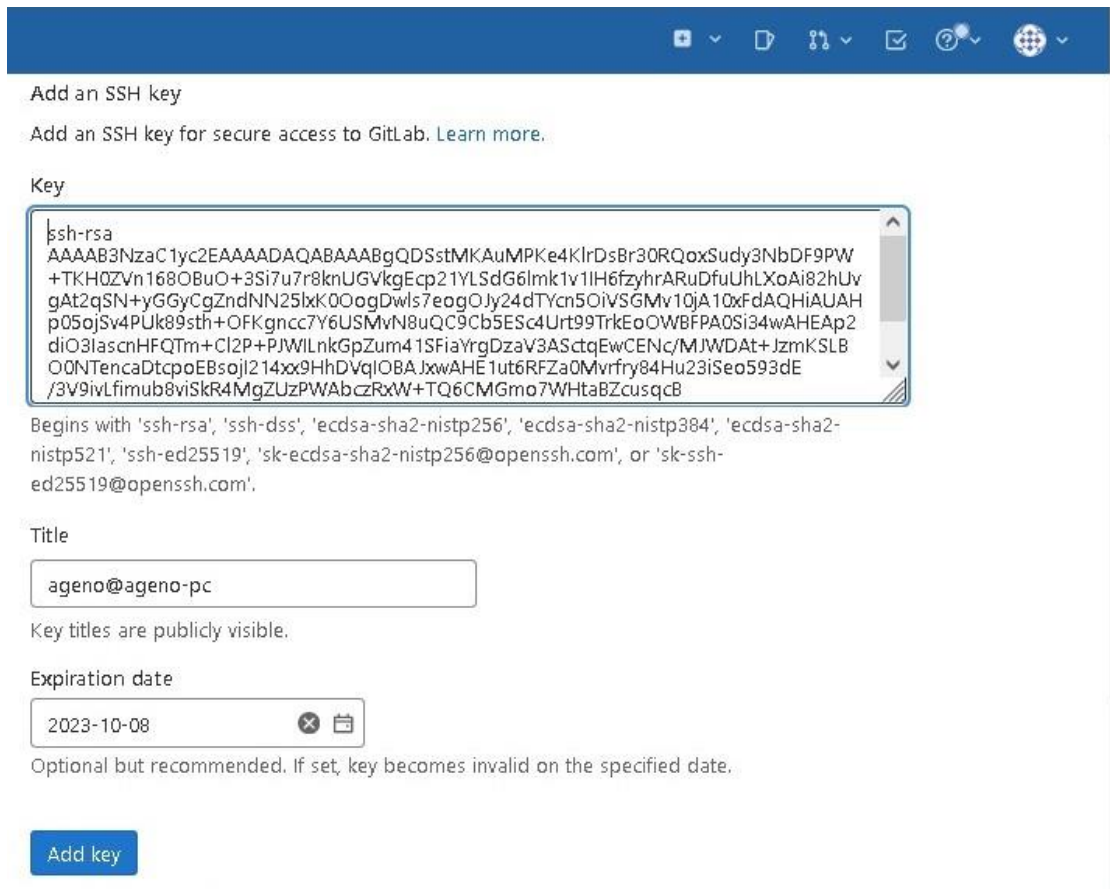
Què no és Git?

- NO és emmagatzemament al cloud
 - No està optimitzat per guardar versions de fitxers binaris
- NO és una eina de backup
 - L'històric d'un sistema de control de versions no és immutable i es pot modificar (*rebase*)
- Per què no utilitzar doncs Dropbox/Google Cloud/WhatsApp/e-mail?
 - Gestió de conflictes pobre
 - Capacitat de navegar per l'històric inexistent o poc usable
 - Les eines de suport al desenvolupament avui en dia totes s'integren amb Git
 - IDEs
 - integració contínua
 - portals de visualització i edició de codi (GitHub, Gitlab, Bitbucket, etc)

Preparació per utilitzar git dins de la VPN

- Connectar a la VPN: <https://serveistic.upc.edu/ca/upclink/documentacio>
- Instal·lar git: <https://git-scm.com/downloads>
- git config (username y email)
- Si no teniu una clau privada creada (normalment estarà a ~/.ssh/id_rsa):
 - Crear una clau privada:
 - ssh-keygen -t rsa -b 4096 (opcional i recomanat posar password)
 - A windows: https://www.tutorialspoint.com/gitlab/gitlab_ssh_key_setup.htm
 - Afegir el contingut de la clau pública al vostre perfil de Gitlab:
 - cat ~/.ssh/id_rsa.pub <- copiar
 - <https://repo.fib.upc.edu/-/profile/keys> <- enganxar
- Això us permetrà utilitzar el protocol ssh en comptes del protocol https
 - Recomanat per no haver d'introduir constantment correu i contrasenya de la UPC

Preparació per utilitzar git



The image shows a web form for adding an SSH key to GitLab. The form has a blue header bar with icons for video, document, settings, email, help, and a globe. Below the header, the text 'Add an SSH key' is followed by a link 'Add an SSH key for secure access to GitLab. Learn more.' The 'Key' section contains a text area with a long SSH key starting with 'ssh-rsa'. Below the text area, there is a paragraph explaining the key format. The 'Title' section has a text input field with the value 'ageno@ageno-pc'. Below this, there is a note 'Key titles are publicly visible.' The 'Expiration date' section has a date input field with the value '2023-10-08' and a calendar icon. Below this, there is a note 'Optional but recommended. If set, key becomes invalid on the specified date.' At the bottom, there is a blue button labeled 'Add key'.

Add an SSH key

Add an SSH key for secure access to GitLab. [Learn more.](#)

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGDStMKAuMPKe4KlrDsBr30RQoxSudy3NbDF9PW
+TKH0ZVn1680BuO+3Si7u7r8knUGVkgEcp21YLSdG6lmk1v1IH6fzyhrARuDfuUhLXoAi82hUv
gAt2qSN+yGGyCgZndNN25lxK0OogDwls7eogOJy24dTYcn5OiVSGMv10jA10xFdAQHiAUAH
p05ojSv4PUk89sth+OFGgncc7Y6USMvN8uQC9Cb5ESc4Urt99TrkEoOWBFPA0Si34wAHEAp2
diO3lascnHFQTm+Cl2P+PJWILnkGpZum41SFiaYrgDzaV3ASctqEwCENC/MJWDAt+JzmKSLB
OONTencaDtcpoEBsojl214xx9HhDVqIOBAJxwAHE1ut6RFZa0Mvrfry84Hu23iSeo593dE
/3V9ivLfimub8viSkR4MgZUzPWAbczRxW+TQ6CMGmo7WHTaBZcusqcb
```

Begins with 'ssh-rsa', 'ssh-dss', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

Title

ageno@ageno-pc

Key titles are publicly visible.

Expiration date

2023-10-08

Optional but recommended. If set, key becomes invalid on the specified date.

Add key

Com crear un repositori?

```
bash-3.2$ mkdir nou-projecte  
bash-3.2$ cd nou-projecte/  
bash-3.2$ git init  
Initialized empty Git repository in /private/tmp/nou-projecte/.git/  
bash-3.2$ █
```

Graf del repositori local

Graf del repositori remot

Com pujar el repositori?

(per HTTPS)

```
git remote add origin https://repo.fib.upc.edu/alicia.ageno/nou-projecte.git
```

(per SSH)

```
git remote add origin git@repo.fib.upc.edu:alicia.ageno/nou-projecte.git
```

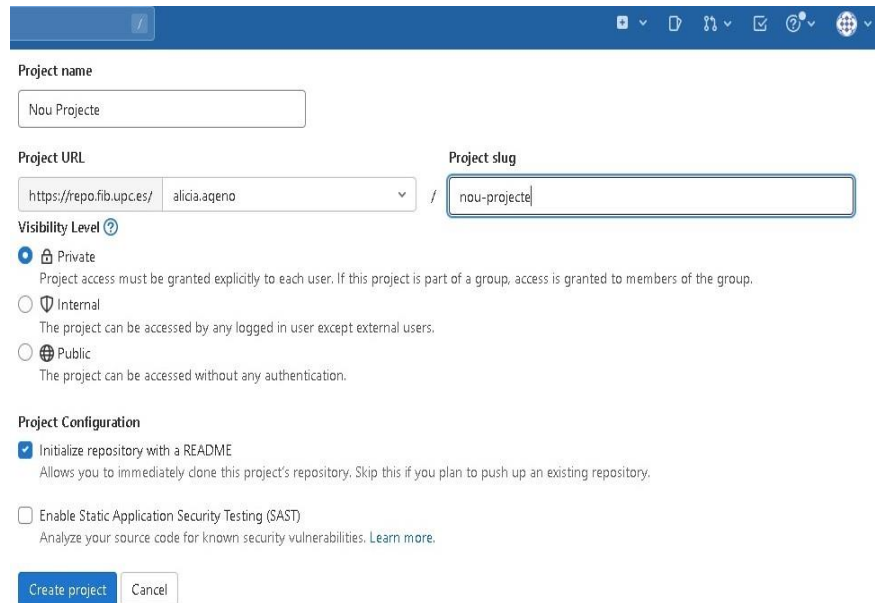
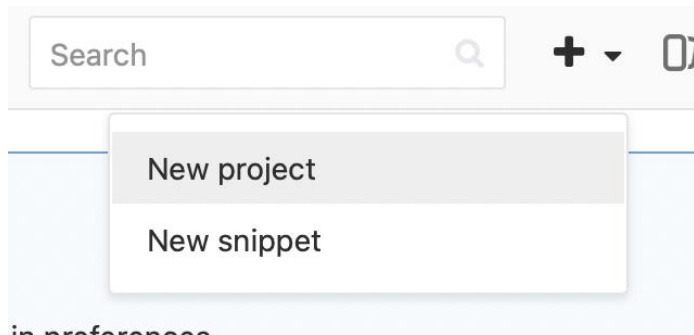
```
git add .
```

```
git commit -m "Commit inicial, no hi ha res"
```

```
git push -u origin master
```

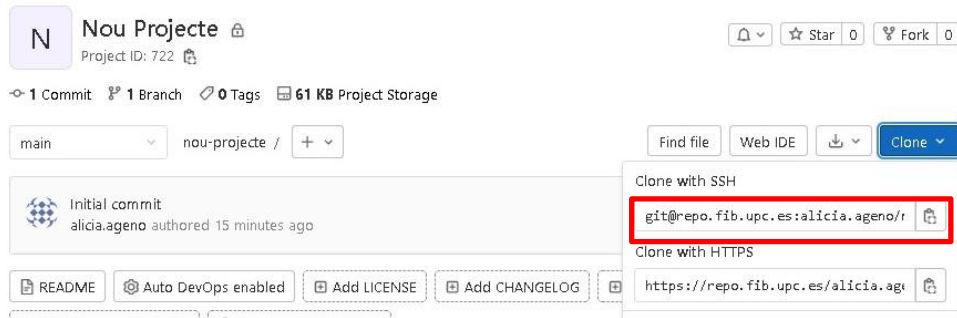
Com clonar un repositori?

Normalment no crearem un repositori localment sinó que el crearem a una interfície com Gitlab i després en farem un clone:



A screenshot of the 'New project' form in the GitLab interface. The form is titled 'Project name' and has a text input field containing 'Nou Projecte'. Below this, the 'Project URL' section shows a dropdown menu with 'https://repo.fib.upc.es/' and 'alicia.ageno' selected. To the right of the URL, there is a 'Project slug' field containing 'nou-projecte'. The 'Visibility Level' section has three radio buttons: 'Private' (selected), 'Internal', and 'Public'. Below this, the 'Project Configuration' section has two checkboxes: 'Initialize repository with a README' (checked) and 'Enable Static Application Security Testing (SAST)' (unchecked). At the bottom, there are two buttons: 'Create project' and 'Cancel'.

Com clonar un repositori?*



\$ **git clone git@repo.fib.upc.edu/usuari/nou-projecte.git**
Cloning into 'nou-projecte'...
warning: You appear to have cloned an empty repository.

Graf del repositori local

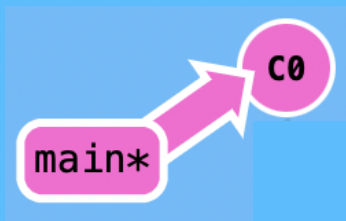
Graf del repositori remot

* GradlePOC: usar **git fork**

Primer commit

```
$ echo "foo bar" >> README.md  
$ git add README.md  
$ git commit -m "Primer commit"  
[master (root-commit) dd5d06c] Primer commit  
1 file changed, 1 insertion(+)  
create mode 100644 README.md
```

Graf del repositori local



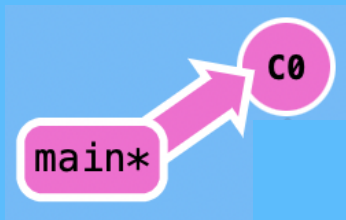
Graf del repositori remot

Primer push

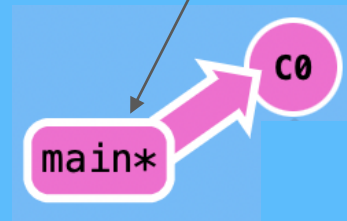
```
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 221 bytes | 221.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@repo.fib.upc.edu:usuari/nou-projecte.git
* [new branch]      master -> master
```

Normalment la branca
per defecte es diu
master o main

Graf del repositori local



Graf del repositori remot

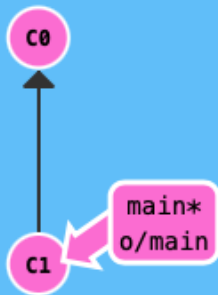


Segon commit (dos fitxers)

```
$ echo "bar baz" >> README.md
$ echo "segon fitxer" >> fitxer
$ git add README.md fitxer
$ git commit -m "Segon commit, dos fitxers"
[master 3097215] Segon commit, dos fitxers
 2 files changed, 2 insertions(+)
 create mode 100644 fitxer
```

```
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 24 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 321 bytes | 321.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To git@repo.fib.upc.edu:usuari/nou-projecte.git
 dd5d06c..3097215  master -> master
```

Graf del repositori local



Graf del repositori remot



Veure l'històric

```
$ git log
```

```
commit 30972156974011bbc9903064c2e0545f1bf74c06 (HEAD -> master, origin/master, origin/HEAD)
```

```
Author: Usuari 1 <usuari1@prop.fib.upc.edu>
```

```
Date:   Fri Mar 5 10:41:27 2021 +0100
```

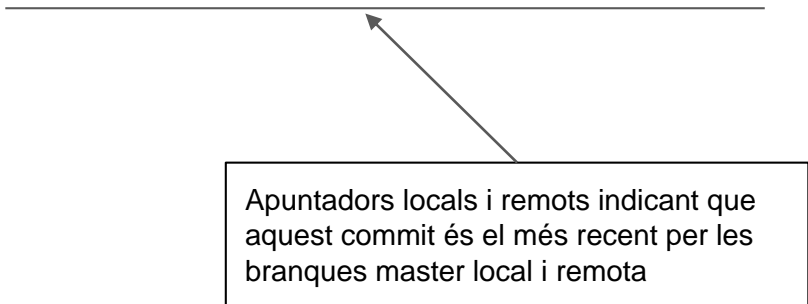
Segon commit, dos fitxers

```
commit dd5d06c6a255a2cc66156187dbd8f0ba34ce64ae
```

```
Author: Usuari 1 <usuari1@prop.fib.upc.edu>
```

```
Date:   Fri Mar 5 10:37:49 2021 +0100
```

Primer commit



Apuntadors locals i remots indicant que aquest commit és el més recent per les branques master local i remota

Dos commits en dos repositoris locals, en paral·lel

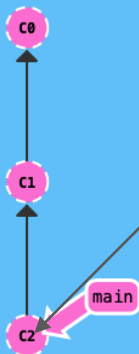
Usuari 1

```
$ echo "1" >> README.md
$ git add README.md
$ git commit -m "Modified README"
[master 01b702a] Modified README
1 file changed, 1 insertion(+)
```

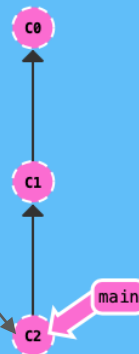
Usuari 2

```
$ echo "2" >> fitxer
$ git add fitxer
$ git commit -m "Modified fitxer"
[master 516a9bf] Modified fitxer
1 file changed, 1 insertion(+)
```

Graf del repositori local Usuari 1



Graf del repositori local Usuari 2



Aquests C2 són diferents!
Però no hi haurà conflicte
perquè no hi ha intersecció
entre els diffs

Veure l'històric

Usuari 1

```
$ git log --oneline
```

```
01b702a (HEAD -> master) Modified README  
3097215 (origin/master) Segon commit, dos fitxers  
dd5d06c Primer commit
```

Usuari 2

```
$ git log --oneline
```

```
516a9bf (HEAD -> master) Modified fitxer  
3097215 (origin/master, origin/HEAD) Segon commit,  
dos fitxers  
dd5d06c Primer commit
```



Commit hashes diferents

Sincronitzar commits

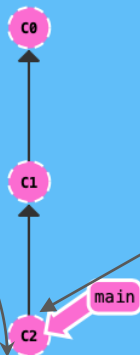
Usuari 1

```
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 24 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 295 bytes | 295.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@repo.fib.upc.edu:usuari/nou-projecte.git
3097215..01b702a master -> master
```

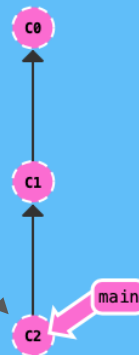
Usuari 2

```
$ git push
To git@repo.fib.upc.edu:usuari/nou-projecte
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@repo.fib.upc.edu:usuari/nou-projecte'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Graf del repositori remot



Graf del repositori local Usuari 2



Aquests C2 segueixen sent diferents i el repositori remot rebutja el segon push

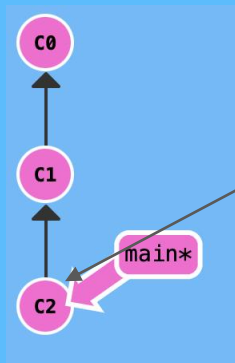
Sincronitzar commits, estratègia *merge*

Usuari 1

Usuari 2

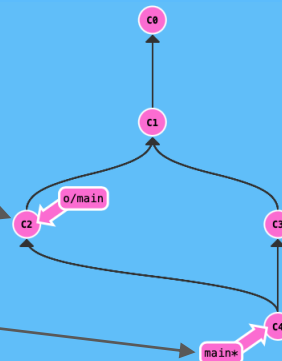
```
$ git pull
.. apareix un editor demanant la descripció del merge, no cal modificar ..
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From git@repo.fib.upc.edu:usuari/nou-projecte
 3097215..01b702a master    -> origin/master
Merge made by the 'recursive' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

Graf del repositori **remot**



S'agafa C2 de remot

Graf del repositori local Usuari 2



C3 = C2 original d'Usuari 2

S'ha creat un "merge commit"

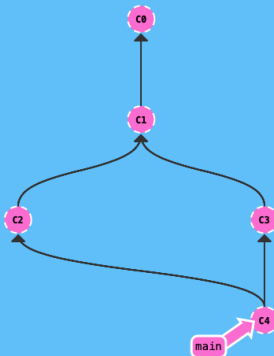
Sincronitzar commits, estratègia *merge*

Usuari 1

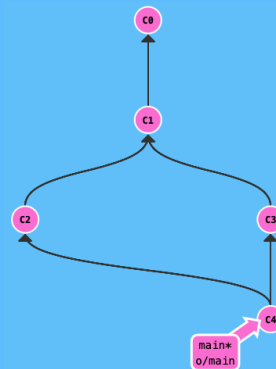
Usuari 2

```
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 24 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 608 bytes | 608.00 KiB/s,
done.
Total 5 (delta 0), reused 0 (delta 0)
To git@repo.fib.upc.edu:usuari/nou-projecte
01b702a..7a21a93  master -> master
```

Graf del repositori **remot**



Graf del repositori local Usuari 2

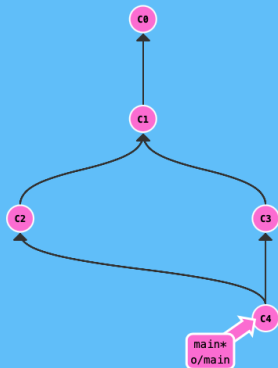


Sincronitzar commits, estratègia *merge*

Usuari 1

```
$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (5/5), done.
From git@repo.fib.upc.edu:usuari/nou-projecte
 01b702a..7a21a93  master    -> origin/master
Updating 01b702a..7a21a93
Fast-forward
 fitxer | 1 +
 1 file changed, 1 insertion(+)
```

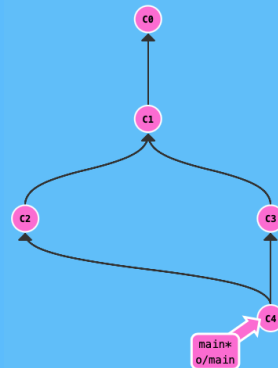
Graf del repositori local **Usuari 1**



Els dos repositoris
locals estan
sincronitzats!

Usuari 2

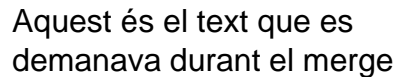
Graf del repositori local **Usuari 2**



Veure l'històric (en mode graf)

```
$ git log --oneline --graph
```

```
* 7a21a93 (HEAD -> master, origin/master) Merge branch 'master' of git@repo.fib.upc.edu:usuari/nou-projecte
| \
| * 01b702a Modified README
* | 516a9bf Modified fitxer
|/
* 3097215 Segon commit, dos fitxers
* dd5d06c Primer commit
```



Aquest és el text que es
demanava durant el merge

Dos commits en dos repositoris locals, amb conflicte

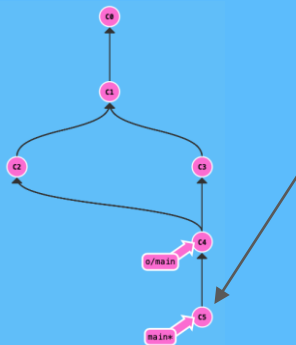
Usuari 1

```
$ echo "Usuari 1" > fitxer  
$ git add fitxer  
$ git commit -m "Set 1 in fitxer"  
[master c1d14a4] Set 1 in fitxer  
1 file changed, 1 insertion(+), 2 deletions(-)
```

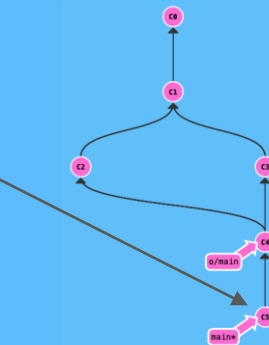
Usuari 2

```
$ echo "Usuari 2" > fitxer  
$ git add fitxer  
$ git commit -m "Set 2 in fitxer"  
[master c2bb96a] Set 2 in fitxer  
1 file changed, 1 insertion(+), 2 deletions(-)
```

Graf del repositori local Usuari 1



Graf del repositori local Usuari 2




Aquests C5 són diferents!
En aquest cas sí hi haurà conflicte

Sincronitzar commits (amb conflicte)

Usuari 1

```
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 24 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@repo.fib.upc.edu:usuari/nou-projecte.git
+ bbcdd10...c1d14a4 master -> master
```

Conflicte, no es pot
solucionar de manera
automàtica (és la mateixa
línia del mateix fitxer)

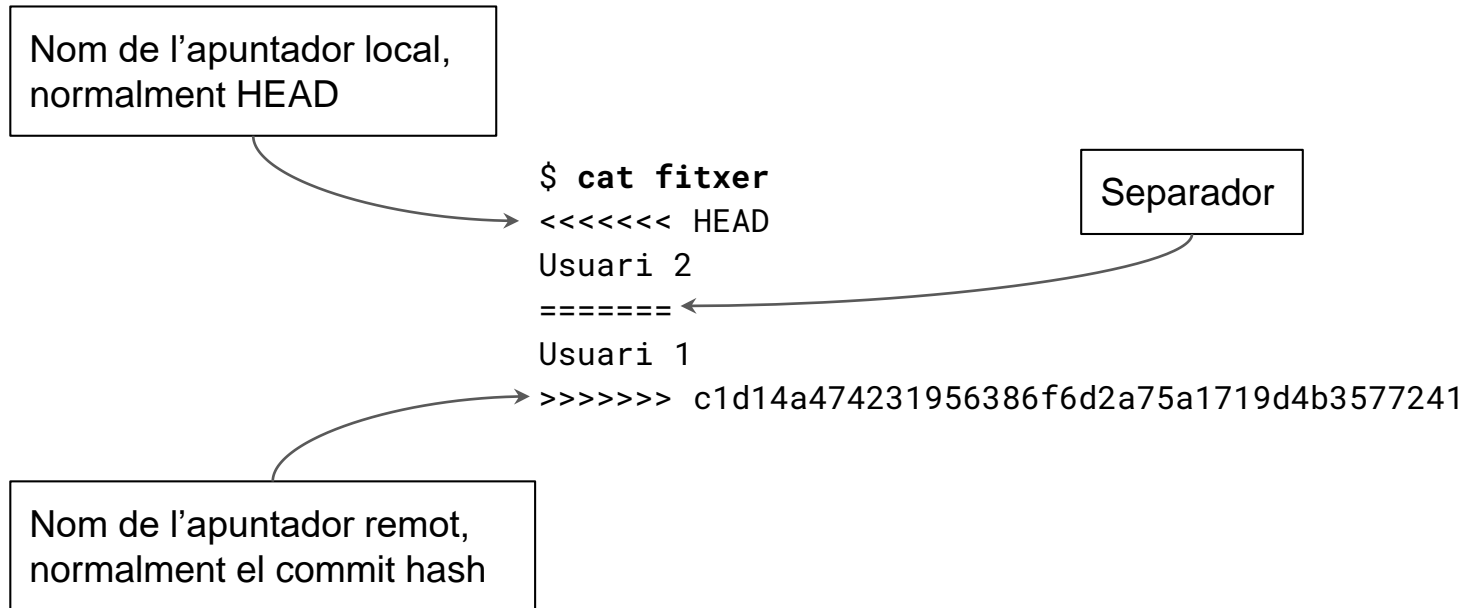


Usuari 2

```
$ git push
To git@repo.fib.upc.edu:usuari/nou-projecte
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@repo.fib.upc.edu:usuari/nou-projecte'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From git@repo.fib.upc.edu:usuari/nou-projecte
+ bbcdd10...c1d14a4 master -> origin/master (forced update)
Auto-merging fitxer
CONFLICT (content): Merge conflict in fitxer
Automatic merge failed; fix conflicts and then commit the result.
```

Format de conflicte

Un conflicte tindrà sempre el mateix patró:



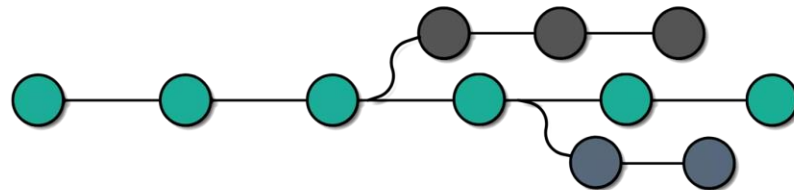
Resolució del conflicte

- Es pot utilitzar **git mergetool** o alguna eina de suport de l'IDE (e.g. IntelliJ)
- O bé es pot fer a mà:

```
$ vi fitxer    # Es pot editar amb qualsevol editor de text
$ cat fitxer   # Hem eliminat els indicadors de conflicte de diff i hem ordenat
Usuari 1
Usuari 2
$ git add fitxer    # Afegim el fitxer al commit actual (buit)
$ git commit -m "Sort file contents"
[master d71a344] Sort file contents
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 24 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 572 bytes | 572.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To git@repo.fib.upc.edu:usuari/nou-projecte
c1d14a4..d71a344  master -> master
```


Branques

- Representa una línia de desenvolupament o entorn independent
- Al mateix projecte, però separat del projecte principal (branca “**main**”)
- Per què fer servir branques?
 - Mantenir sempre una versió funcional a “**main**”
 - Evitar conflictes innecessaris durant el desenvolupament de diferents parts de la implementació que en principi no tenen dependència
 - Experimentar amb noves funcionalitats que es poden descartar fàcilment sense haver de revertir canvis

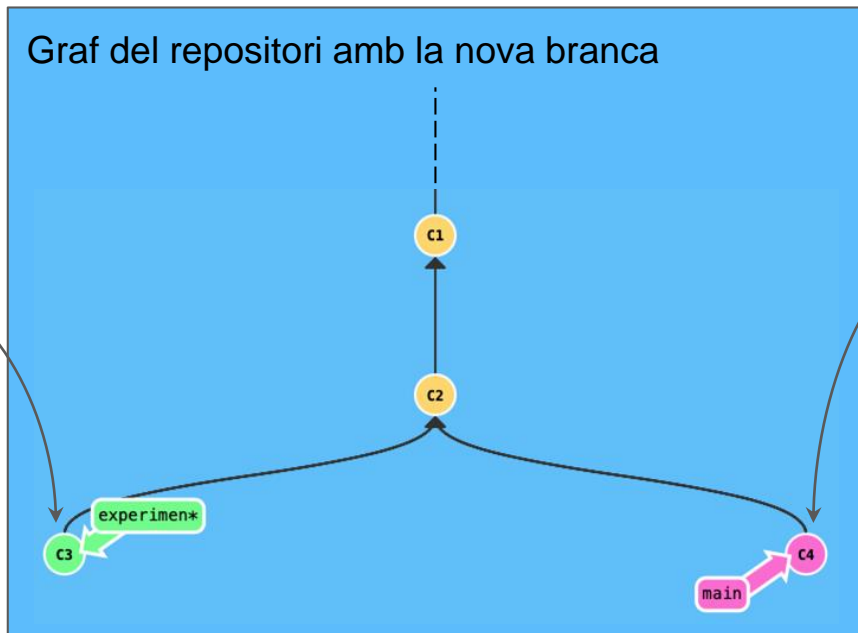


Branques - Creació

```
$ git checkout -b experiment
Switched to a new branch 'experiment'
$ echo "3" >> fitxer
$ git add fitxer
$ git commit -m "Modified fitxer"
[experiment 7e199b1] Modified fitxer
1 file changed, 1 insertion(+)
```

```
$ git checkout master
Switched to branch 'master'
$ echo "1" >> test
$ git add test
$ git commit -m "Create other file"
[master 1de3ebf] Create other file
1 file changed, 1 insertion(+)
create mode 100644 test
```

Graf del repositori amb la nova branca



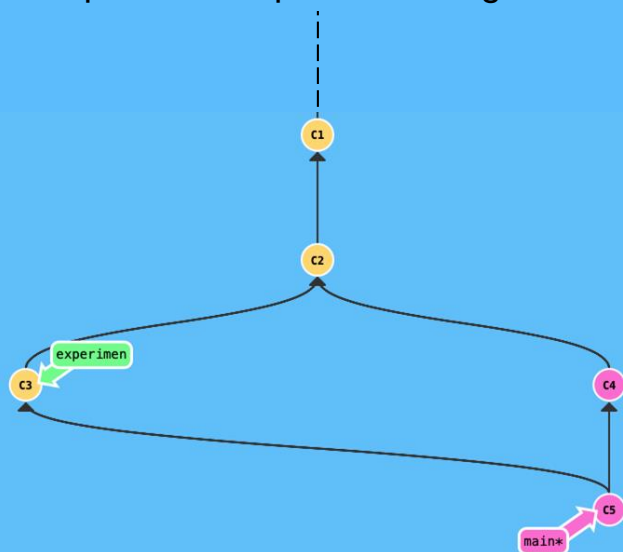
Podem avançar en les dos branques en paral·lel, tant en local com en remot

Branques - Merge

- En algun moment, voldrem unir els nous canvis de la branca amb la branca del projecte principal

```
$ git checkout master
Switched to branch 'master'
$ git merge experiment
Merge made by the 'recursive' strategy.
  fitxer | 1 +
  1 file changed, 1 insertion(+)
$ git log --oneline --graph
*   754edfe (HEAD -> master) Merge branch 'experiment'
| \
| * 7e199b1 (experiment) Modified fitxer
* | 1de3ebf Create other file
|/
* d71a344 (origin/master, origin/HEAD) Sort filecontents
..
```

Graf del repositori després del merge

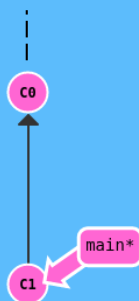


Desfer canvis – no destructiu

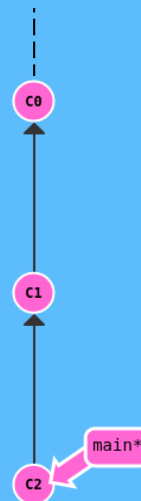
- Podem crear un commit que desfa tots els canvis d'un altre commit

```
$ echo "1" >> prova
$ git add prova
$ git commit -m "Add 1"
[main 8c7f573] Add 1
1 file changed, 1 insertion(+)
create mode 100644 prova
$ echo "2" >> prova
$ git add prova
$ git commit -m "Add 2"
[main 1ecb543] Add 2
1 file changed, 1 insertion(+)
$ git revert 1ecb543
[main c523711] Revert "Add 2"
1 file changed, 1 deletion(-)
$ cat prova
1
```

Graf del repositori abans del revert



Graf del repositori després del revert

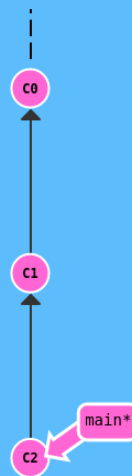


Desfer canvis – destructiu!

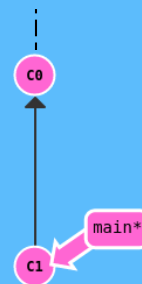
- Amb `git push --force-with-lease` podem modificar permanentment l'històric. Aneu amb compte!

```
$ echo "1" >> prova
$ git add prova
$ git commit -m "Add 1"
[main 8c7f573] Add 1 ...
$ echo "2" >> prova
$ git add prova
$ git commit -m "Add 2"
[main 1ecb543] Add 2 ...
$ git push
Enumerating objects: 5, done.
...
$ git reset --hard 8c7f573
HEAD is now at 8c7f573 Add 1
$ git push --force-with-lease
Total 0 (delta 0), reused 0 (delta 0)
To git@repo.fib.upc.edu:usuari/nou-projecte
+ 1ecb543...8c7f573 master -> master (forced update)
```

Graf del repositori després del primer push



Graf del repositori després del segon push



Desfer canvis – destructiu!

- Què es pot fer amb `git push --force-with-lease`?
- Juntar molts commits en un (squash)
 - <https://www.studytonight.com/git-guide/git-squash>
- Modificar un commit en concret (per exemple, per esborrar d'un commit un fitxer que no s'hauria d'haver pujat o per canviar el missatge)
 - <https://www.bryanbraun.com/2019/02/23/editing-a-commit-in-an-interactive-rebase/>
- Diferències entre `--force` i `--force-with-lease`:
 - <https://blog.developer.atlassian.com/force-with-lease/>

Altres conceptes avançats

- Estratègies de branques

- <https://www.creativebloq.com/web-design/choose-right-git-branching-strategy-121518344>
- <https://www.atlassian.com/git/tutorials/merging-vs-rebasing>
- Joc interactiu: <https://learngitbranching.js.org>

- Integració contínua

- Automatització de proves unitàries, execució de codi, desplegament a producció, etc. amb cada push
- Gitlab (e.g. al de la FIB): <https://docs.gitlab.com/ee/ci/>
- GitHub: <https://github.com/features/actions>

- En cas de pànic

- **git reflog**: <https://www.unleashed-technologies.com/blog/git-reflog-has-your-back>
- **git cherry-pick**: <https://www.atlassian.com/git/tutorials/cherry-pick>
- 17 casos pràctics: <https://www.git-tower.com/blog/surviving-with-git-videos>

- Per aprendre més

- <https://www.youtube.com/watch?v=HiXLkL42tMU>
- Per fer servir Git com una eina més de comunicació entre els membres de l'equip:
<https://chris.beams.io/posts/git-commit/>