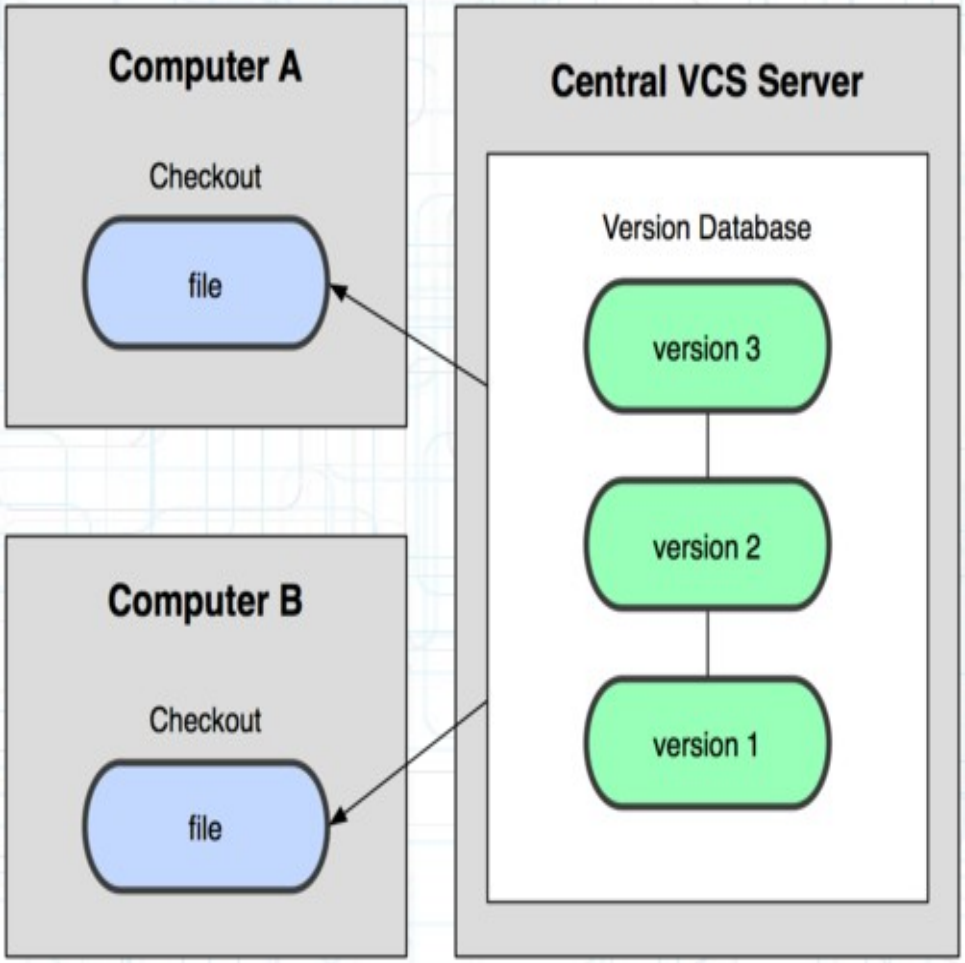
Git: sistema di versionamento file distribuito

- by Daniele Segato
- daniele.segato@gmail.com
- http://natonelbronx.wordpress.com

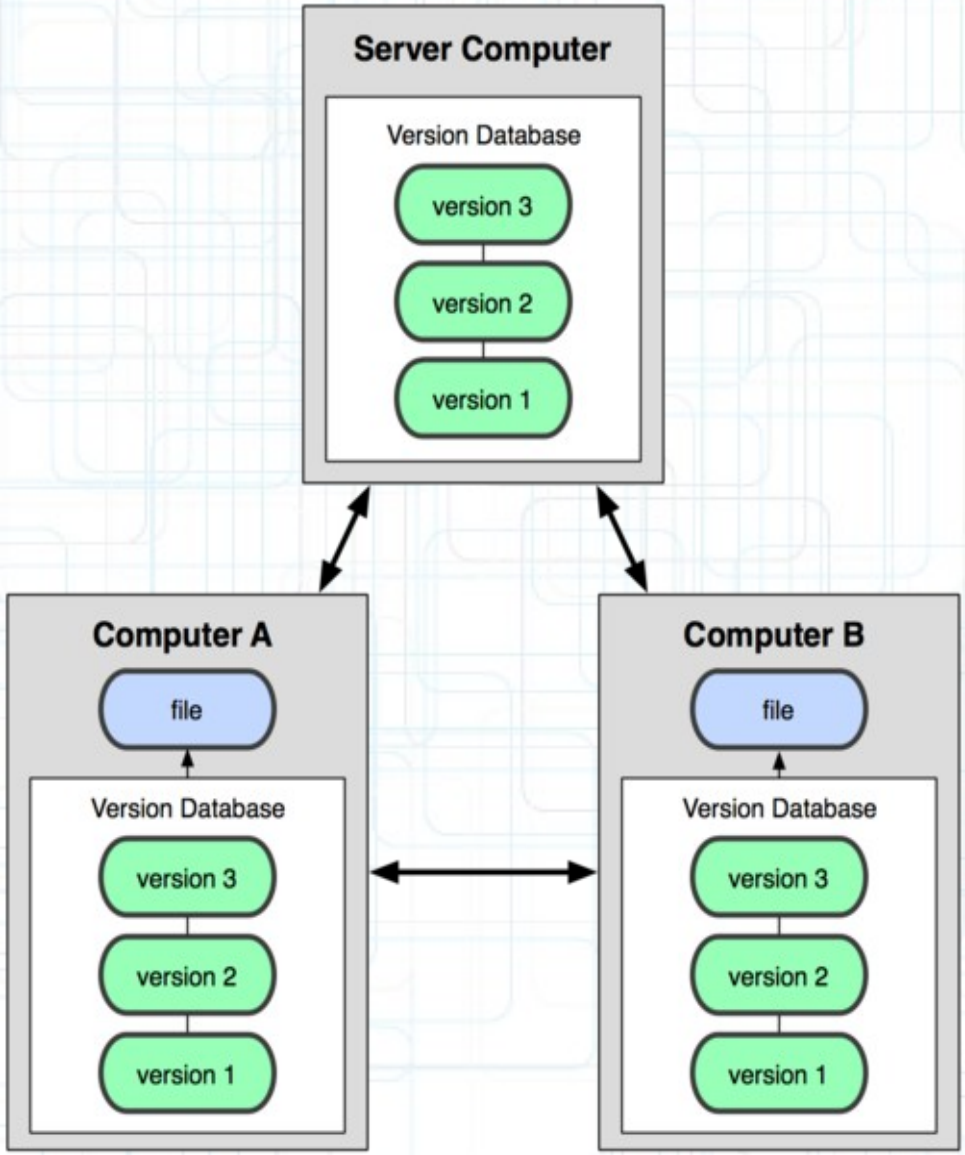
Git permette di gestire in modo efficiente il software e la collaborazione.
Branching e merging non saranno più un problema.

Versionamento Centralizzato



- Un solo server centrale
- Ogni utente scarica una versione
- Sviluppo lineare
- Subversion (SVN)

Versionamento Distribuito

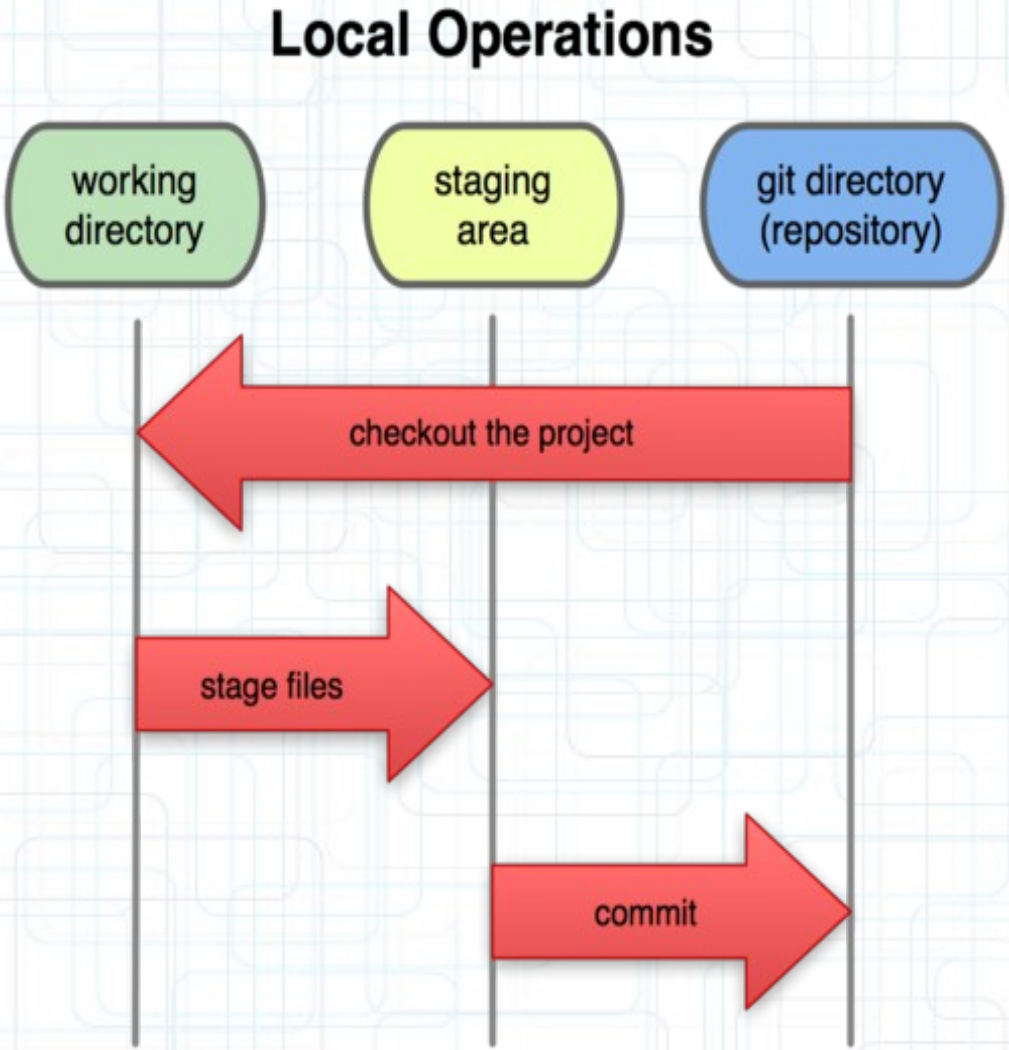


- Può non esistere un server centrale
- O possono essercene diversi
- Ogni utente possiede TUTTA la storia
- Sviluppo NON lineare
- Git, Mercurial (hg)

Git: cominciare a lavorare

- Installazione:
  - Linux: ***apt-get*** / ***yum install git-core***
  - Windows: msysgit, tortoiseGit, ...
  - Mac: git-osx-installer
- Configurazione iniziale
  - git config --global **user.name** "Mario Rossi"
  - git config --global **user.email** "rossi@gmail.com"
  - git config --global **core.editor** gedit
  - git config --global **merge.tool** meld
  - git config --global **color.ui** “auto”

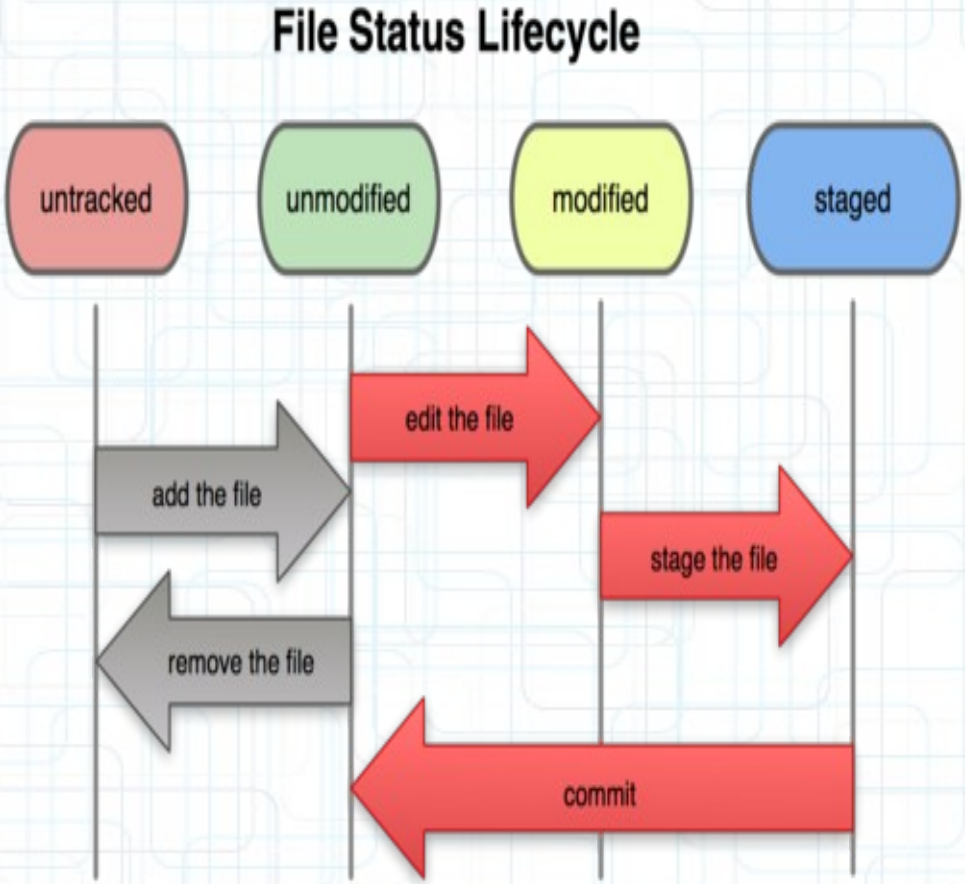
Git: cominciare a lavorare (2)



- git init
- git clone <url>

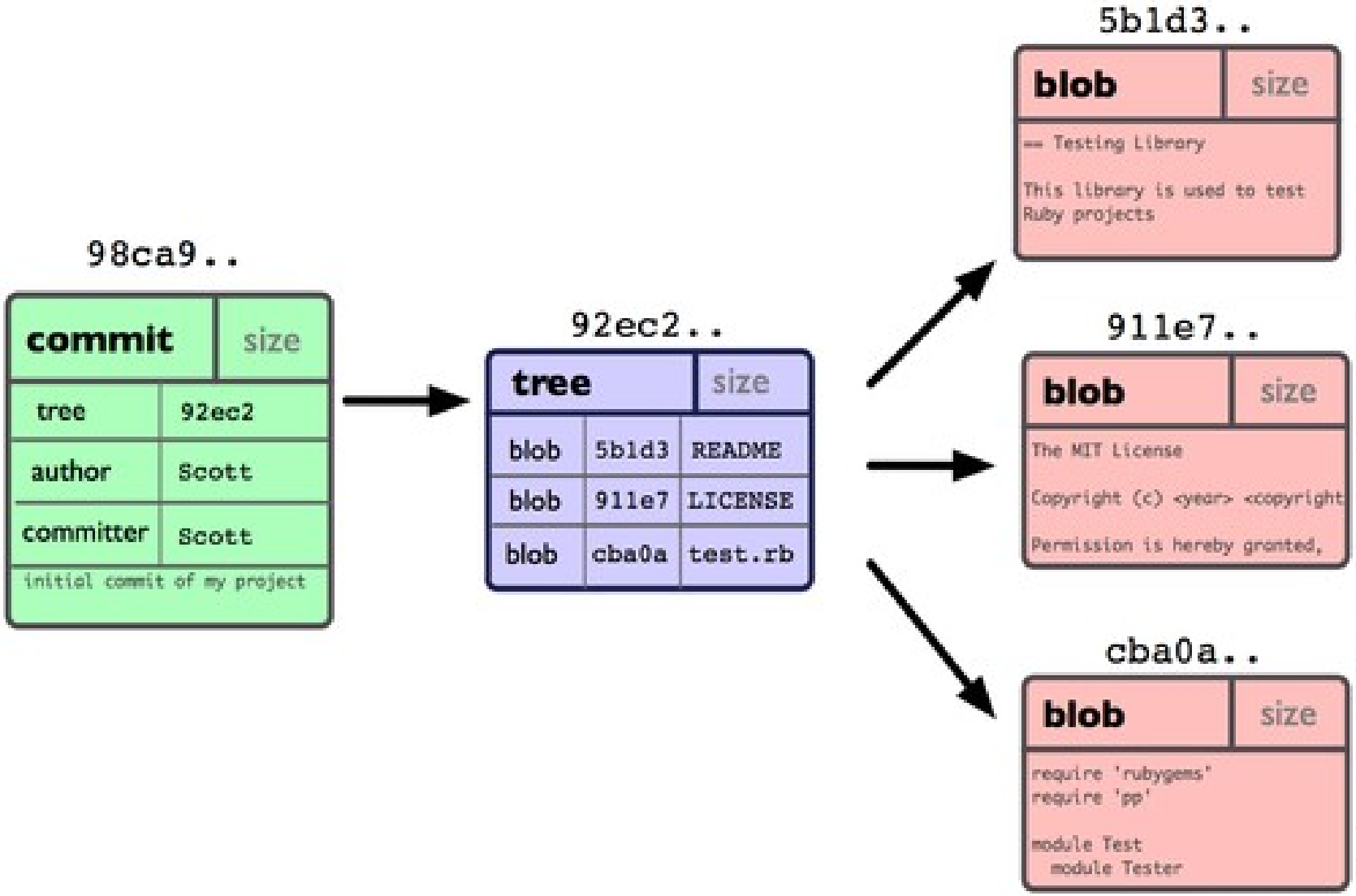
  # hack hack hack
- git status
- git add <file>
- git status
- git commit
- git checkout ...

Cominciare a lavorare (3)

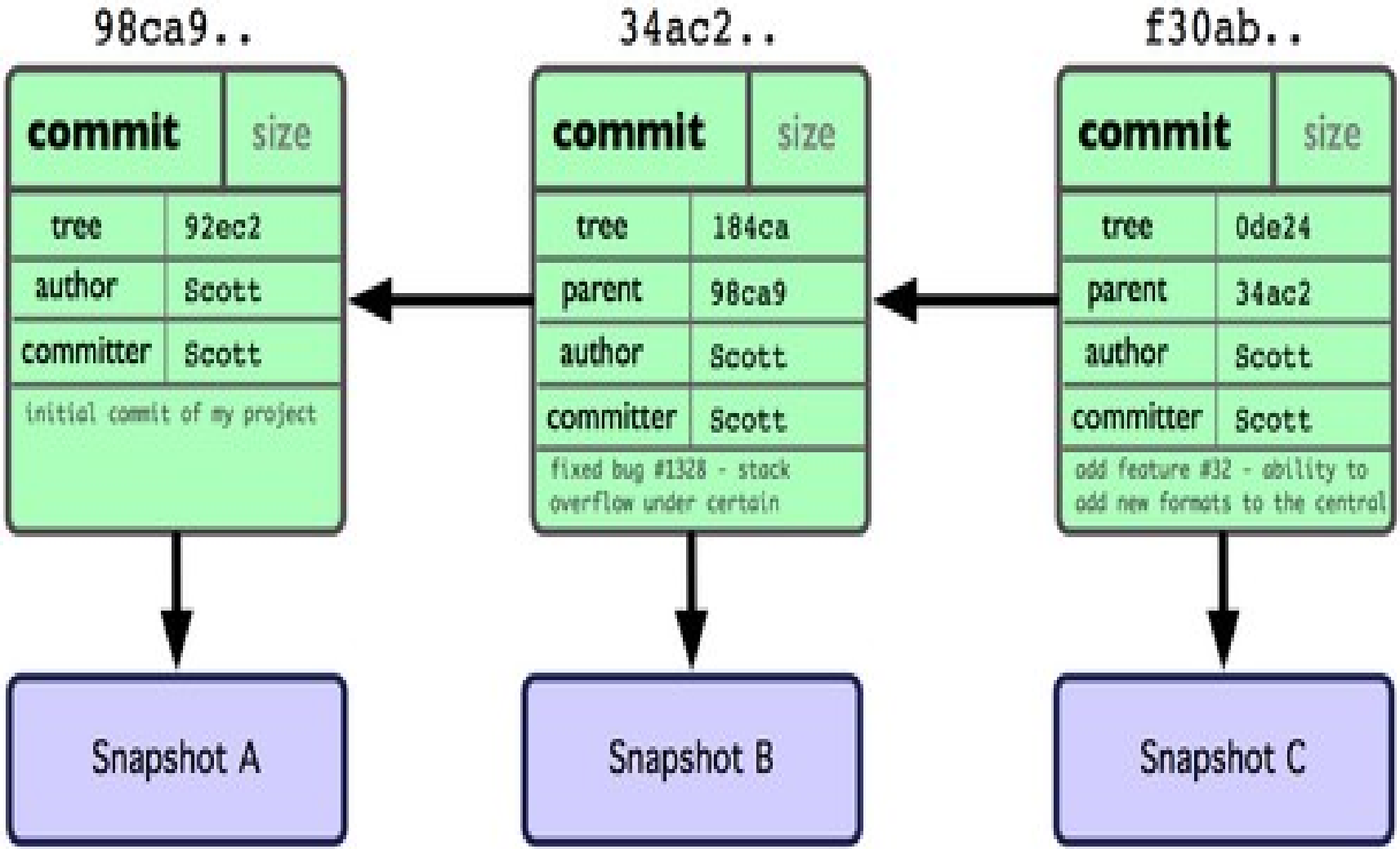


- **Untracked**: non indicizzato: nuovo
- **Unmodified**: file identico a quello nell'indice
- **Modified**: file diverso da quello dell'indice
- **Stage**: il file è stato preparato per il commit

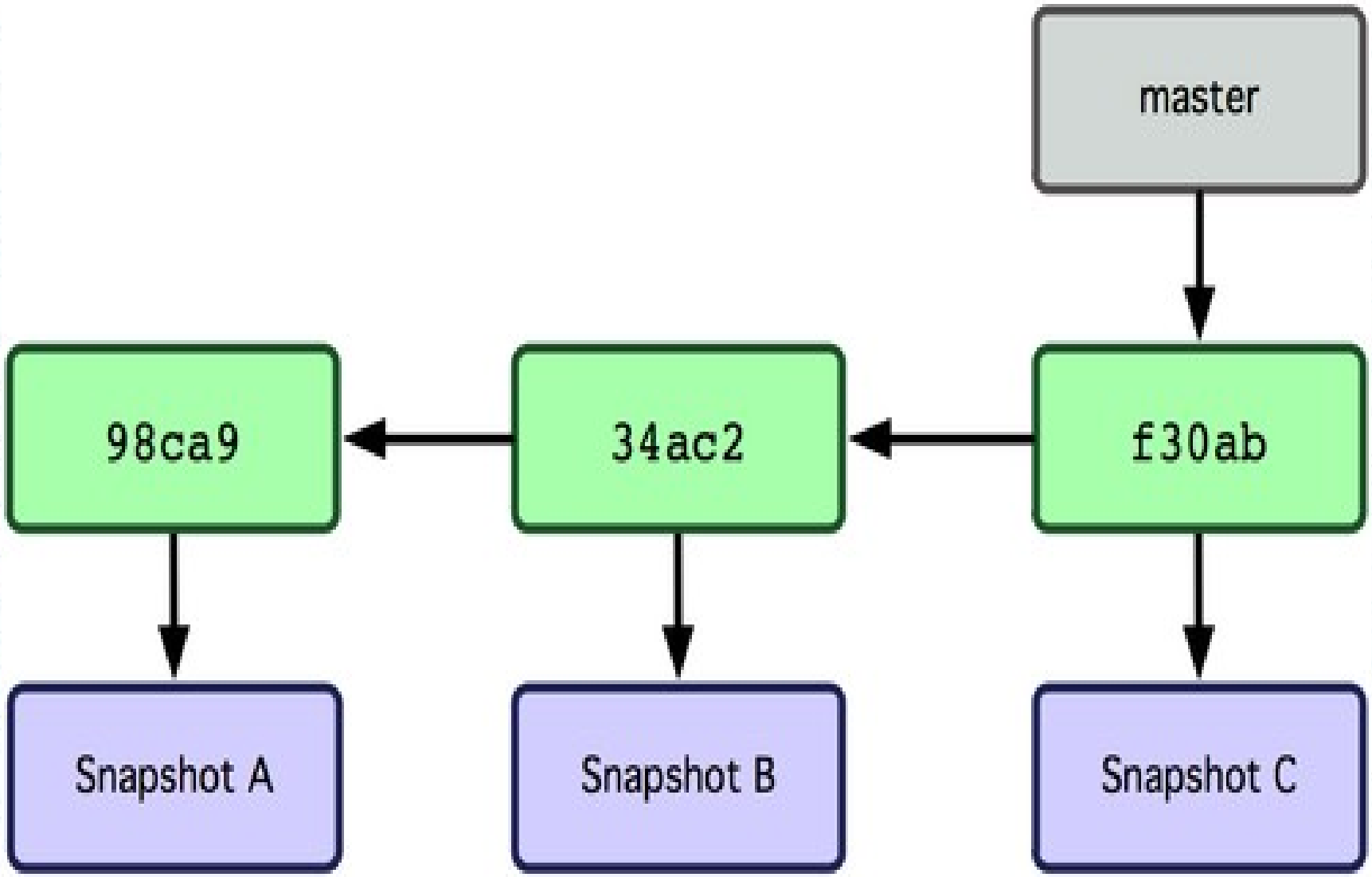
Cos'è un commit?



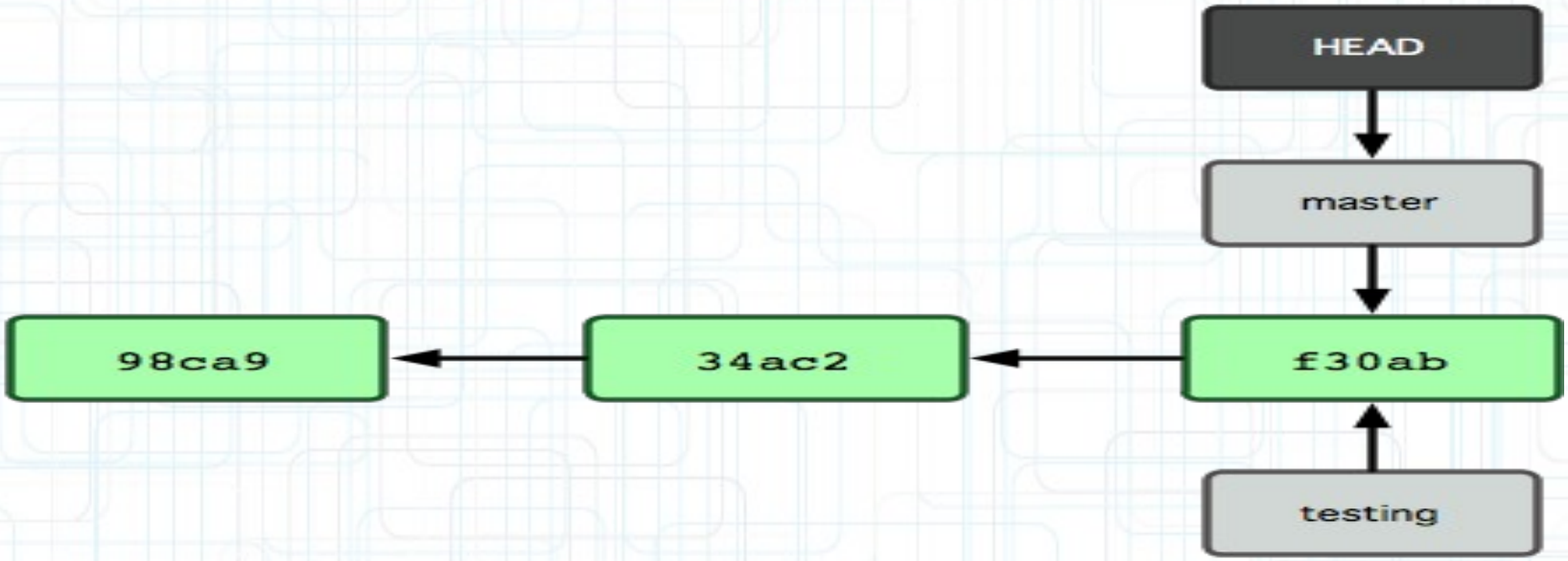
E la history? (git log)



Sha-1: per ogni commit e file

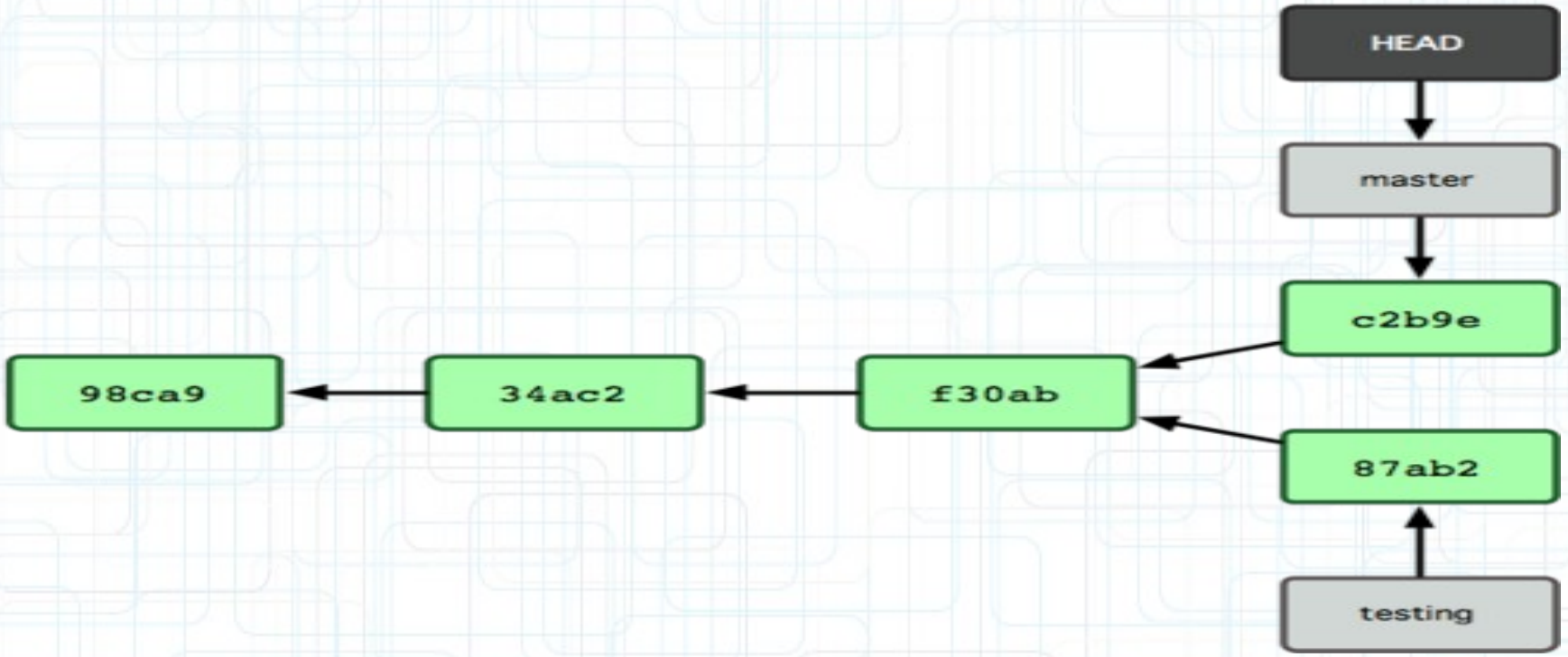


Cos'è un branch?



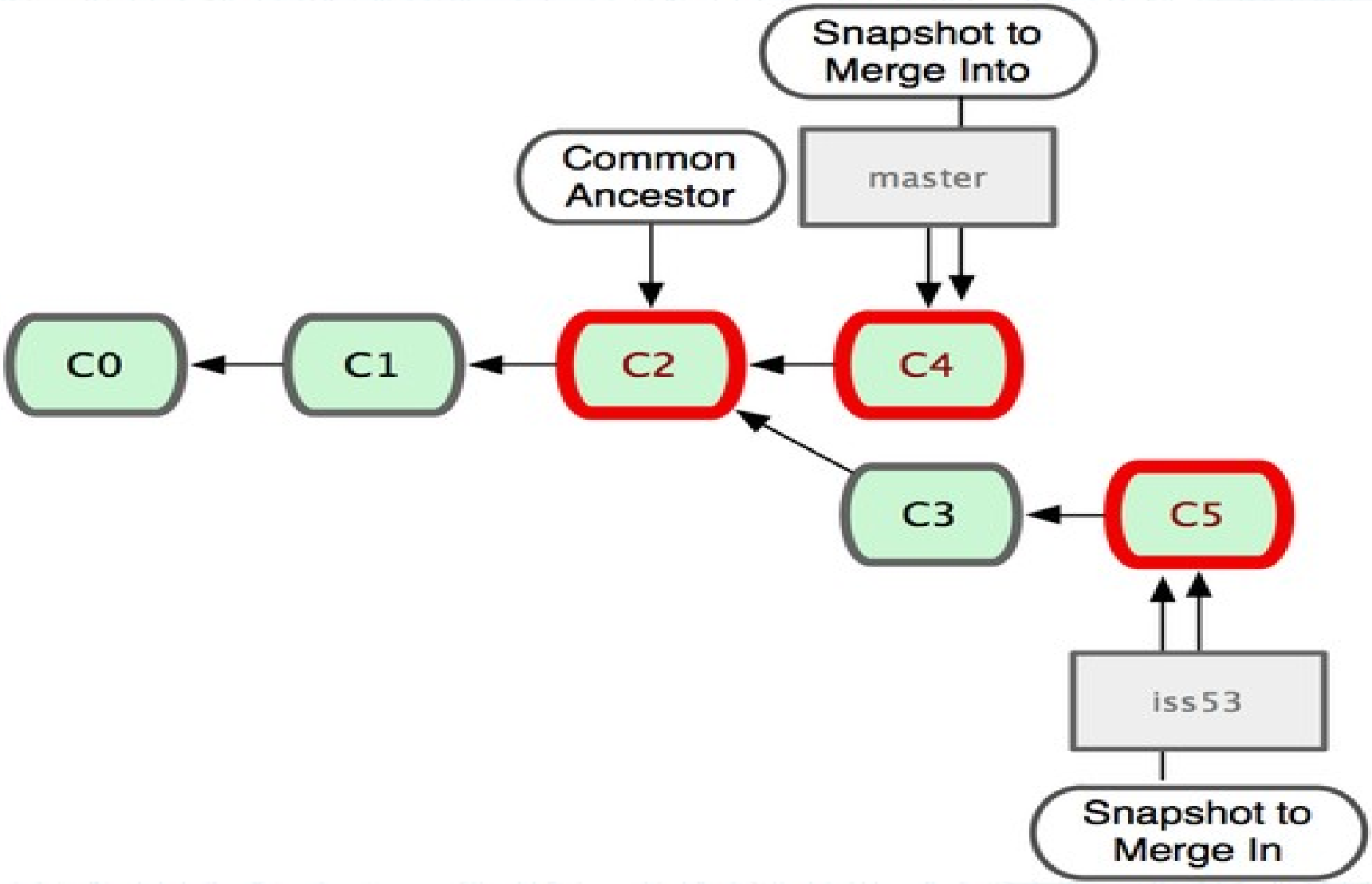
- **git branch** testing
- **git checkout** testing
- **git branch** *# lista dei branch*
- **git checkout -b** altrobranch master
- *HEAD*: puntatore al commit corrente

Analizzare la history



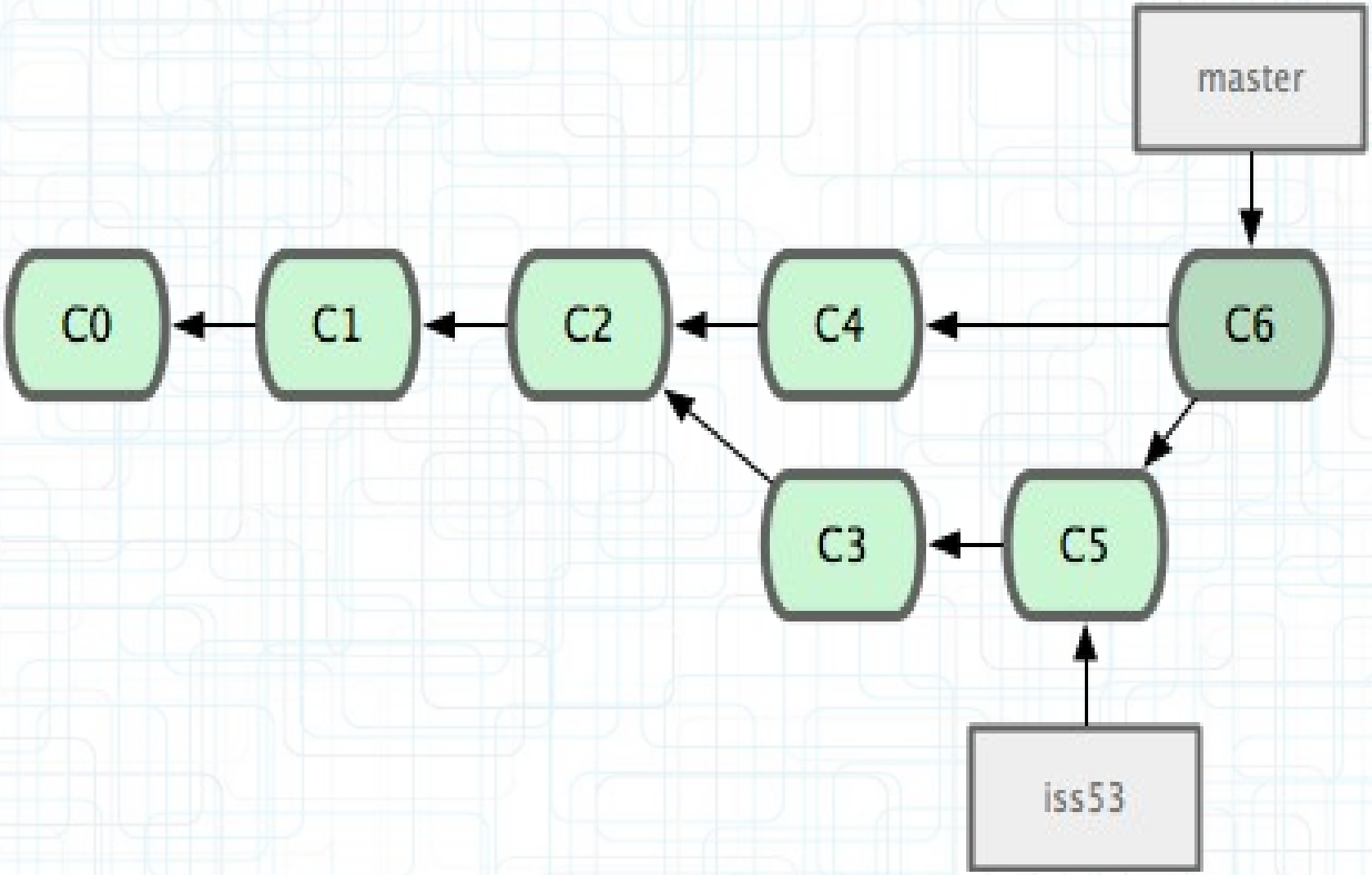
- **git diff** testing – **git diff** testing master
- **git diff** f30ab – **git diff** HEAD~
- **git log -p** *# history con diff per ogni commit*
- **git show** 34ac2

Merge di due branch



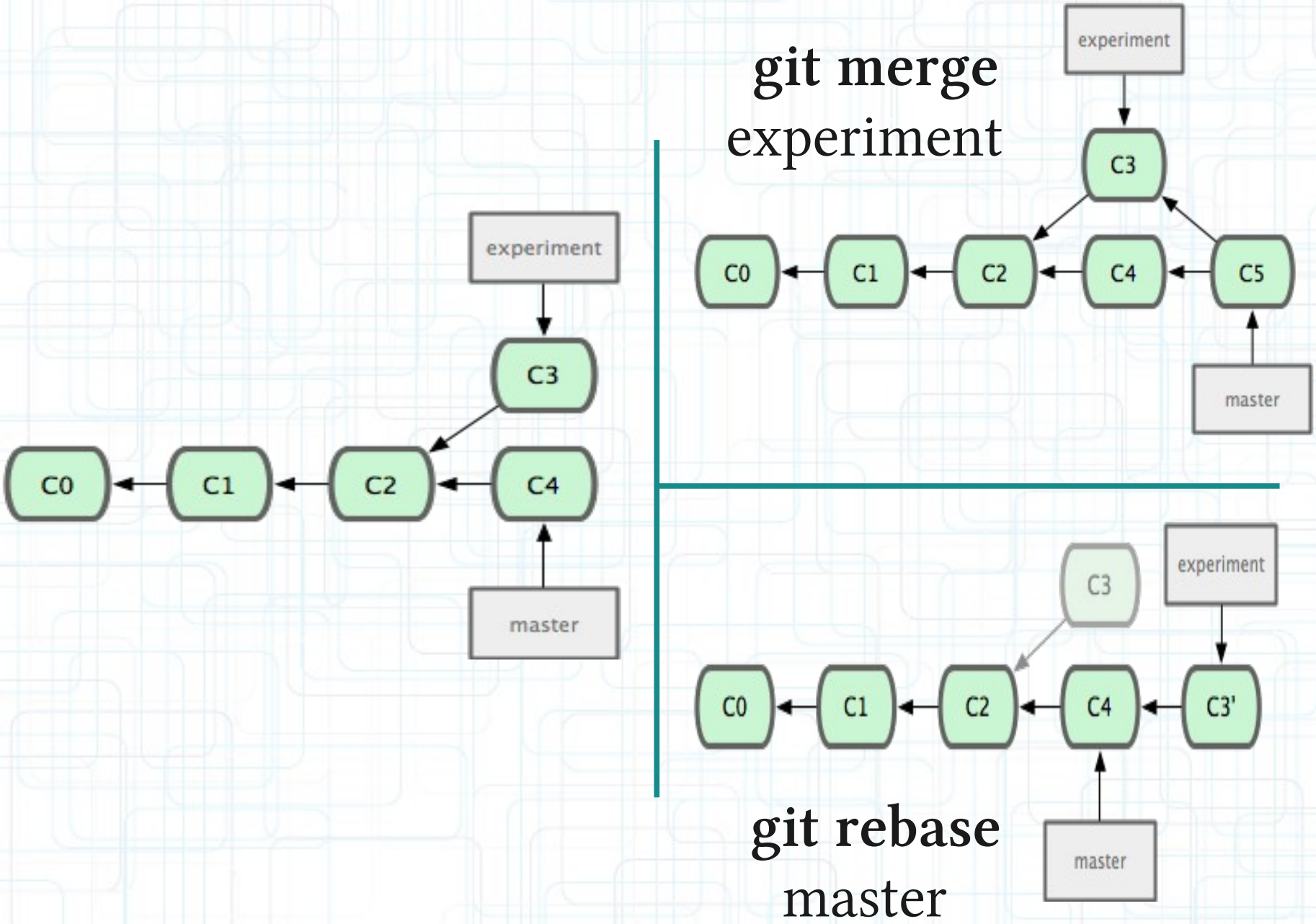
- **git merge** iss53

Merge di due branch (2)



- **git merge** iss53

Git rebase


git merge
experiment
experiment
C3
C0
C1
C2
C4
C5
master
experiment
C3
C0
C1
C2
C4
master
C3
experiment
C0
C1
C2
C4
C3'
master
git rebase
master

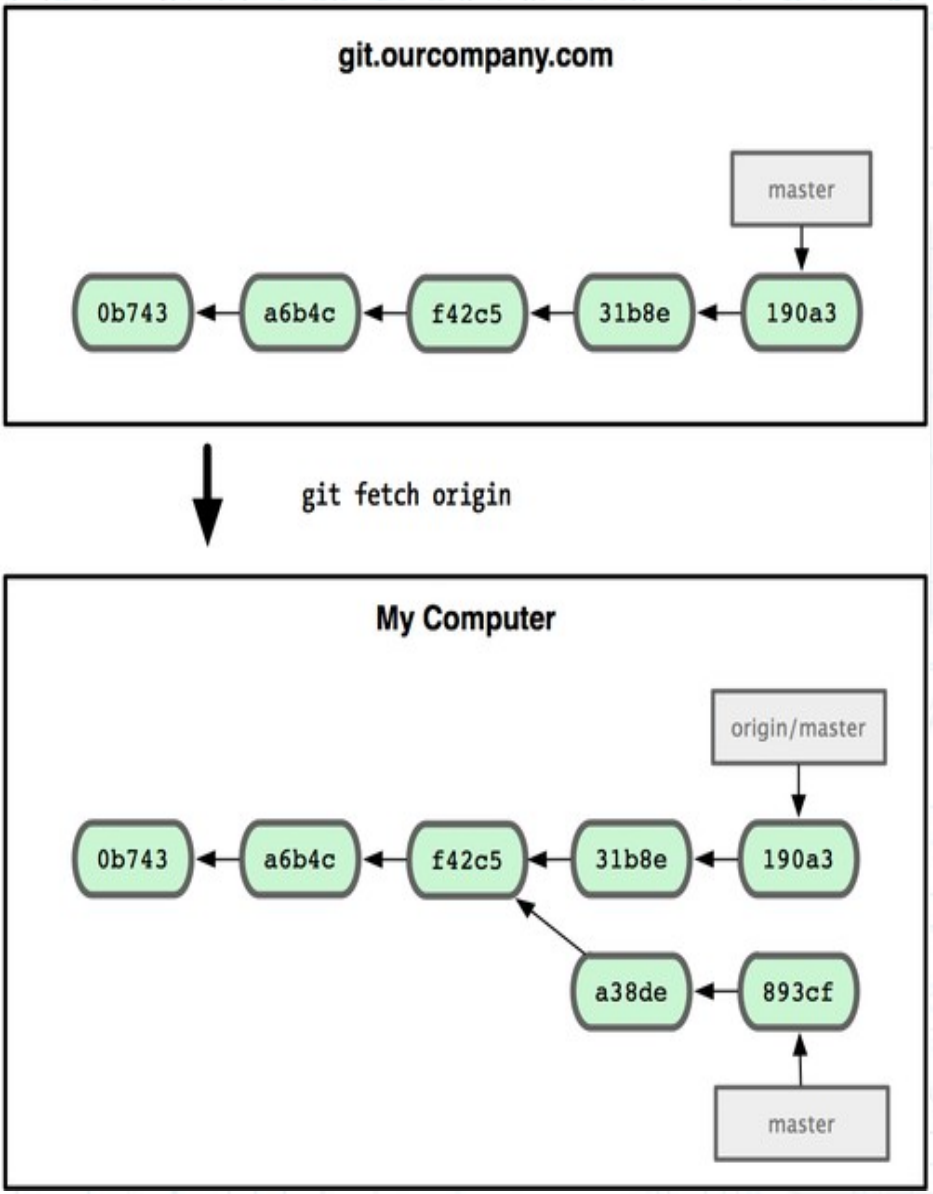
Risolvere i conflitti

- Risoluzione dei conflitti automatica efficiente
- Nel caso in cui l'automatismo non riesca git mette a disposizione dei tool:
- *git* **mergetool** *# file per file apre il vostro tool di merge files preferito* (meld, opendiff, kdiff3, vimdiff, ...)
- Risolti i conflitti basta un ***git commit*** per completare il merge
- O un ***git rebase --continue*** in caso di rebase

Collaborazione multiutente (1)

- Molti “sorgenti” remote per il vostro repository locale (git **remote** -v)
- E molti branch “remoti” (git **branch -r**)
- Il repository locale si aggiorna effettuando merge (o rebase) da quello remoto (git **pull** o git **pull** *origin master*)
- Se si ha accesso in scrittura al repository remoto si può inviare le proprie modifiche (git **push** o git **push** *origin master*)
- Questi comandi generano confusione, rivediamo tutto con più calma

Collaborazione multiutente (2)



- Il nome del remote predefinito è "**origin**"
- Il **pull** è composto di due parti: **fetch** + **merge**
- Fetch significa: *aggiorna i dati dal remote senza toccare quelli locali*
- È come avere un altro branch
- Fetch aggiorna TUTTI i branch remoti

Collaborazione multiutente (3)

- Il merge funziona esattamente come abbiamo già visto: git merge origin/master, dove "origin/master" è il nome del branch remoto che abbiamo appena scaricato
- git **pull origin master** quindi cosa vuol dire?
  - git **fetch origin** *# aggiorna remote-branches*
  - git **merge origin/master** *# merge di origin/master con....* ***con cosa?***
  - Con il branch "corrente", con la vostra HEAD
- Attenzione quindi: fate il checkout di master PRIMA di effettuare il pull di origin/master!

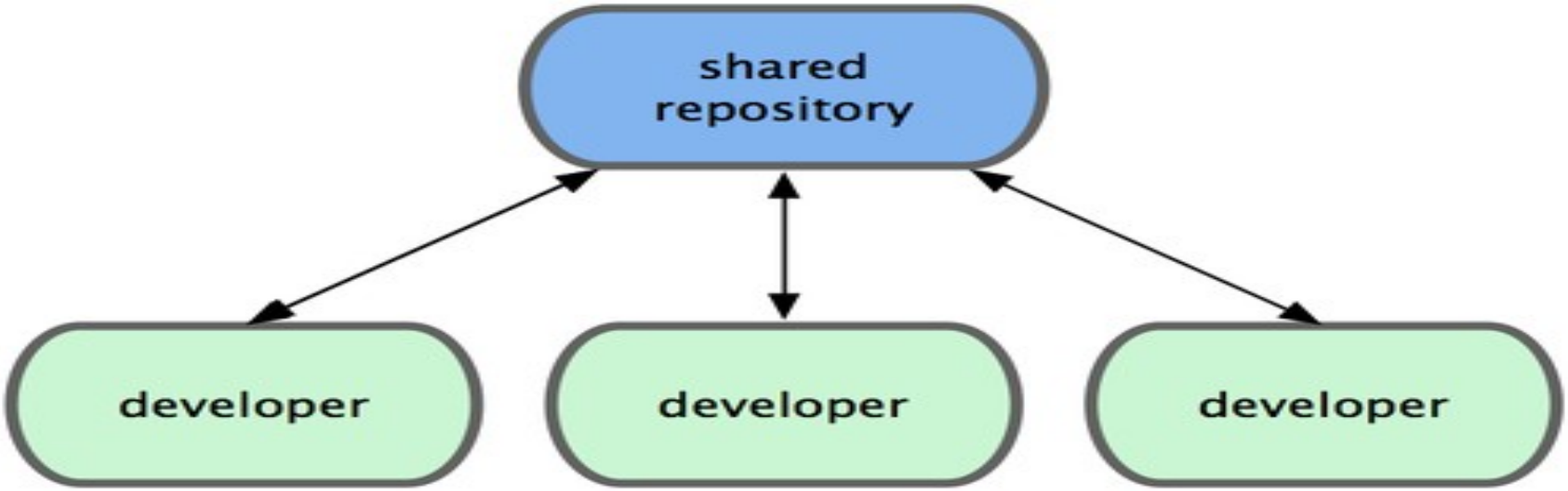
Collaborazione multiutente (4)

- E git **push** come funziona?
- git **push origin master** *# cosa fa?*
- Cerca in “origin” un branch “master”
- Cerca nel repository locale il branch “master”
- Quindi il branch dove vi trovate quando lanciate il comando non ha alcuna importanza: funziona in modo diverso da git pull...
- Ci sono molti altri parametri per questi due comandi ma l'utilizzo di base è questo

Alcune cose da sapere

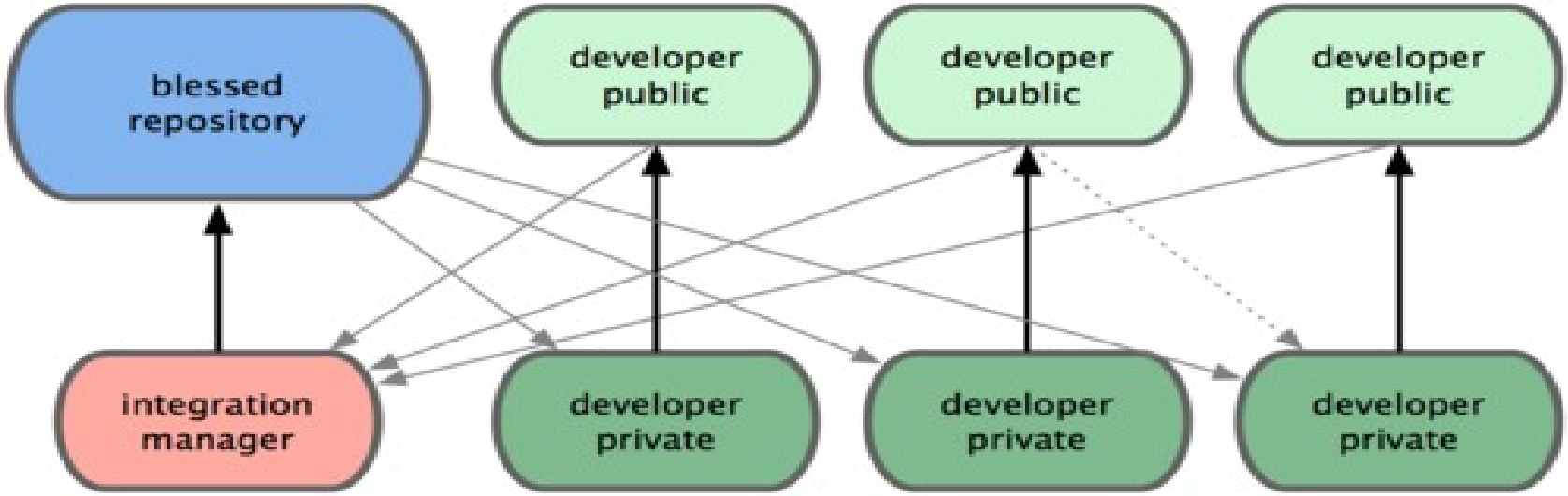
- git **rebase**: da evitare sui commit di cui è già stato effettuato il push
- Lavorando con ambienti remoti è in genere meglio utilizzare git merge.
- Per il lavoro in locale (ancora non condiviso) va benissimo anche il rebase.
- Con git-svn è il contrario: mai usare git merge!
- Esistono servizi che mettono a disposizione dei server git (es. http://github.com): è più facile cominciare usando uno di questi.

Workflow: centralizzato



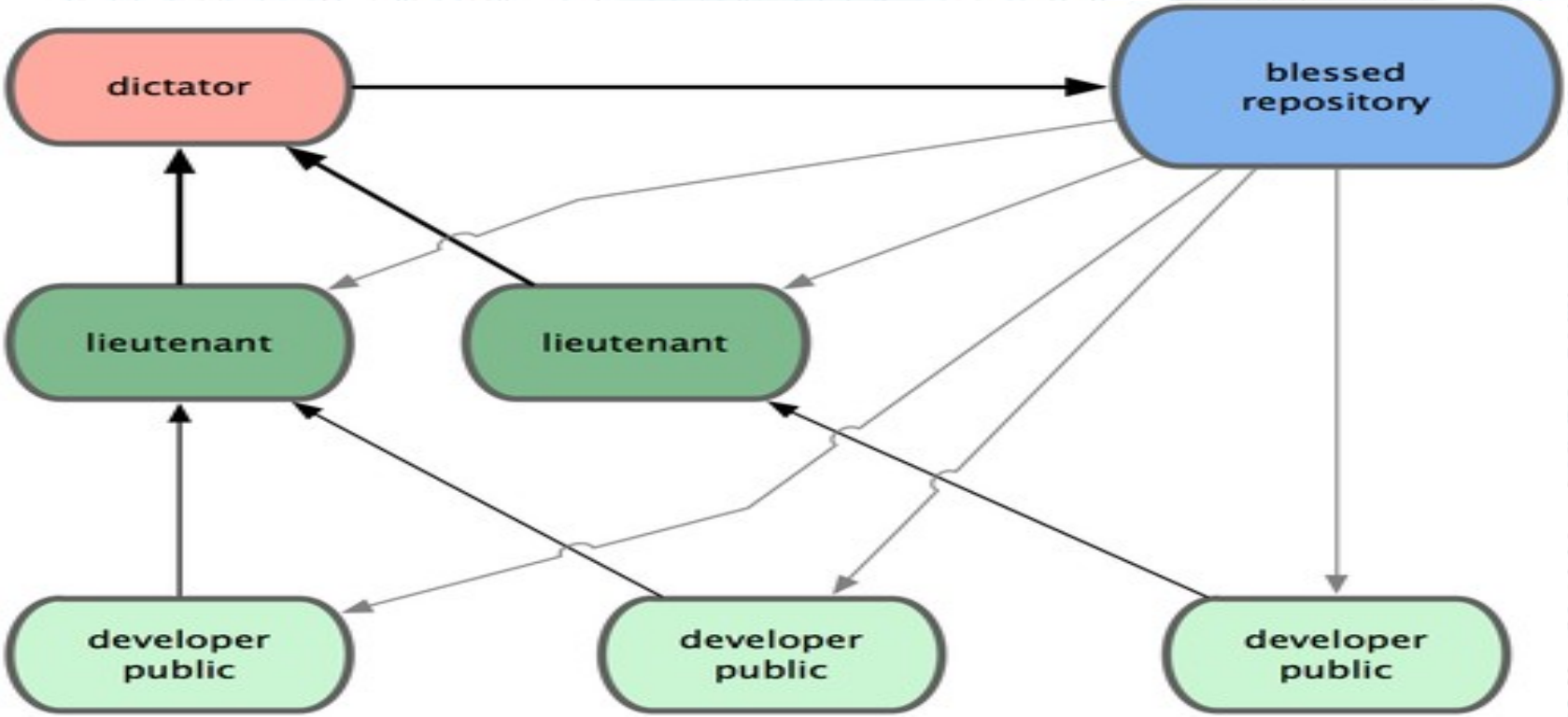
- Come SVN
- Ma con tutti i benefici di git per il lavoro in locale
- Più semplice per chi non è abituato a lavorare su sistemi distrubuiti (es.: per chi arriva da SVN)
- Buono per progetti con pochi sviluppatori (1-10)

Workflow: integration-manager



- Tutti hanno accesso in sola lettura ai repository degli altri (fetch/pull)
- Un “integratore” prende i lavori degli sviluppatori (fetch/pull), li integra e li mette sul repository “ufficiale” (push)
- Grande controllo e facile collaborazione

Workflow: dittatore e tenente



- Estensione dell'integration manager: gerarchico!
- Il kernel Linux è gestito così
- Linus Torvalds è il dittatore :P
- Progetti grandi e/o enormi

Altri strumenti (1)

- git **commit --amend** *# modifica l'ultimo commit*
- git **rebase -i** *# riscrive la storia dei commit*
  - **Pick**: prende il commit così com'è
  - **Edit**: si ferma e permette di modificare il commit
  - **Squash**: unifica con il commit precedente
- Git **stash** *# metter da parte lavori temporanei*
- git **cherry-pick** <commit-sha-1> *# applica le modifiche apportate da un singolo commit al branch corrente*

Altri strumenti (2)

- git **format-patch** *# per creare patch che comprendano tutti i dati di un commit: si può poi inviare via email*
- git **am** *# applica le patch create con format-patch*
- git **bisect** *# per scovare quale commit ha introdotto un bug: ricerca per bisezione marcando "good" e "bad" i commit via via che git li propone*
- git **blame** *# chi ha scritto questa riga di codice? Quando? In quale commit?*
- **gitk** e **git gui**, interfacce grafice ufficiali

Altri strumenti (3)

- git **bundle** *# per creare archivi dei commit e inviarli a chi non ha accesso al repository centrale*
- git **revert** *# applico al contrario le modifiche di un commit (per annullarle)*
- git **whatchanged** *# come git log ma con la lista dei file modificati/aggiunti/rimossi*
- git **log -Sregex** *# per cercare in qualunque commit una particolare stringa*
- git **reset** (*--hard*) *# reset dell'indice del branch (o anche i file) ad un particolare commit*

Integrazione con SVN

- git **svn clone** svn://mySVN.it/repo/project/ -T trunk -b branches -t tags
- git **svn fetch** *# scarica modifiche remote senza applicarle*
- git **svn rebase** *# svn update*
- git **svn dcommit** *# commit di tutte le modifiche locali su svn (ogni commit locale 1 commit su SVN)*

Links

- Home page di git: http://git-scm.com
- Doc ufficiale:
  - www.kernel.org/pub/software/scm/git/docs/
- Canale IRC: irc.freenode.net, #git
- Mailing list:
  - www.mail-archive.com/git@vger.kernel.org/
- Perché git è meglio di SVN, HG, ...
  - http://whygitisbetterthanx.com
- Libro libero (e gratuito) su git (noob-to-pro)
  - http://progit.org/

Links (2)

- Video sul branching
  - http://www.viddler.com/explore/fraserspeirs/videos/3/
- Video sul merging:
  - http://speirs.org/blog/2008/9/29/git-branching-and-fast-forward-merging.html
- Git per i pigri (tutorial):
  - http://www.spheredev.org/wiki/Git_for_the_lazy
- 3 (buoni) motivi per passare a git (da SVN):
  - http://markmcb.com/2008/10/18/3-reasons-to-switch-to-git-from-subversion/
- Tips su git divisi per livello di conoscenza
  - http://gitready.com/

Links (3)

- Altro libro libero su git
  - http://book.git-scm.com/
- Linus Torvalds che parla di git (Video 1:10 ore)
  - http://www.youtube.com/watch?v=4XpnKHJAok8
- Elenco interfacce grafiche e tools per git
  - http://git.or.cz/gitwiki/InterfacesFrontendsAndTools
- Guida all'uso di git bisect per scovare i bug
  - http://ivanz.com/2009/03/27/git-bisect-the-awesome-way-to-find-the-mr-bug-commit/
- Confronto SVN – Git: uso da riga di comando
  - http://git.or.cz/course/svn.html

Grazie dell'attenzione

daniele.segato@gmail.com
http://natonelbronx.wordpress.com