



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

Introducció a GIT, Github i Markdown

Gestió de projectes software (GPS)

Curs 2016-17, Quadrimestre Tardor

- **Creeu un repositori a Bitbucket.org**
 - » Treballareu en els equips de 6 o 7 membres que m'hagueu comunicat (us donaré un número d'equip)
 - » Us heu de donar d'alta a Bitbucket.org amb el vostre correu de la FIB
 - » Un membre del grup crearà el repositori buit amb el nom **fib_gps_equip_N**
 - » Tots els membres del grup us clonareu aquest repositori
- **Doneu-me accés com administrador al repositori (alex@itnove.com)**
- **Practiqueu les comandes que anireu veient a l'enunciat**
 - » No és obligatori que lliureu el resultat de totes les comandes, però si és recomanable que exploreu bé GIT perquè l'usarem a la resta de laboratoris
 - » Podeu trobar més informació a la web d'Atlassian → <https://www.atlassian.com/git>
- **Practiqueu també com escriure fitxers Markdown (darrer apartat)**
- **Deixeu tots els resultats al repositori del vostre grup**



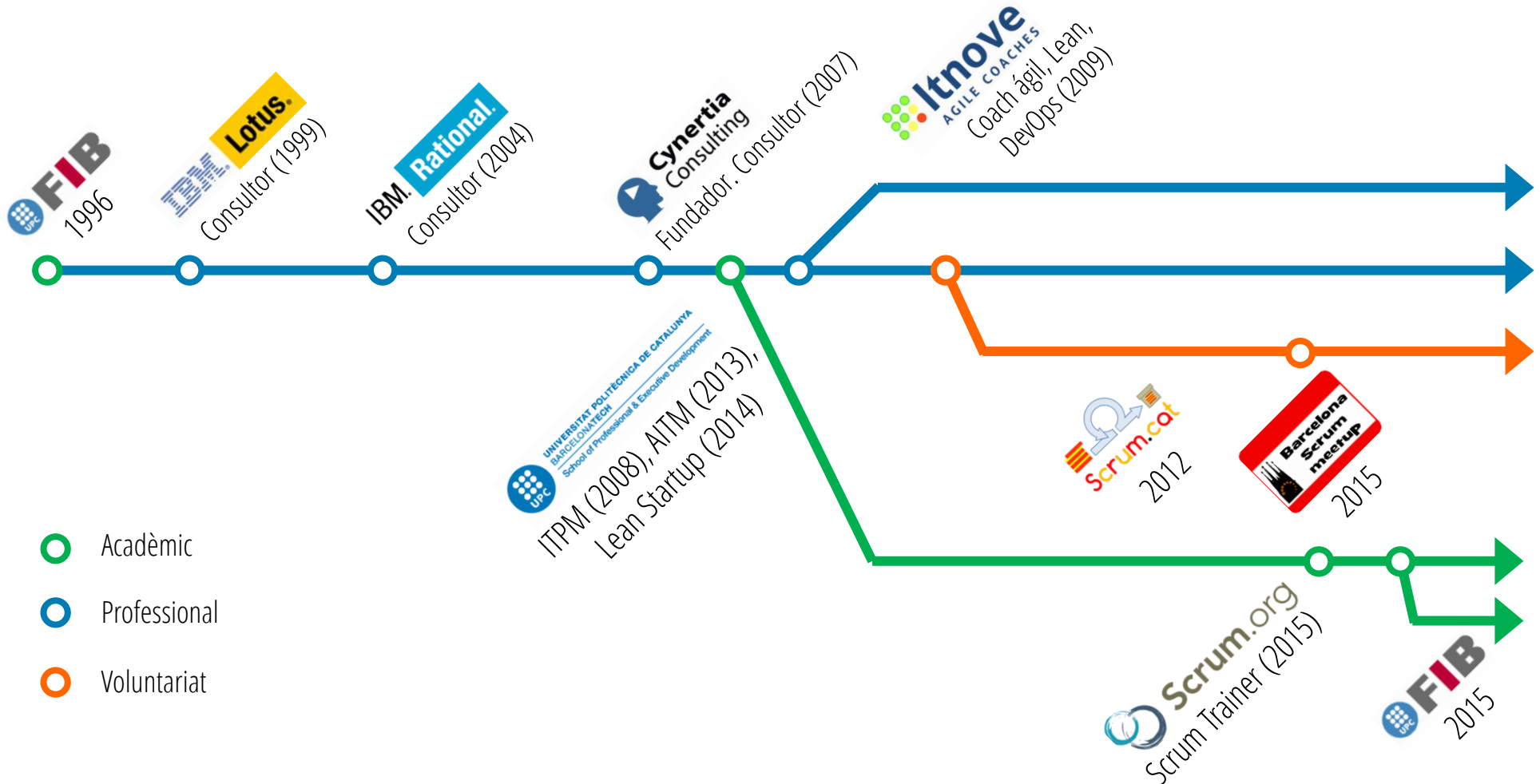
Alex Ballarin Latre

🏠 ESSI-UPC (Omega 108)

@ aballarin@essi.upc.edu

🌐 linkedin.com/in/alexballarin

🐦 twitter.com/AlexBallarin76



Continguts

- 1 Conceptes de controls de versions i GIT
- 2 GIT: Gestió de repositoris
- 3 GIT: Versionat de fitxers
- 4 GIT: Workflow centralitzat
- 5 GIT: Altres workflows
- 6 Llenguatge Markdown

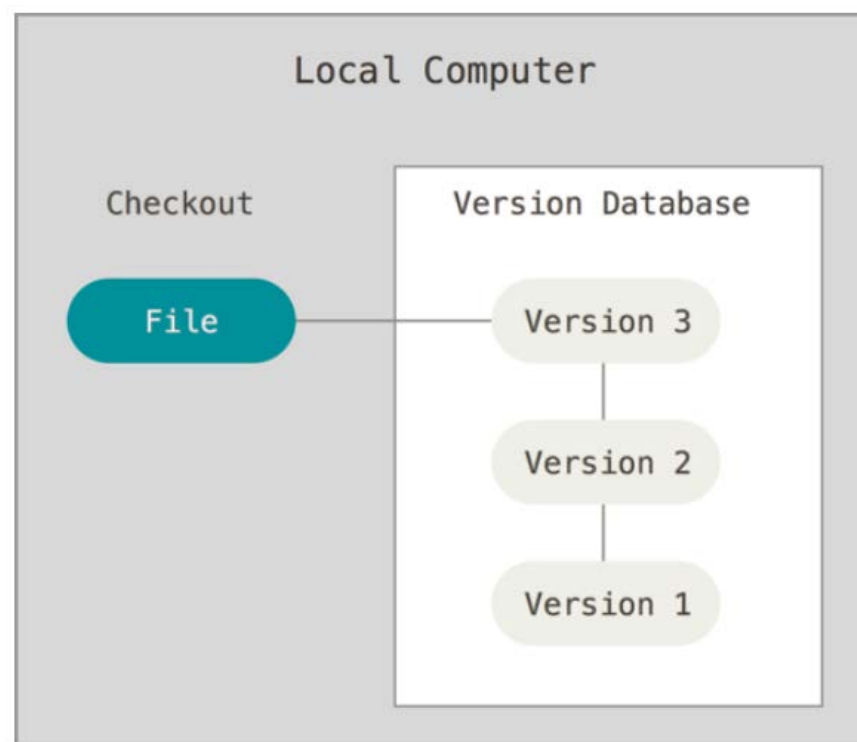
Processos de gestió de configuració i objectius d'un control de versions

- Processos de gestió de configuració
 - » Planificació i gestió
 - » Identificació de la configuració
 - » Control de la configuració
 - » Informació de l'estat de la configuració
 - » Auditoria i verificació de la gestió de la configuració
- Línia base
 - » Estat controlat d'un conjunt de fitxers relacionats (p.e. "versió d'un sistema")
- Objectius del control de la configuració
 - » Registrar els canvis als elements de configuració (p.e. fitxers)
 - » Revertir l'estat actual d'un fitxer (o d'una línia base) a un estat anterior registrat
 - » Comparar els canvis (p.e. continguts)
 - » Revisar l'història dels canvis (p.e. motius o autors)
 - » En general, garantir un desenvolupament ordenat i eficient (per la informació del context) i segur (sense perdre informació)

Tipus de controls de versions (1)

■ Local

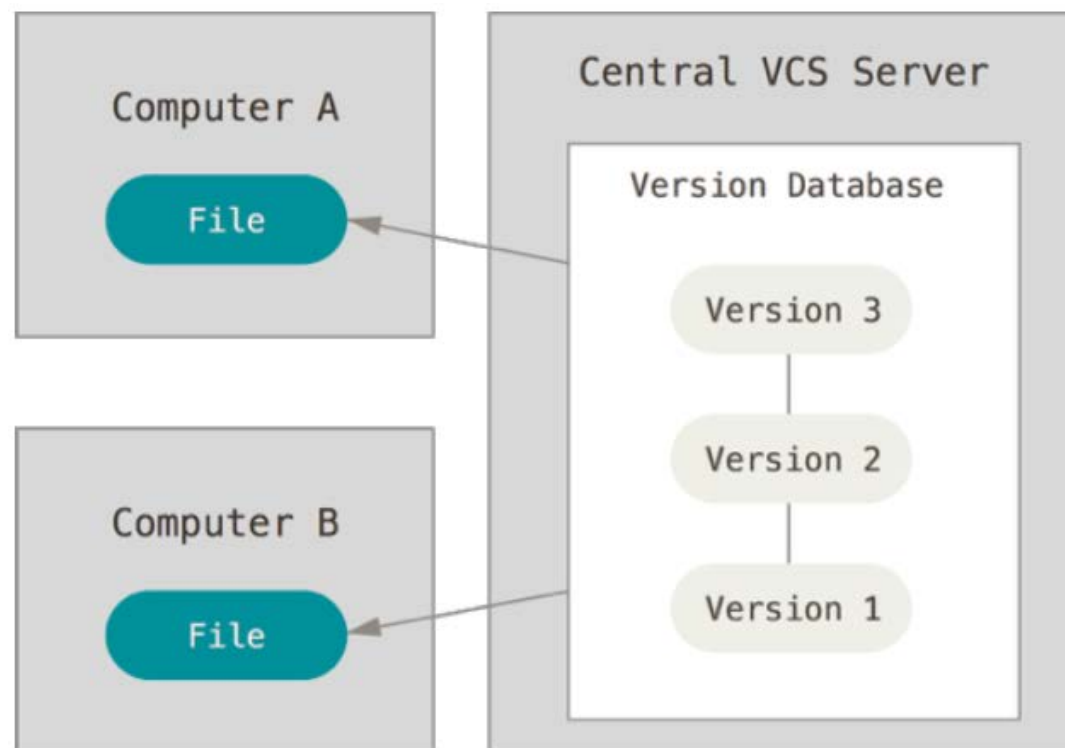
- » Els fitxers i la BD de versions estan a una màquina local (no permet compartir)



Tipus de controls de versions (2)

■ Centralitzat

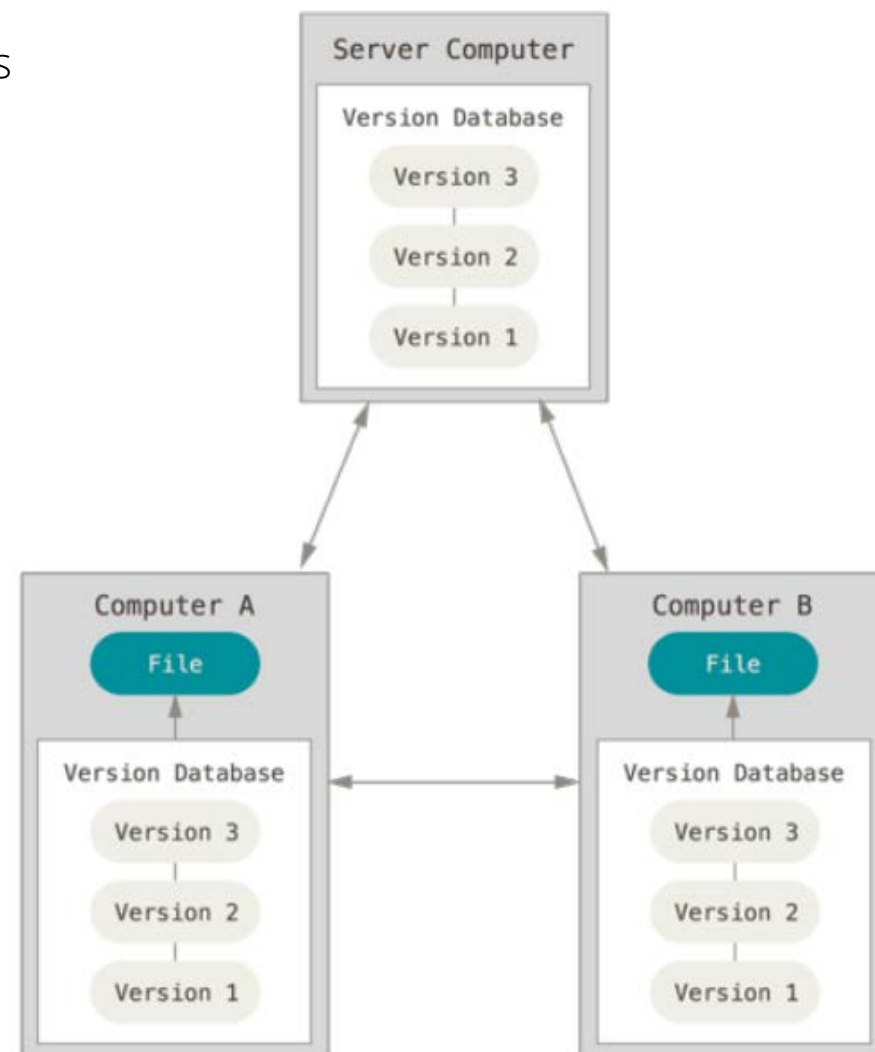
- » La BD de versions (i potser també els fitxers) estan a un servidor centralitzat
- » Exemples: CVS, SourceSafe i Team Foundation Server, Subversion



Tipus de controls de versions (3)



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

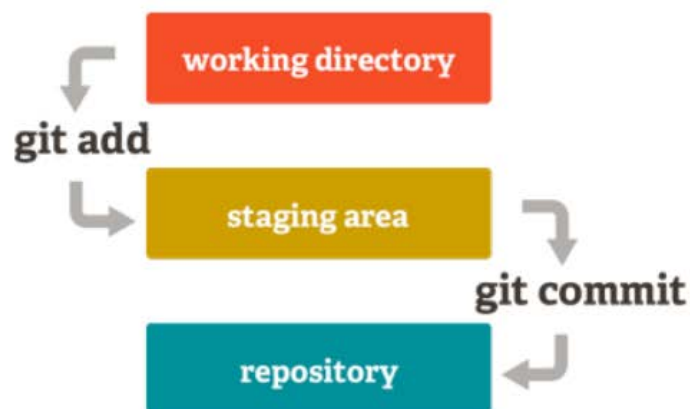


Conceptes de GIT (1)



■ Espais

- » Working directory: fitxers locals (del programador)
- » Staging area: espai temporal per crear línies base endreçades (commits)
- » Repository: espai on hi ha els fitxers (empaquetats en commits) i la seva història

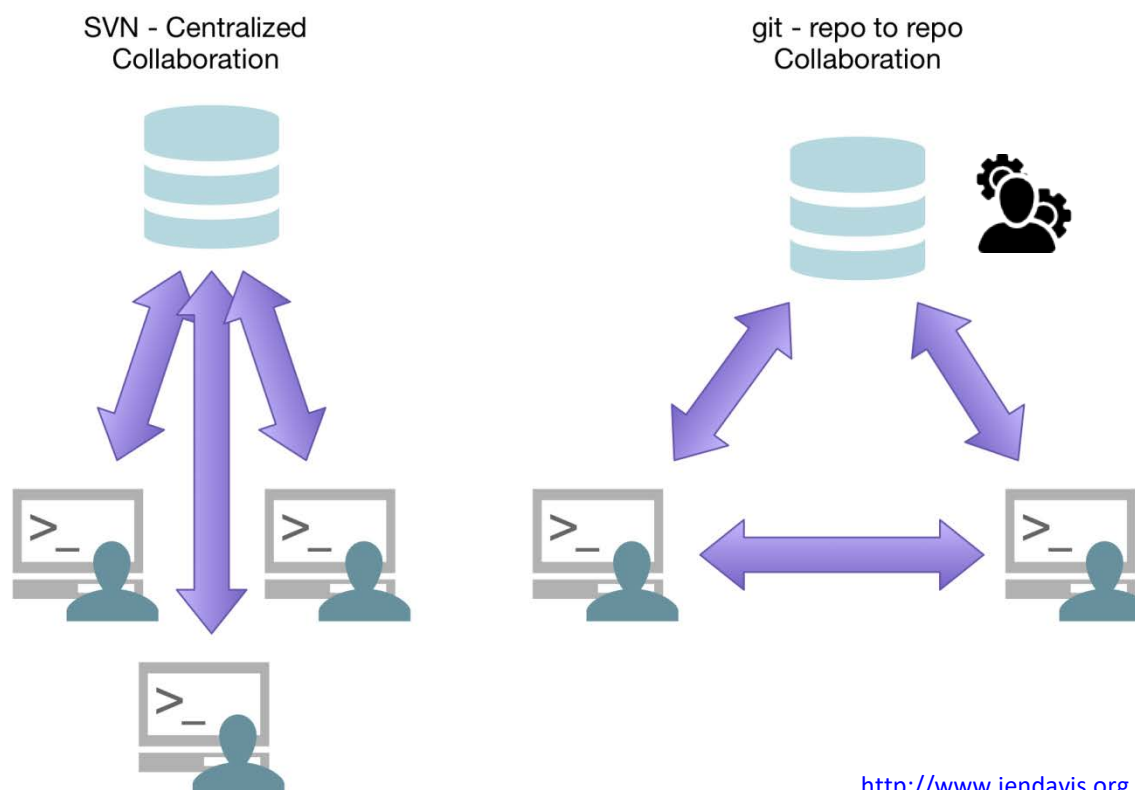


Conceptes de GIT (2)



■ Arquitectura de repositoris

- » Client / Servidor (model habitual): s'utilitza un repositori concret per a centralitzar els canvis d'altres repositoris
- » Acostuma a requerir un rol l'administració d'aquest "repositori central"

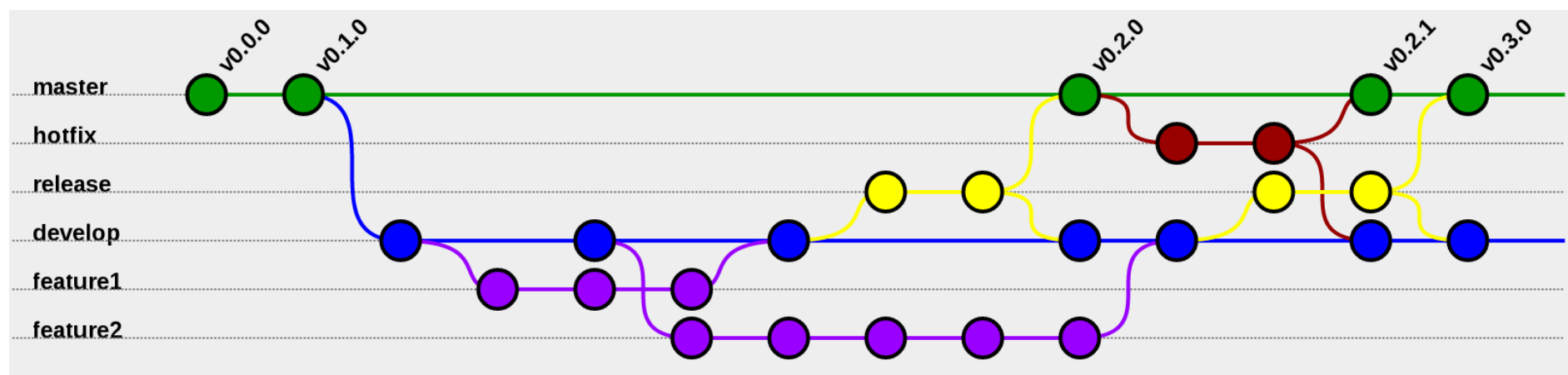


<http://www.jendavis.org>

Conceptes de GIT (3)



- Branques
 - » Espais de treball independents dins d'un repositori
- Fluxe de treball (workflow)
 - » Model d'ús de les branques a una organització



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

Creació del repositori local

■ Git init

- » Crea un repositori local (a una carpeta amb o sense fitxers)
- » Es crea una subcarpeta **.git** amb les metadades del repositori



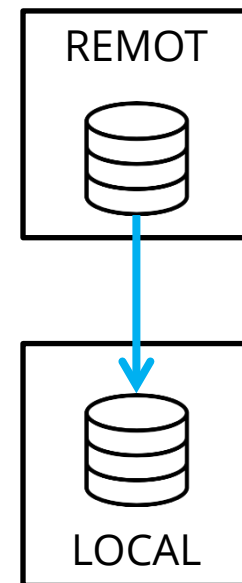
```
$ cd /c/fib/lab_gps  
alex@Alex-toshiba MINGW64 /c/fib/lab_gps  
  
$ git init  
Initialized empty Git repository in C:/fib/lab_gps/.git/  
alex@Alex-toshiba MINGW64 /c/fib/lab_gps (master)
```

Clonació d'un repositori remot

■ Git clone

- » Clona un repositori existent (normalment remot) a una màquina local
- » Es pot accedir al repositori remot via **SSH** o **HTTPS**

```
$ cd /c/fib/  
$ git clone https://bitbucket.org/itnove/lab_gps  
Cloning into 'lab_gps'...  
remote: Counting objects: 9, done.  
remote: Compressing objects: 100% (6/6), done.  
remote: Total 9 (delta 2), reused 0 (delta 0)  
Unpacking objects: 100% (9/9), done.  
Checking connectivity... done.
```



Configuració de GIT

■ Git config

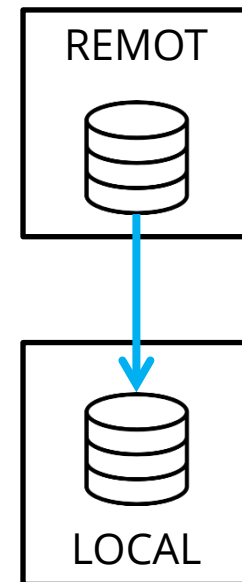
» Configura GIT (nivell del sistema, de l'usuari o del repositori actual)

- × Sistema → `$(prefix)/etc/gitconfig`
- × Usuari → `~/.gitconfig`
- × Repositori → `<repo>/.git/config`

```
$ git config --global user.name itnove  
$ git config --global user.email alex@itnove.com  
$ git config --system core.editor vim  
$ git config --global --edit
```

```
[user]  
name = itnove  
email = alex@itnove.com  
[core]  
editor = vim
```

} Editat a VIM



Coneixer l'estat dels canvis

■ Git status

- » Ens diu l'estat del repositori (working directory, staging i repositori)
 - × Untracked: fitxers que només estan al working directory (no estan controlats)
 - × Changes to be committed: fitxers que estan a la zona staging (estan controlats pero no confirmats)

```
$ echo 'contingut fitxer 1' > fitxer1.txt  
$ git status
```

On branch master

Your branch is up-to-date with 'origin/master'.

Untracked files: 

(use "git add <file>..." to include in what will be committed)

fitxer1.txt

nothing added to commit but untracked files present (use "git add" to track)

Atenció a cometes simples (echo),
dobles (commit) i als caràcters que
genera el copy & paste!

Afegir fitxers a la zona temporal (staging)

■ Git add

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

```
$ echo 'contingut fitxer 2' > fitxer2.txt
```

```
$ git add fitxer2.txt
```

```
$ git status
```

On branch master

Your branch is up-to-date with 'origin/master'.

Changes to be committed: ←

(use "git reset HEAD <file>..." to unstage)

new file: fitxer2.txt

Untracked files: ←

(use "git add <file>..." to include in what will be committed)

fitxer1.txt

Confirmar un conjunt de canvis de la zona temporal

■ Git commit

- » Envia tots els fitxers de la zona staging al repositori empaquetats en un commit
- » Flags
 - × -a → envia tots els canvis (sinó és un procés interactiu)
 - × -m → permet incloure un missatge descriptiu (**molt recomanat**)

```
$ git add .  
$ git commit -m "add fitxer1.txt i fitxer2.txt"  
[master 9976b89] missatge descriptiu del commit  
2 files changed, 2 insertions(+)  
create mode 100644 fitxer1.txt  
create mode 100644 fitxer2.txt
```

Inclou ID (9976b89) i la descripció del commit

```
$ git status  
On branch master  
Your branch is ahead of 'origin/master' by 1 commit.
```

El nostre repositori (branca master) està avançat respecte el repositori original

Ignorar fitxers a GIT

■ Fitxer **.gitignore**

- » Hi ha fitxers que normalment no interessa incloure a GIT (p.e. ninaris)
- » Aquest fitxer inclou la llista de regles que GIT usa per ignorar fitxers
- » Els fitxers ignorats no apareixen a **git status** ni es consideren a **git add** .

```
$ echo 'fitxer binari fals' > fitxer.exe
```

```
$ git status
```

```
On branch master
```

```
Untracked files:
```

```
fitxer.exe
```

```
$ echo '*.exe' > .gitignore
```

```
$ git add .gitignore
```

```
$ git commit -m "add .gitignore"
```

```
$ git status
```

```
On branch master
```

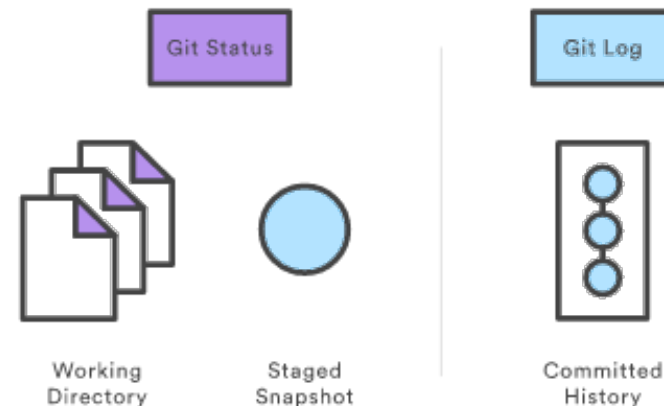
```
nothing to commit, working directory clean
```

.gitignore s'ha d'afegir i 'commitejar'
com els altres fitxers !!!

Accés a l'històric de canvis al repositori

■ **Git log <file>**

- » Llista els canvis al repositori (commits)
- » Flags habituals
 - × --oneline → Mostra l'històric en format resumit
 - × -p → Mostra els patches (canvis) entre commits



```
$ git log --oneline
751dce1 add .gitignore
2c6bbe3 add fitxer1.txt i fitxer2.txt

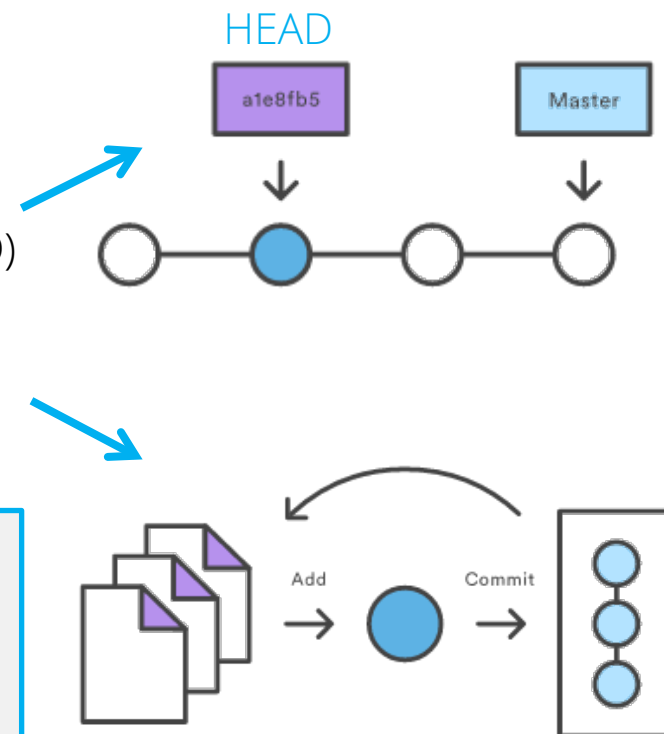
$ git log -p
commit 9153b6831ff2c1c23b8eeee6513dd23af1bc77dc
...
diff --git a/fitxer2.txt b/fitxer2.txt
...
contingut fitxer 2
+nou contingut a un altre commit
```

Ens mostra les diferències dels commits

Accés al contingut de commits anteriors

■ **Git checkout <commit> <file>**

- » Tornar els fitxers del working directory a una versió anterior
- » No es perd la versió actual (estem “veient” el punter al HEAD)
 - × Estat “**detached HEAD**”
- » Podem “revertir” la versió actual d’alguns fitxers a una antiga
 - × És més pràctic amb la comanda **Git revert**



<https://www.atlassian.com/git/tutorials>

```
$ git log --oneline
0abca67 add fitxer3
751dce1 add .gitignore
2c6bbe3 add fitxer1.txt i fitxer2.txt

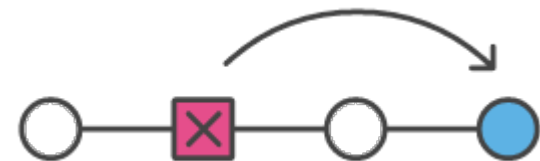
$ ll
total 4 ←
...

$ git checkout 9976b89
$ ll
total 3 ←
...
```

Revertint l'estat dels fitxers a un commit anterior

■ Git revert <commit>

- » Crea un nou commit amb els continguts d'un anterior
- » Es pot tornar al commit "actual" en qualsevol moment
- » Es similar a **git reset** però podent desfer el canvi del HEAD



```
$ git checkout master ← Tornem al darrer commit de la branca
```

```
$ rm fitxer3.txt
```

```
$ git commit -a -m "eliminem fitxer3 per error"
```

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

```
2ad0d5d eliminem fitxer3 per error
```

```
0abca67 add fitxer3
```

```
751dce1 add .gitignore
```

```
$ git revert 751dce1 ← Ens mostra VIM per editar el text del commit
```

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

```
125d0a3 Revert "add .gitignore"
```

```
2ad0d5d eliminem fitxer3 per error ← Podriem recuperar el darrer commit
```

```
0abca67 add fitxer3
```

```
751dce1 add .gitignore
```

Eliminant els carrers commits

■ Git reset <commit>

- » Torna un fitxer (o tots) de la [zona de staging](#) a un commit anterior
- » Només afecta el [working directory](#) si afegim **--hard**
- » Permet reorganitzar el contingut del proper commit a la zona de staging



```
$ git log --oneline
125d0a3 Revert "add .gitignore"
2ad0d5d eliminem fitxer3 per error
0abca67 add fitxer3
751dce1 add .gitignore
2c6bbe3 add fitxer1.txt i fitxer2.txt

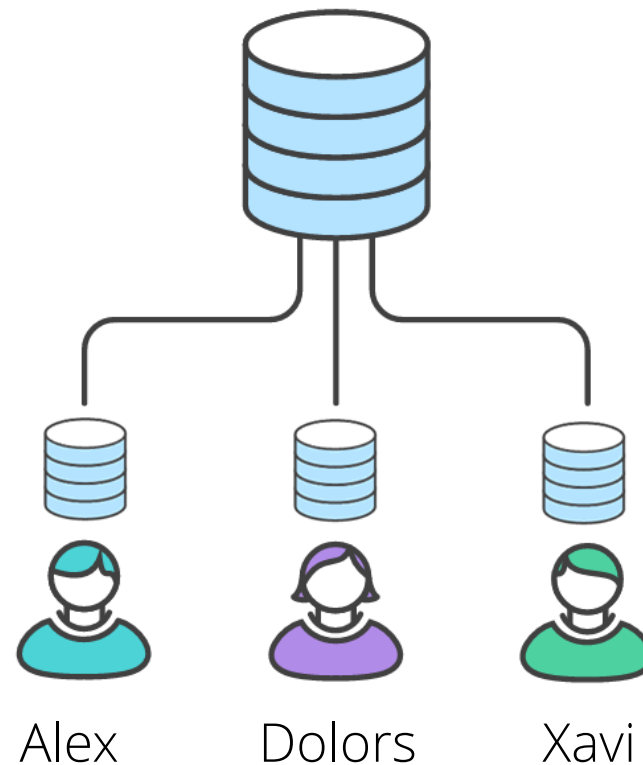
$ git reset 0abca67 --hard
$ git log --oneline
0abca67 add fitxer3
751dce1 add .gitignore
2c6bbe3 add fitxer1.txt i fitxer2.txt
```



L'opció **--hard** ens pot fer perdre fitxers del repositori i canvia l'històric de canvis, s'ha d'anar en compte i no fer-la servir mai a repositoris centrals.

Repositori central mestre

- Els desenvolupadors poden treballar en aïllament
- No s'usen branques, els canvis es sincronitzen sovint amb el repositori central



Gestors de repositori

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



The screenshot shows the 'Create a new repository' form in Bitbucket. The form is titled 'Create a new repository' and has a link for 'Import repository' in the top right corner. The form fields are as follows:


- Owner:** A dropdown menu showing 'alex_ballarin'.
- Repository name:** A text input field containing 'lab'.
- Access level:** A checkbox labeled 'This is a private repository' which is checked.
- Repository type:** Radio buttons for 'Git' (selected) and 'Mercurial'.
- Advanced settings:** A section with a dropdown arrow, containing:
 - Description:** A text area containing 'Repositori pel laboratori de GPS 2016QT'.
 - Forking:** A dropdown menu showing 'Allow only private forks'.
 - Project management:** Checkboxes for 'Issue tracking' and 'Wiki', both of which are checked.
 - Language:** A dropdown menu showing 'Java'.
 - Integrations:** A checkbox labeled 'Enable HipChat notifications' which is unchecked.

At the bottom of the form are two buttons: 'Create repository' (in blue) and 'Cancel'.

Creant àlies a repositoris remots

- **Git remote add <name> <url>**
 - » Gestiona els àlies del repositori actual
 - » No crea connexions estables, és simplement una utilitat

```
$ git remote add lab_gps https://bitbucket.org/alex_ballarin/lab_gps  
$ git remote -v  
origin https://bitbucket.org/alex_ballarin/alex_gps (fetch)  
origin https://bitbucket.org/alex_ballarin/alex_gps (push)
```



En cas de ser un clone, els àlies ja estan creats


Exportant commits cap altres repositoris




■ Git push <remote> <branch>

- » Envia els canvis a un repositori remot
- » L'alternativa es demanar un **pull request** a 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
Total 3 (delta 1), reused 0 (delta 0)
To https://bitbucket.org/itnove/lab_gps
0abca67..055c949 master -> master
```



Author	Commit	Message
 Alex Ballarin	0abca67	add fitxer3
 Alex Ballarin	751dce1	add .gitignore
 Alex Ballarin	2c6bbe3	add fitxer1.txt i fitxer2.txt



Author	Commit	Message
 Alex Ballarin	055c949	add fitxer4.txt
 Alex Ballarin	0abca67	add fitxer3
 Alex Ballarin	751dce1	add .gitignore
 Alex Ballarin	2c6bbe3	add fitxer1.txt i fitxer2.txt

Important commits des d'altres repositoris (1)

■ Git fetch <remote> <branch>

- » Importa els commits d'un repositori remot (o d'una branca concreta)
- » Aquestes branques remotes són de “només lectura”
- » Podriem integrar els canvis a una branca local (p.e. master) amb **git fetch** seguit d'un **git merge** , però podem fer el mateix més fàcilment amb **git pull**

```
$ git fetch lab_gps branca1
```

```
From https://bitbucket.org/alex_ballarin/lab_gps
```

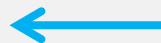
```
* branch          branca1      -> FETCH_HEAD
```

```
* [new branch]     branca1      -> lab_gps/branca1
```

```
$ git branch -r
```

```
lab_gps/branca1
```

```
lab_gps/master
```



Tenim accés a la branca1 remota (repositori lab_gps),
podriem inspeccionar els seus continguts amb
git checkout

Important commits des d'altres repositoris (2)

■ Git pull <remote>

- » Busca la còpia remota de la branca actual i la integra a la còpia local
- » Es pot fer via git **pull --rebase** per evitar crear commits addicionals amb el “merge”

```
$ git pull --rebase origin
```

```
...
```






```
From https://bitbucket.org/itnove/lab_gps
```

```
055c949..845eeff master -> origin/master
```

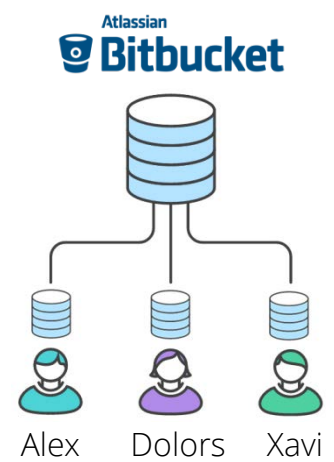
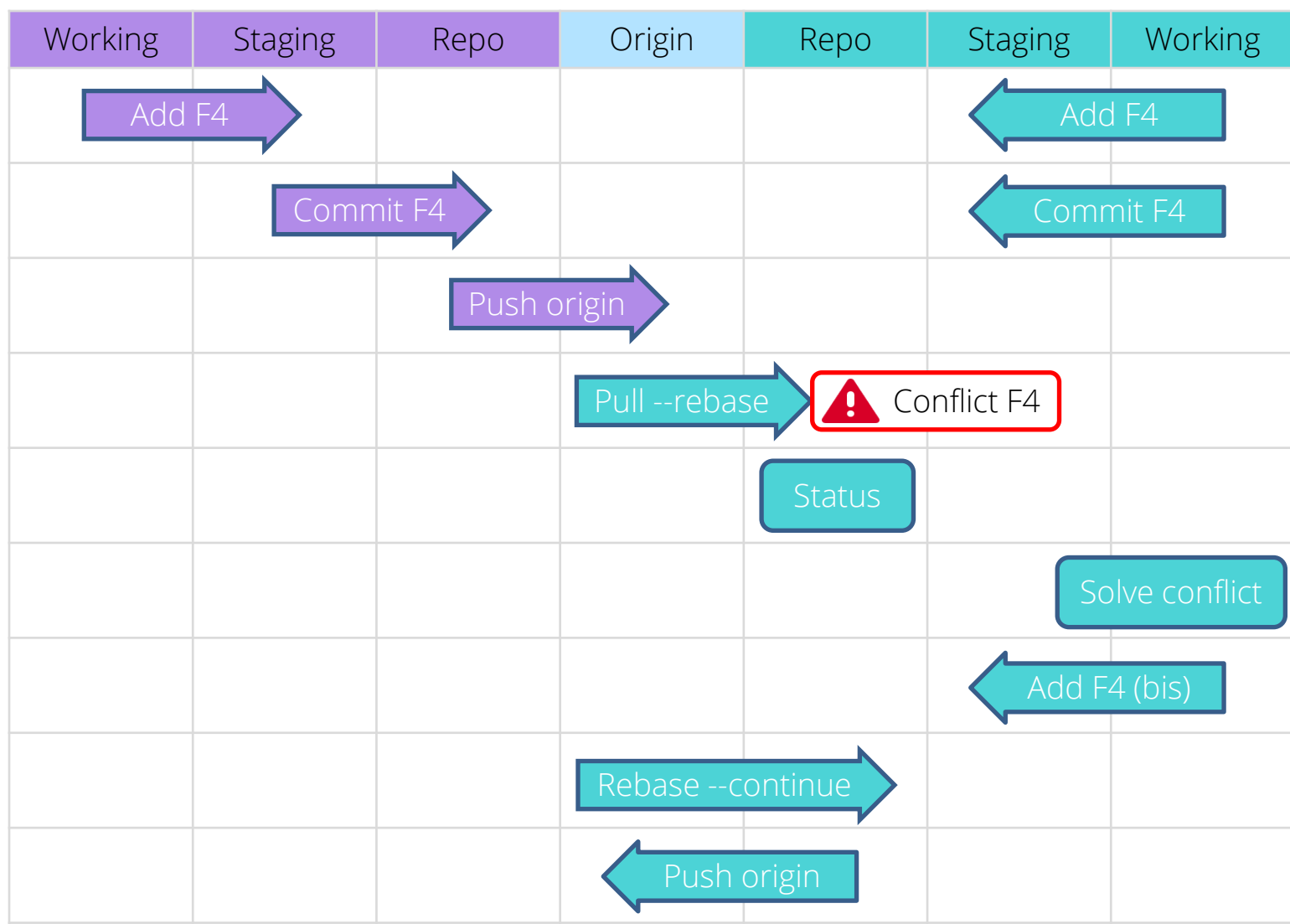
```
$ ll
```

```
alex 197609 25 set. 10 17:22 fitxer.exe*  
alex 197609 19 set. 10 17:20 fitxer1.txt  
alex 197609 19 set. 10 17:20 fitxer2.txt  
alex 197609 19 set. 10 17:52 fitxer3.txt
```

Importem els commits remots
que eliminen el fitxer4 en local

Author	Commit	Message
 Alex Ballarin	845eeff	fitxer4.txt deleted online with Bitbucket
 Alex Ballarin	055c949	add fitxer4.txt
 Alex Ballarin	0abca67	add fitxer3
 Alex Ballarin	751dce1	add .gitignore
 Alex Ballarin	2c6bbe3	add fitxer1.txt i fitxer2.txt

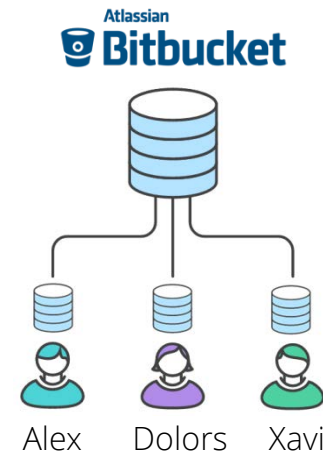
El model més semblant a un control de versions centralitzat



L'exemple anterior en comandes... (1)

```
alex@Alex-toshiba MINGW64 /c/fib/lab_gps (master)
$ echo 'alex crea fitxer4' > fitxer4.txt
$ git add .
$ git commit -m "Alex crea fitxer4.txt"
[master 4c8b3d3] Alex crea fitxer4.txt
```

Aquest commit d'Àlex no
apareixerà al repositori central



```
Dolors@Alex-toshiba MINGW64 /c/fib/lab_gps_dolors/lab_gps (master)
$ echo 'dolors crea fitxer4' > fitxer4.txt
$ git add fitxer4.txt
$ git commit -m "Dolors crea fitxer4.txt"
[master 19589ce] Dolors crea fitxer4.txt

$ git push origin
To https://bitbucket.org/itnove/lab_gps
845eeff..19589ce master -> master
```

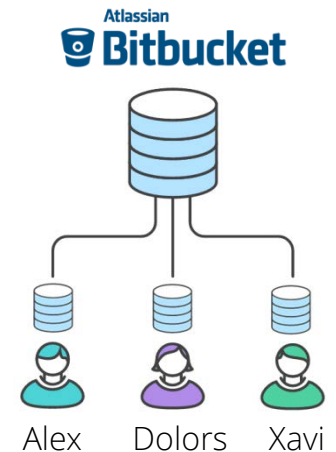
Aquest commit de Dolors si apareixerà

L'exemple anterior en comandes... (2)

```
alex@Alex-toshiba MINGW64 /c/fib/lab_gps (master)
$ git pull origin --rebase
From https://bitbucket.org/itnove/lab_gps
   845eeff..19589ce  master    -> origin/master
Applying: Alex crea fitxer4.txt
CONFLICT (add/add): Merge conflict in fitxer4.txt
error: Failed to merge in the changes.
Patch failed at 0001 Alex crea fitxer4.txt
The copy of the patch that failed is found in: .git/rebase-
apply/patch$ echo 'alex crea fitxer4' > fitxer4.txt

$ echo 'canvis alex i dolors integrats' > fitxer4.txt
$ git add .
$ git rebase --continue
Applying: Alex crea fitxer4.txt
```

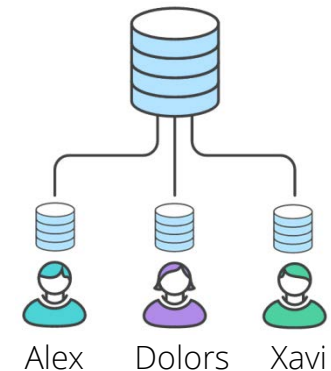
Aquest rebase d'Àlex farà
un commit (c35ffe7) que sí
anirà al repositori central



L'exemple anterior en comandes... (3)

```
alex@Alex-toshiba MINGW64 /c/fib/lab_gps (master)
$ git log --oneline
c35ffe7 Alex crea fitxer4.txt
19589ce Dolors crea fitxer4.txt
845eeff fitxer4.txt deleted online with Bitbucket
055c949 add fitxer4.txt
0abca67 add fitxer3
751dce1 add .gitignore
2c6bbe3 add fitxer1.txt i fitxer2.txt

$ git push origin
To https://bitbucket.org/itnove/lab_gps
19589ce..c35ffe7 master -> master
```

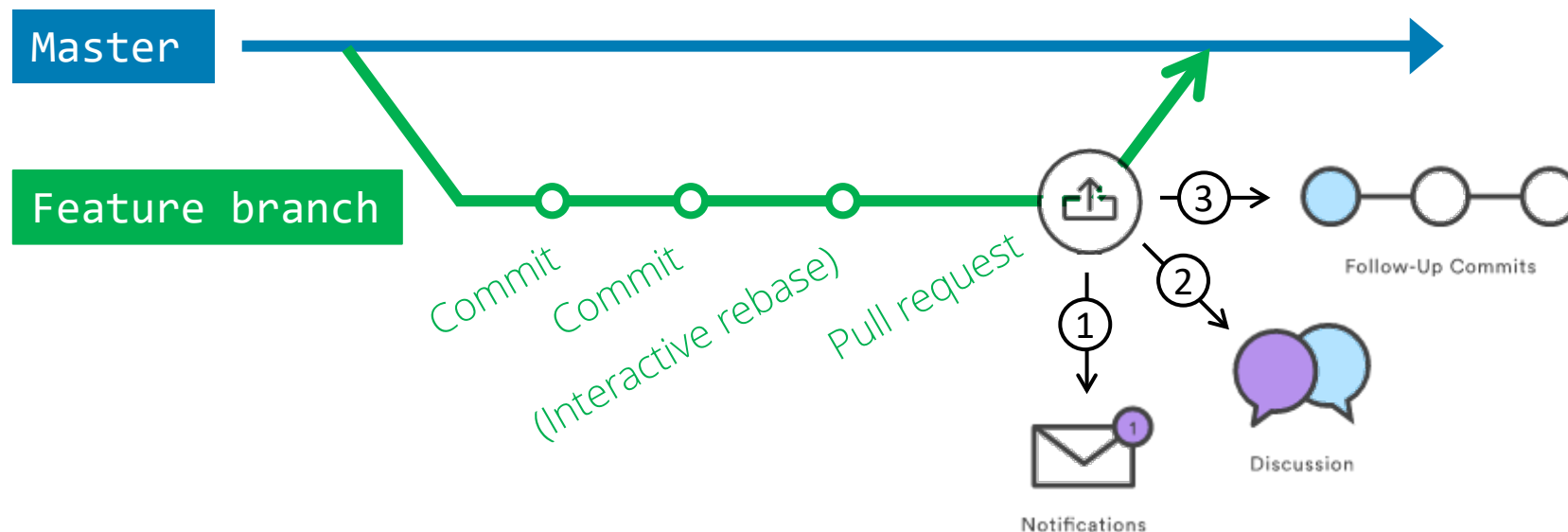


	Author	Commit	Message
•	Alex Ballarin	c35ffe7	Alex crea fitxer4.txt
•	Alex Ballarin	19589ce	Dolors crea fitxer4.txt
•	Alex Ballarin	845eeff	fitxer4.txt deleted online with Bitbucket
•	Alex Ballarin	055c949	add fitxer4.txt
•	Alex Ballarin	0abca67	add fitxer3
•	Alex Ballarin	751dce1	add .gitignore
•	Alex Ballarin	2c6bbe3	add fitxer1.txt i fitxer2.txt

Workflows amb branques de funcionalitat (feature branch)

■ Pull request (des de Bitbucket)

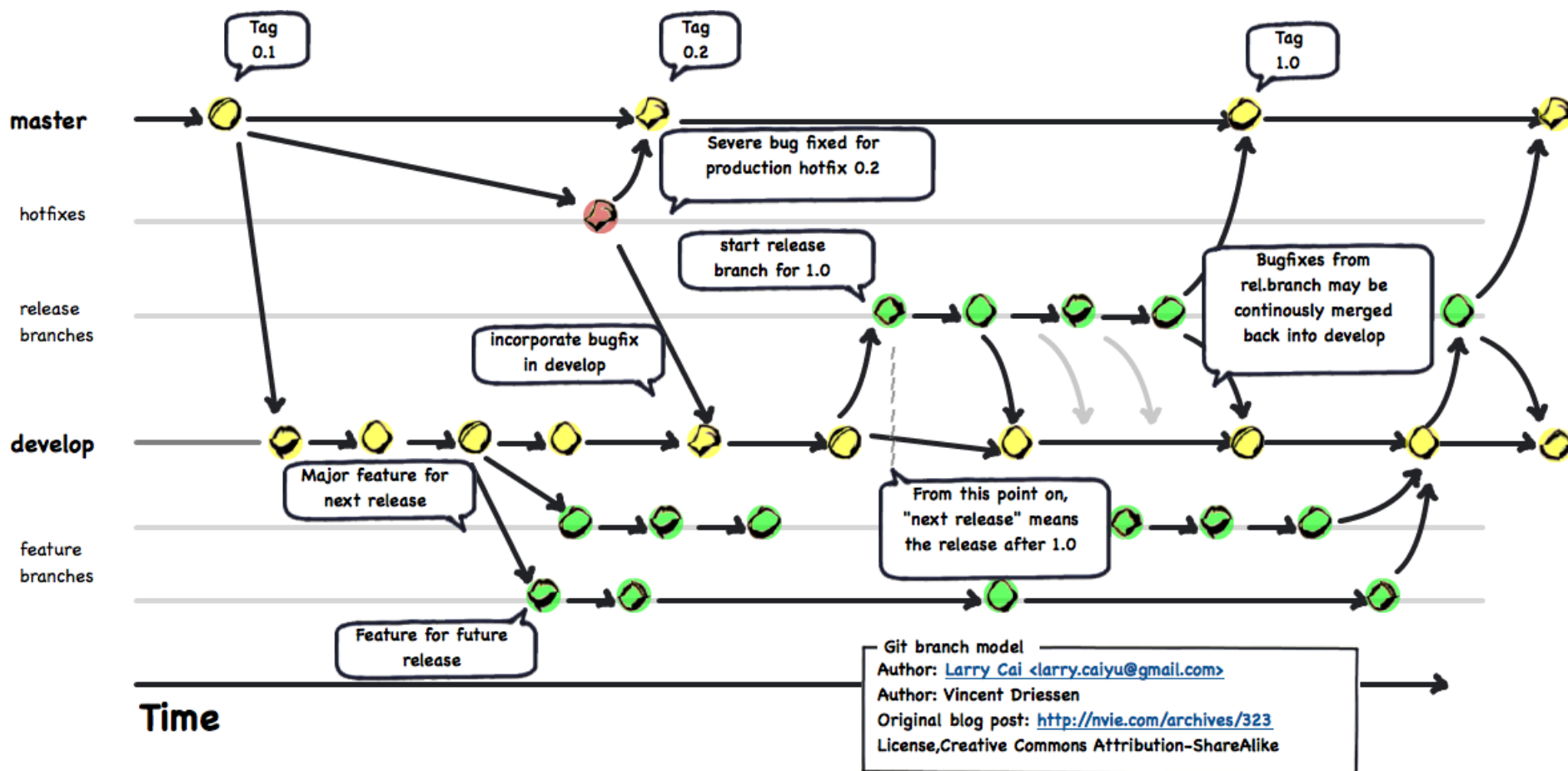
- » Els desenvolupadors creen branques per desenvolupar una funcionalitat en aïllament i per tant *és un model diferent a la integració continua (1)*.
- » Abans de demanar la integració (pull request) al propietari de la branca master, s'acostuma a fer un "interactive rebase" per simplificar la llista de commits que es lliura.
- » El pull request dispara un procés col·laboratiu per revisar el contingut dels commits aportats i eventualment modificar algun aspecte (*follow-up commits*).



(1) https://en.wikipedia.org/wiki/Continuous_integration

Gitflow

- Un model de branques que defineix específicament el paper de cada branca.
 - » És més complexe però també útil per projectes grans



Llenguatge simplificat per escriure HTML

- L'usen eines com Bitbucket o Github, i el farem servir a l'assignatura
- Cloneu el següent repositori i practiqueu amb un exemple bàsic de Markdown
<https://bitbucket.org/jordipradel/gps-markdown>
- Sintàxi de Markdown
<https://github.com/adam-p/markdown-here/wiki/Markdown-Here-Cheatsheet>
- Exemples d'eines i utilitats
 - » Plugin per previsualitzar MD → <https://github.com/adam-p/markdown-here>
 - » Editor Windows → <http://markdownpad.com>
 - » Editor Mac → <http://macdown.uranusjr.com>