Gestió de Projectes Software:

Laboratori

GIT i Markdown

Curs 2017-18, QT

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Control de versions

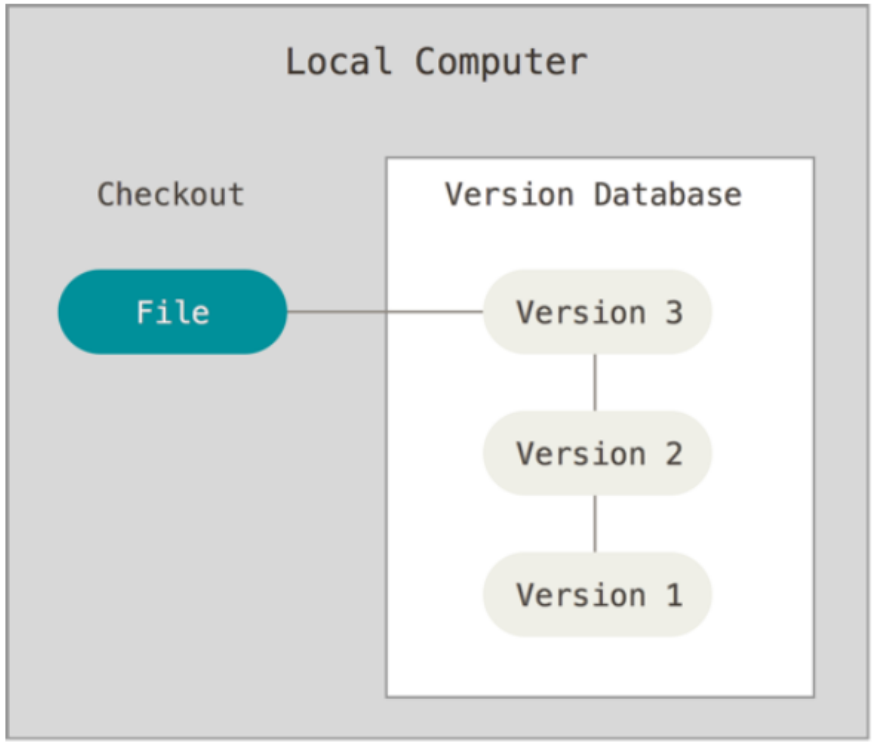
Version control is a system that **records changes** to a file or set of files over time so that you can recall specific versions later. You can do this with nearly any type of file on a computer.

It **allows** you to **revert files** back to a previous state, **revert the entire project** back to a previous state, **compare changes** over time, **see who last modified something** that might be causing a problem, who introduced an issue and when, and more.

Using a VCS also generally means that **if you screw things up** or lose files, **you can easily recover**. In addition, you get all this for very little overhead.

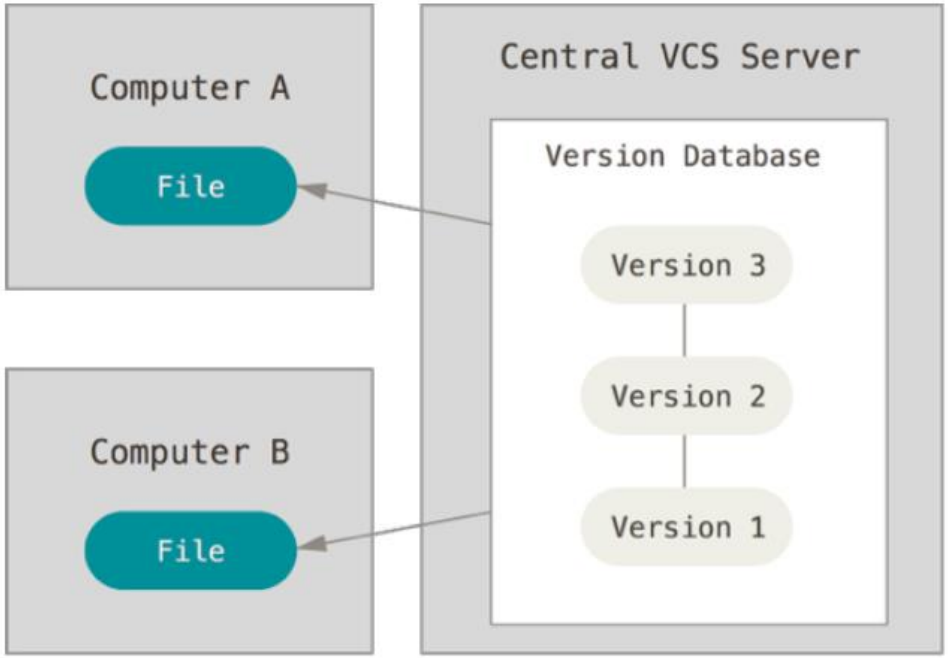
Tipus de control de versions
Local

- Local
  - Els fitxers i la BD de versions estan a una màquina local (no permet compartir)
  - Exemple: RCS



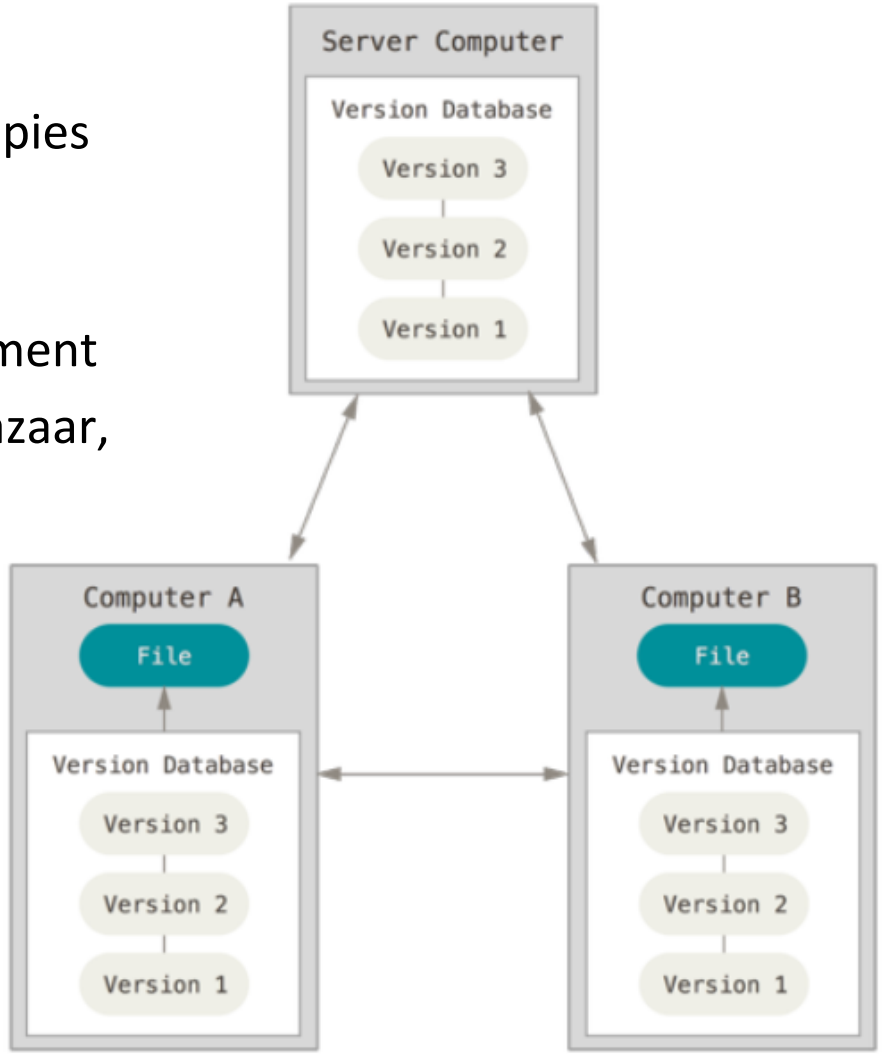
Tipus de control de versions
Centralitzat

- Centralitzat
  - La BD de versions (i potser també els fitxers) estan a un servidor centralitzat
  - Estàndard durant molts anys. El servidor és un punt de fallida
  - Exemples: CVS, SourceSafe i Team Foundation Server, Subversion



Tipus de control de versions
Distribuït

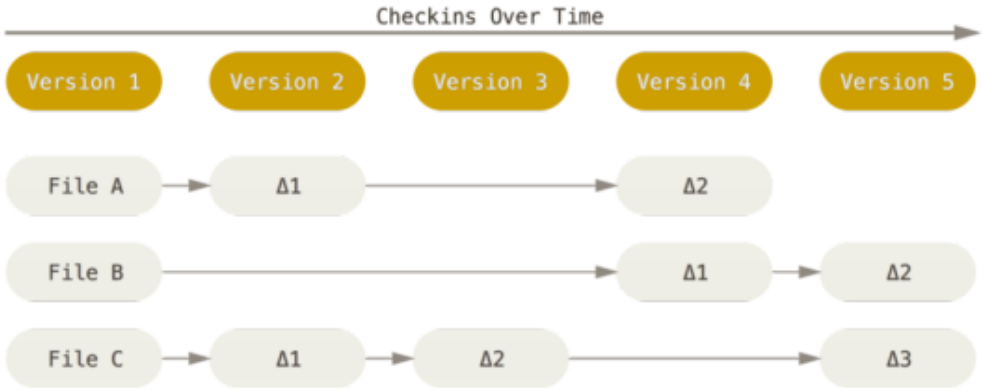
- Distribuït (DCVS)
  - Totes les màquines tenen còpies idèntiques dels repositoris
  - Major eficiència i flexibilitat
  - Paradigma dominant actualment
  - Exemples: GIT, Mercurial, Bazaar, BitKeeper (inicis GIT)



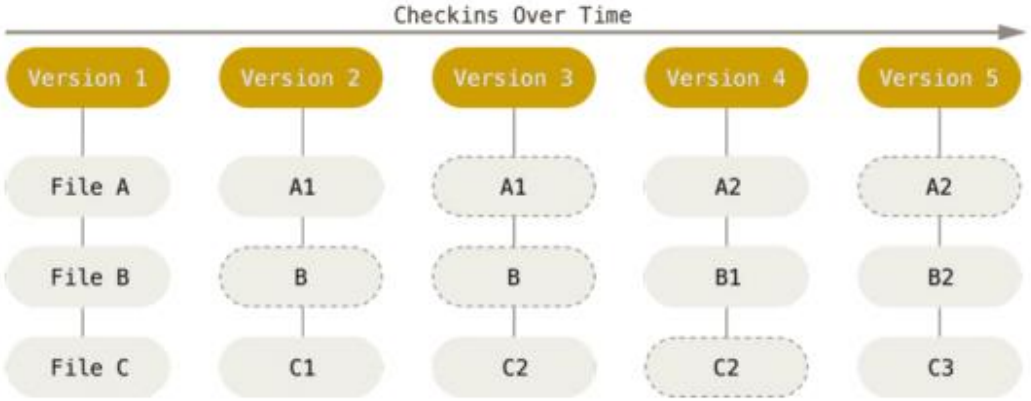
GIT

Diferència principal amb altres sistemes

- Altres sistemes: Emmagatzematge llista canvis als arxius



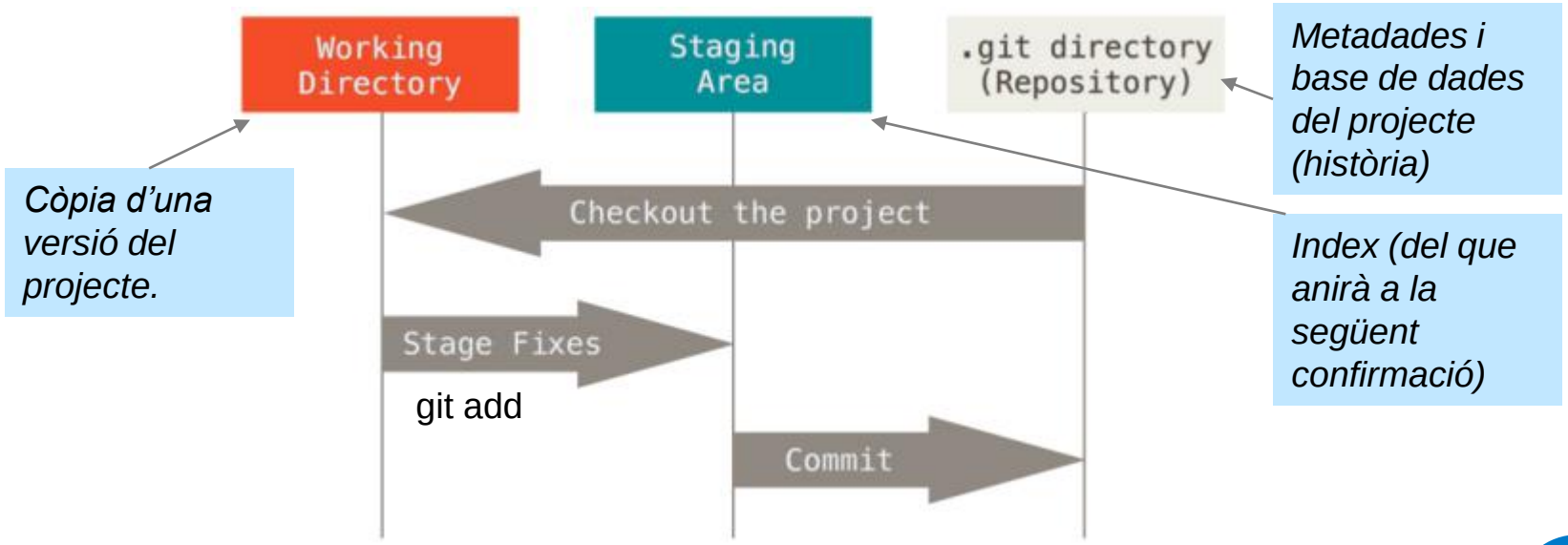
- GIT: Còpies instantànies (cada vegada que confirmem una còpia es fa una foto)



GIT

Estats dels fitxers i espais

- Rastrejats
  - Modificat Arxiu encara no preparat
  - Staged (preparat) Arxiu marcat per ser inclòs a la següent confirmació
  - Commited (confirmat). Dades de forma segura a la base de dades local
- Sense rastrejar
- Ignorats



GIT

Configuració

- Configuració global
  - Identitat
  - Editor

```
$ git config --global user.email "usuari@correu.domini"
$ git config --global user.name "Nom Cognom1 Cognom2"
$ git config --global core.editor nano
$ git config --global --list
```

GIT

Crear repositori

- `git init`
  - Crea un repositori local (a una carpeta amb o sense fitxers)
  - Es crea una subcarpeta .git amb les metadades del repositori i una branca ‘master’

```
$ mkdir lab1
$ cd lab1
$ git init
S’ha inicialitzat un dipòsit buit del Git en
/home/osboxes/lab1/.git/
```

GIT

Com es troba el repositori

- Estat (git status)
  - Estat del directori de treball i de l'àrea de staging (NO DEL REPOSITORI)
  - Fitxers seguits (staged, unstaged), Fitxers no seguits (untracked)
  - No es mostra informació de la història de commits (s'ha d'usar `git log`)

```
$ echo "Hola món" > doc1.md
$ git status

En la branca master
Comissió inicial
Fitxers no seguits:
  (useu "git add <fitxer>..." per a incloure-ho en què es cometrà)

        doc1.md
no hi ha res afegit a cometre però fitxers no seguits estan presents
(useu "git add" per a seguir-los)
```

GIT

El flux general – Afegir a staging

- `git add`
  - Afegeix els canvis del directori de treball (o d'un únic fitxer) a la zona staging (índex)

```
$ git add doc1.md
$ git status
En la branca master
Comissió inicial

Canvis a cometre:
  (useu "git rm --cached <fitxer>..." per a desallistar)

        fitxer nou:       doc1.md
```

  - Canvis a cometre (Changes to be committed): fitxers zona staging (controlats però no confirmats)

GIT

Ignorar fitxers

- Fitxer .gitignore
  - Hi ha fitxers que normalment no interessa incloure a GIT (p.e. binaris)
  - Fitxers ignorats no apareixen a `status` ni es consideren a `git add .`
  - Afegir fitxer .gitignore al cicle com la resta de fitxers (add, commit)

```
$ echo ‘fitxer binari fals’ > fitxer.exe
$ git status
Fitxers no seguits:
  (useu "git add <fitxer>..." per a incloure-ho en què es cometrà)
        fitxer.exe
$ touch .gitignore
$ nano .gitignore (i afegir una línia *.exe)
$ git add .gitignore
$ git commit -m "add .gitignore“
$ git status
En la branca master
no hi ha res a cometre, l'arbre de treball està net
```

GIT

Fem un breu parèntesi - Markdown

> Markdown és un llenguatge de marques lleuger, originalment creat per John Gruber i Aaron Swartz que permet "escriure utilitzant un format de text planer fàcil d'escriure i de llegir i després convertir-ho en un XHTML o HTML estructuralment vàlid".

- En principi pensat per escriure texts per la web amb més rapidesa i senzillesa
- No hem de tenir gaire atenció al format
- Revisar:
  - https://daringfireball.net/projects/markdown/basics
  - https://bitbucket.org/jordipradel/gps-markdown

GIT

El flux general – Modifiquem el fitxer

- Modificació del fitxer

```
$ nano doc1.md
# Títol 1

I ara... una llista:
1. Primer
2. Segon
3. Tercer
```

- `git status`

```
Canvis a cometre:
  (useu "git rm --cached <fitxer>..." per a desallistar)

        modificat:       doc1.md

Canvis no allistats per a cometre:
  (useu "git add <fitxer>..." per a actualitzar què es cometrà)
  (useu "git checkout -- <fitxer>..." per a descartar els canvis en el directori de treball)

        modificat:       doc1.md
```

GIT

El flux general – Confirmem índex

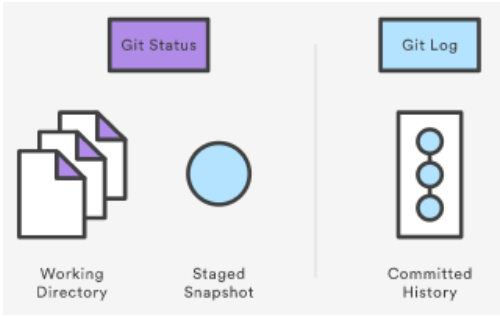
- `git commit` (confirmar fitxers)
  - Envia tots els fitxers de la zona staging al repositori empaquetats en un commit
  - El commit s'identifica amb un hash, però podem usar tags (lightweight o annotated)
    - git tag –m "Tag per la versió 1.0" v1.0
  - Flags
    - Sense flag – S'obre editor per incloure missatge descriptiu
    - –m → permet incloure un missatge descriptiu (***RECOMANAT***)
    - –a → saltar-se l'àrea de preparació. NO USAR!!!!

```
$ git commit -m "Canvis laboratori fitxer markdown"
[master (comissió d'arrel) a53bc11] Canvis laboratori
fitxer markdown
 1 file changed, 6 insertions(+)
 create mode 100644 doc1.md
```

GIT

Com es troba el repositori - Commits

- `git log <idcommit>`
  - Històric dels canvis al repositori (commits)
  - Flags habituals
    - --oneline → Mostra l'històric en format resumit
    - --stat → Fitxers i nombre línies modificades
    - -p → Mostra les diferències entre els commits (el més complet)
    - --grep="<patró>“ / --author=“<patró>” / <des de>..<fins a>



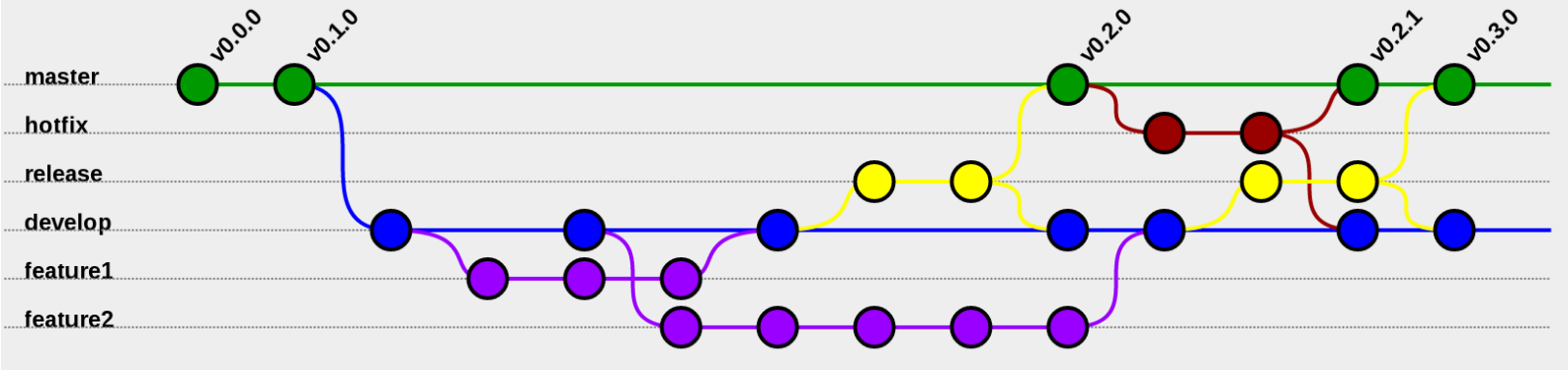

```
$ git log
commit a53bc11430fd2eccfbbf42c507662a14657864b7
Author: Xavier Escudero Sabadell <javier.escudero@upc.edu>
Date:  Wed Aug 23 15:58:30 2017 -0400

$ git log --oneline
a53bc11 Canvis laboratori fitxer markdown
...
```

GIT

TREBALLAR AMB BRANQUES

GIT
Branques

- Branques
  - Espais de treball independents dins d'un repositori
  - Motius:
    - Versió individual d'un client
    - Fase de desenvolupament: prototipus, beta, stable,…
    - Aillament desenvolupament d'una feature o cerca d'un bug complex



http://nurelm.com/our-workflow-git-flow

GIT

Creació i llistat de branques

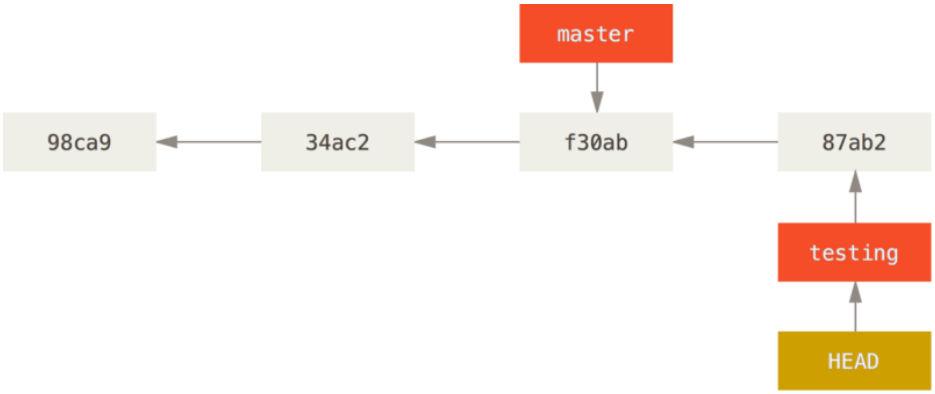
- `git branch nom`
  - Creació. Per defecte s'inicia en el punt que estem
  - No canvia el directori de treball per usar la nova branca (cal checkout)
- `git branch`
  - Llistat de branques
  - * : Branca que tenim checked out al directori de treball
  - Flags
    - -r: Veure branques remotes
    - -a: Veure branques tòpic i remotes
- `git show-branch`
  - Llistat de commits (ordre cronològic invers)
- `git branch –d nom`
  - Esborrat de la branca

```
$ git branch
bug/id-1
dev
* master
```

GIT

Checking out de les branques

- `git checkout <nombranca> [-b][-f]`
  - Canviar de branca. El nostre directori de treball reflecteix l'estat de la branca. **NO perdem els fitxers que teníem abans del checkout**.
  - Flags:
    - -b: Crear i canviar de branca
    - -f: Eliminem els canvis locals!
  - Si els fitxers de la branca actual estan unstaged no es toquen
  - Si tenim modificacions en un fitxer que no es correspon amb el mateix fitxer a la nova branca, no es pot fer l'operació. Usar `git checkout –m <nombranca>`



GIT

Combinar branques

- `git merge <nombranca>`
  - Cal investigar habitualment conflictes entre les branques
  - Usar `git diff`
  - Si volem incloure-ho al nostre master: git checkout master + git merge <nombranca>
  - Una vegada editat el fitxer per resoldre els conflictes fer add i commit del fitxer
  - Hi ha moltes estratègies de merge

Ull amb els merges!

```
$ cat noufitxer
Un títol
<<<<<<< novabranca:noufitxer
Un canvi
=======
Alguna cosa més
>>>>>>> local:noufitxer
```

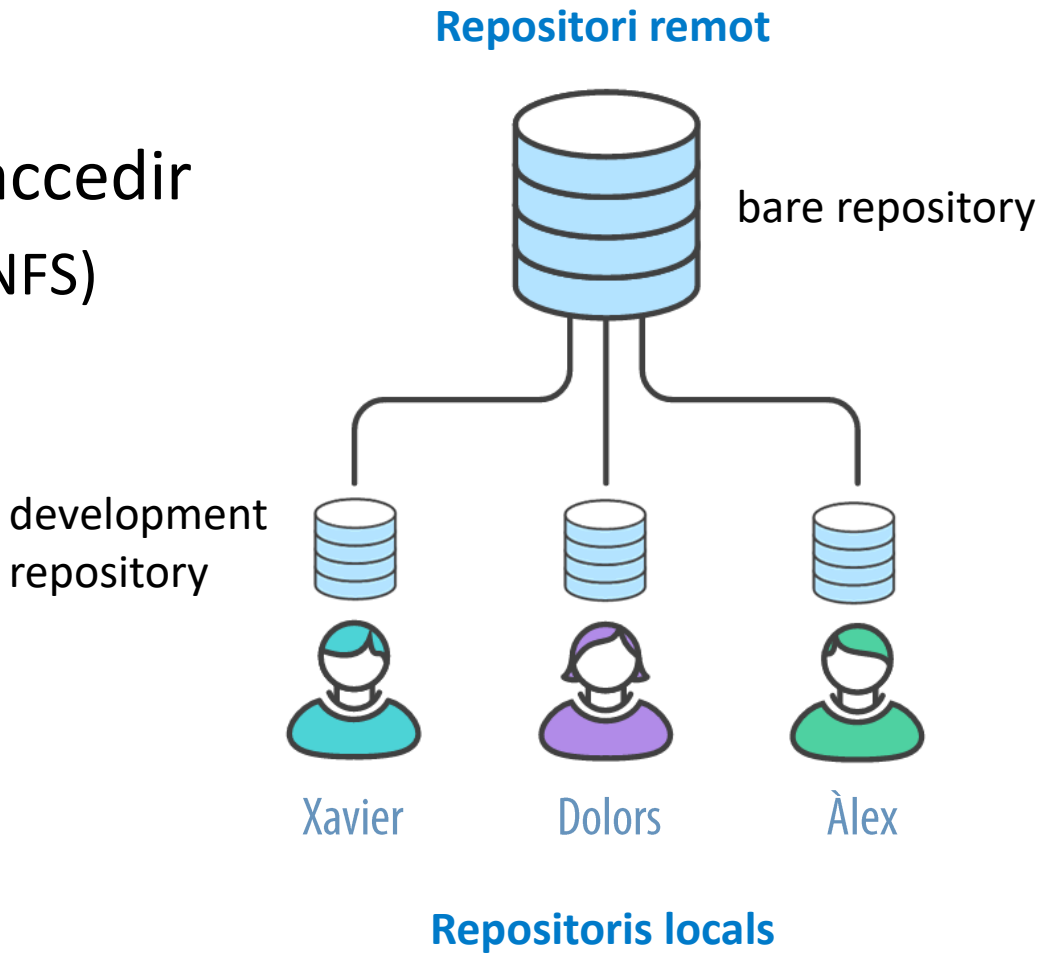
GIT

COL·LABORACIÓ

GIT

Col·laboració

- Repositori comú
- 4 protocols per accedir
  - Local (exemple: NFS)
  - HTTP
  - SSH
  - Git



GIT – Col·laboració

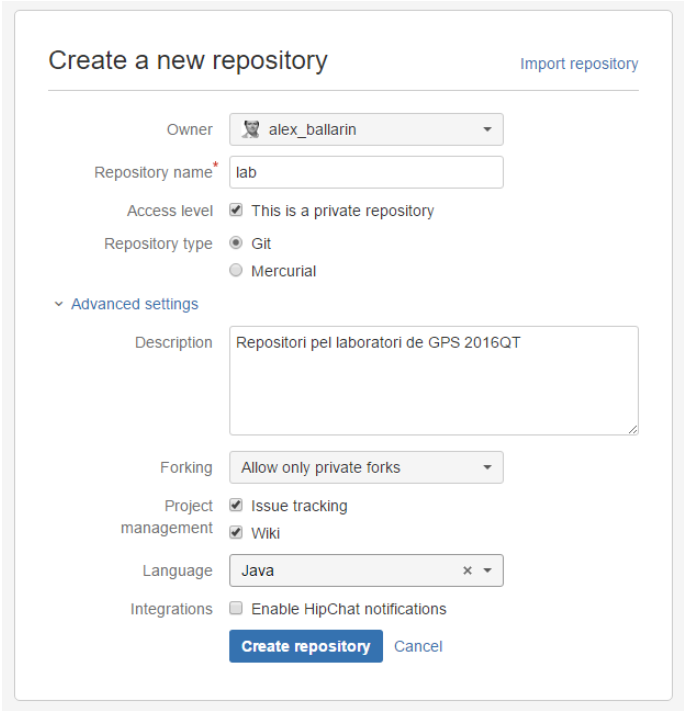
Repositoris remots

- Proporcionen una interfície web per a la gestió dels repositoris compartits
  - A l'assignatura GPS usarem bitbucket
  - També permeten la col·laboració (wiki, seguiment de tasques, etc.)

Atlassian Bitbucket

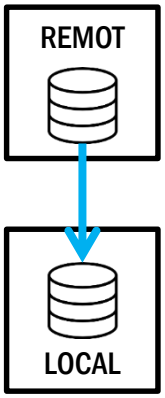
GitHub

GitLab



GIT – Col·laboració

Clonació d'un repositori remot

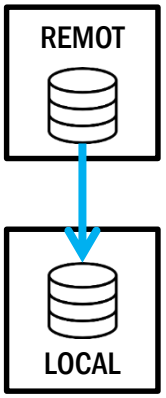


- `git clone`
  - Clona un repositori existent (normalment remot) a una màquina local

```
$ cd /lab1
$ git clone https://bitbucket.fib.upc.edu/lab1
```

GIT – Col·laboració

Creació àlies a repositori remot

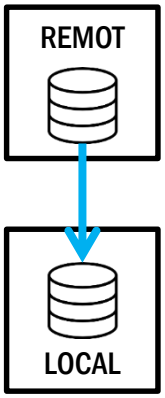


- `git remote add <alias> <url>`
  - Gestiona àlies de repositoris
  - Quan clonem repositori amb git clone, es crea un àlies 'origin'

```
$ git remote add lab1
https://bitbucket.fib.upc.edu/lab1
$ git remote -v
origin https://bitbucket.fib.upc.edu/lab1 (fetch)
origin https://bitbucket.fib.upc.edu/lab1 (push)
```

- `git show-ref`
- `git ls-remote`
  - Hi ha alguna actualització?

GIT – Col·laboració
Importació de commits

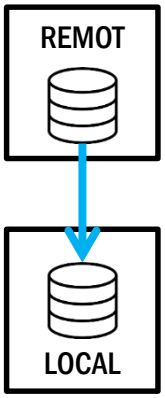


- `git fetch <alias> <branca>`
  - Importa els commits d'un repositori remot (i branca)
  - Aquestes branques remotes són de "només lectura" (no modifica el directori de treball)
  - Podem integrar els canvis a una branca local (p.e. master) amb git merge

```
$ git fetch lab_gps branca1
From https://bitbucket.fib.upc.edu/lab1
 * branch            branca1    -> FETCH_HEAD
 * [new branch]      branca1    -> lab_gps/branca1

$ git branch -r
  lab_gps/branca1
  lab_gps/master
```

GIT – Col·laboració

Importació de commits

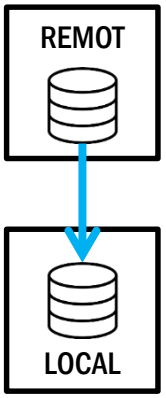


- `git pull <alias>`
  - `git fetch + git merge`
  - `git pull --rebase` per evitar crear commits addicionals amb el "merge"

```
$ git pull --rebase origin
...
From https://bitbucket.fib.upc.edu/lab1
   055c949..845eeff  master     -> origin/master
```

GIT – Col·laboració

Exportant commits al repositori



- `git push <nom-remot> <branca>`
  - Envia els canvis a un repositori remot
  - L'alternativa es demanar un `pull request` al propietari del repositori remot

```
$ git log –oneline
055c949 add fitxer4.txt
0abca67 add fitxer3
751dce1 add .gitignore
2c6bbe3 add fitxer1.txt i fitxer2.txt

$ git push origin master
Total 3 (delta 1), reused 0 (delta 0)
To https://bitbucket.org/itnove/lab_gps
   0abca67..055c949  master -> master
```

GIT

Corol·lari