

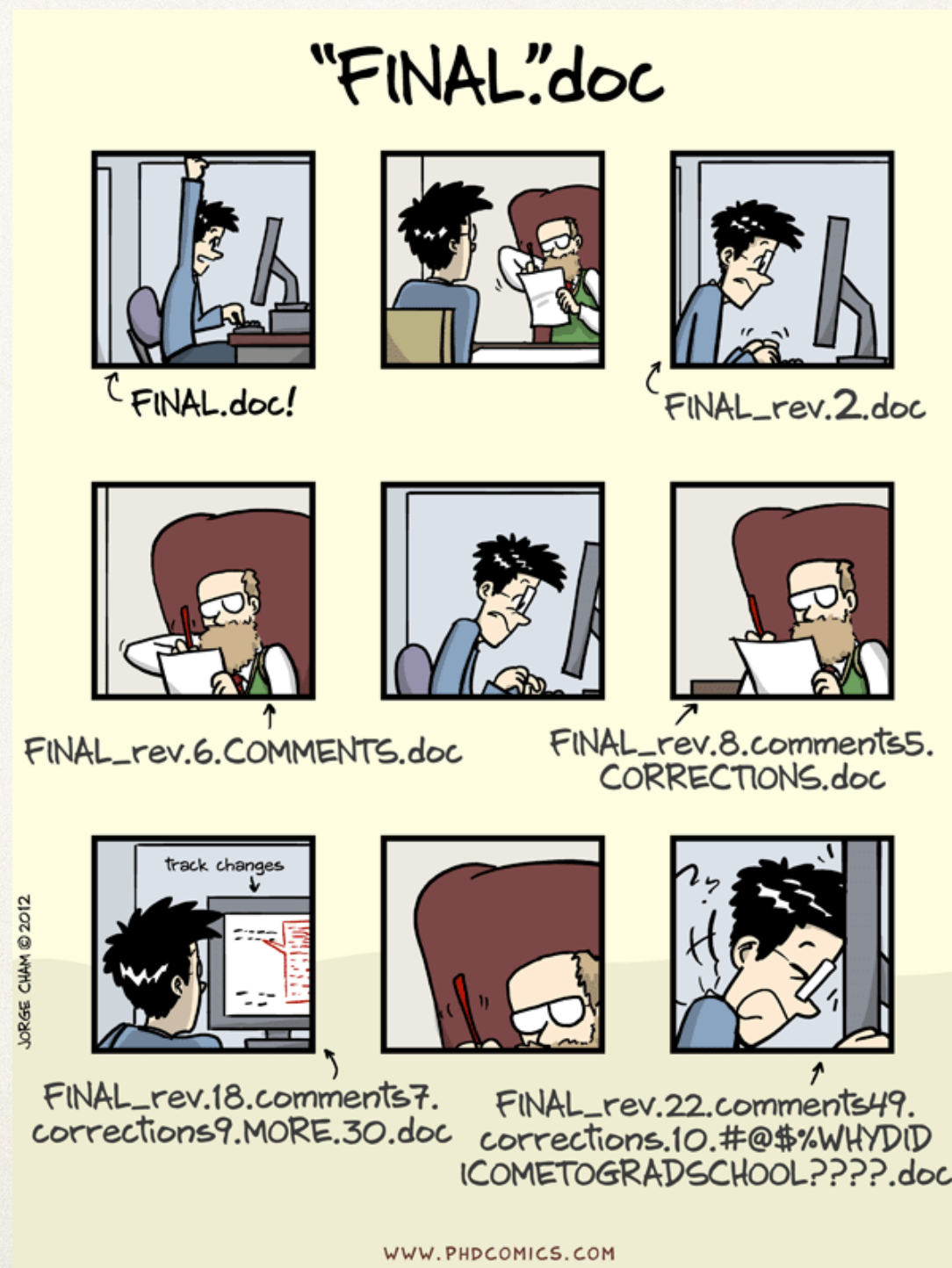
Sviluppo software in gruppi di lavoro complessi

Mattia Monga, Carlo Bellettini, ...

13 Ottobre 2016

Maintain a Single Source Repository

<http://www.phdcomics.com/comics/archive.php?comid=1531>



Source Code Management (versioning)

M. Fowler <http://www.martinfowler.com/articles/continuousIntegration.html> 2006

- ❖ Cambiato drasticamente negli ultimi 10 anni
 - ❖ Martin Fowler
 - ❖ *“The current open source repository of choice is Subversion. (The older open-source tool CVS is still widely used, and is much better than nothing, but **Subversion is the modern choice.**)”*
 - ❖ *“One of the features of version control systems is that they allow you to create **multiple branches**, to handle different streams of development. This is a useful, nay essential, feature - but it's **frequently overused and gets people into trouble**. Keep your use of branches to a minimum.”*

Versioning distribuito: Git

Linus Torvalds (Pt.1)

<https://www.youtube.com/watch?v=4XpnKHJAok8&t=3m2s>



**superior source control
management system than CVS is.**

Linus Torvalds (Pt.2)

<https://www.youtube.com/watch?v=4XpnKHJAok8&t=15m25s>

Distribution

(It's so much more than just off-line work)

- Collaboration
 - Commit changes without disturbing others
 - Branches that affect the main repository is a no-no!
 - Trust your data
 - ... without having to implicitly trust everybody else
 - Get away from the "commit access" mentality
 - No "special write access" class
 - No politics, and no granting/revoking of access
 - ... without having to implicitly trust your hosting
- Release engineering
 - Concurrent development/test/release cycles

**And this is really, really
important and is very**

Git

<https://imgs.xkcd.com/comics/git.png>

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



Cosa vedremo

- ❖ Alcuni internals
- ❖ Alcuni comandi e usi un po' più avanzati
- ❖ Alcuni workflow

Comandi base (che assumo già conosciate)

<http://git-scm.com/docs>

- ❖ git init
- ❖ git clone
- ❖ git diff
- ❖ git status
- ❖ git log
- ❖ git branch
- ❖ git add
- ❖ git commit
- ❖ git checkout
- ❖ git merge
- ❖ git cherry-pick
- ❖ git remote
- ❖ git push
- ❖ git pull

Formato degli objects

```
def put_raw_object(content, type)
  size = content.length.to_s

  header = "#{type} #{size}\0" # type(space)size(null byte)
  store = header + content

  sha1 = Digest::SHA1.hexdigest(store)
  path = @git_dir + '/' + sha1[0...2] + '/' + sha1[2..40]

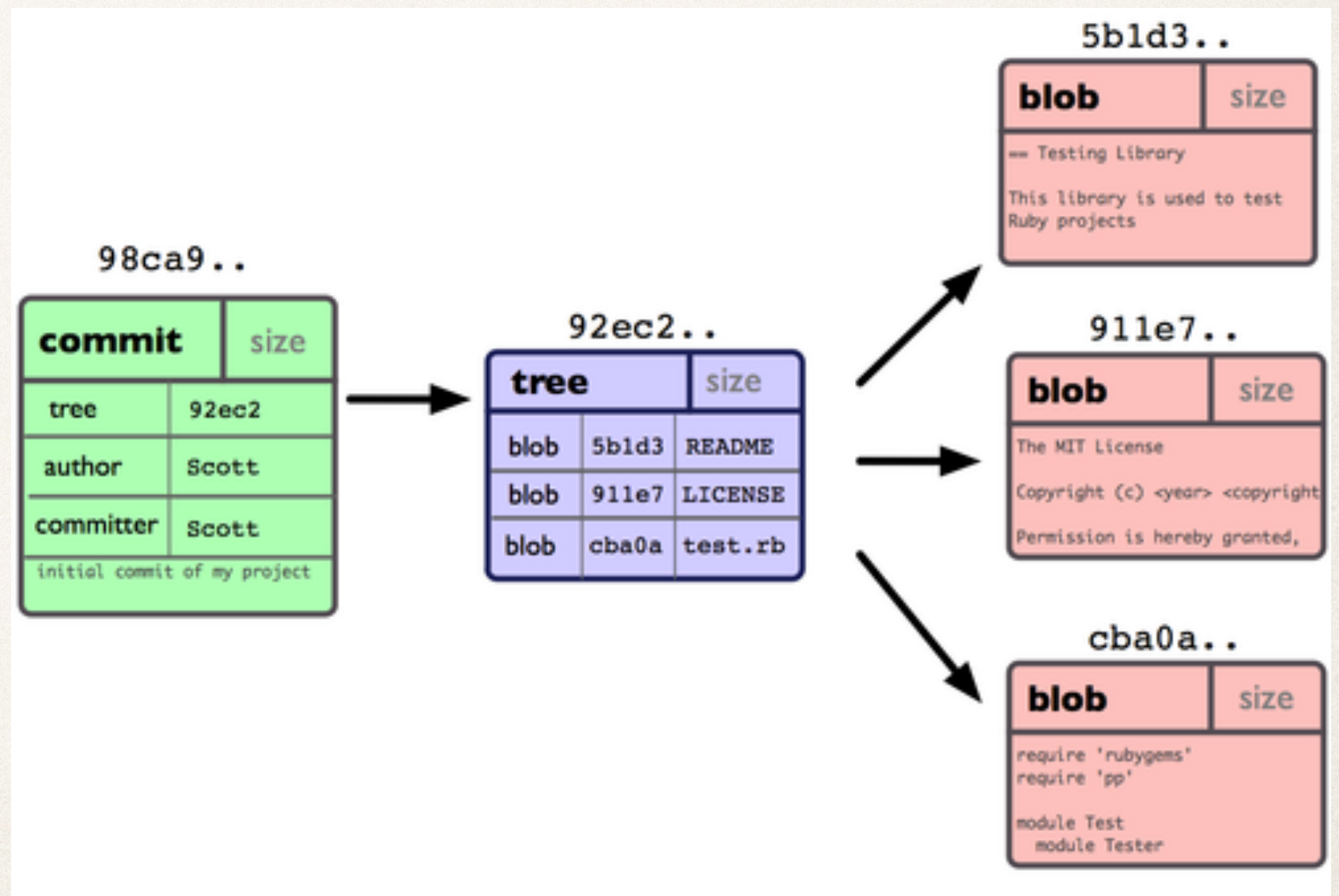
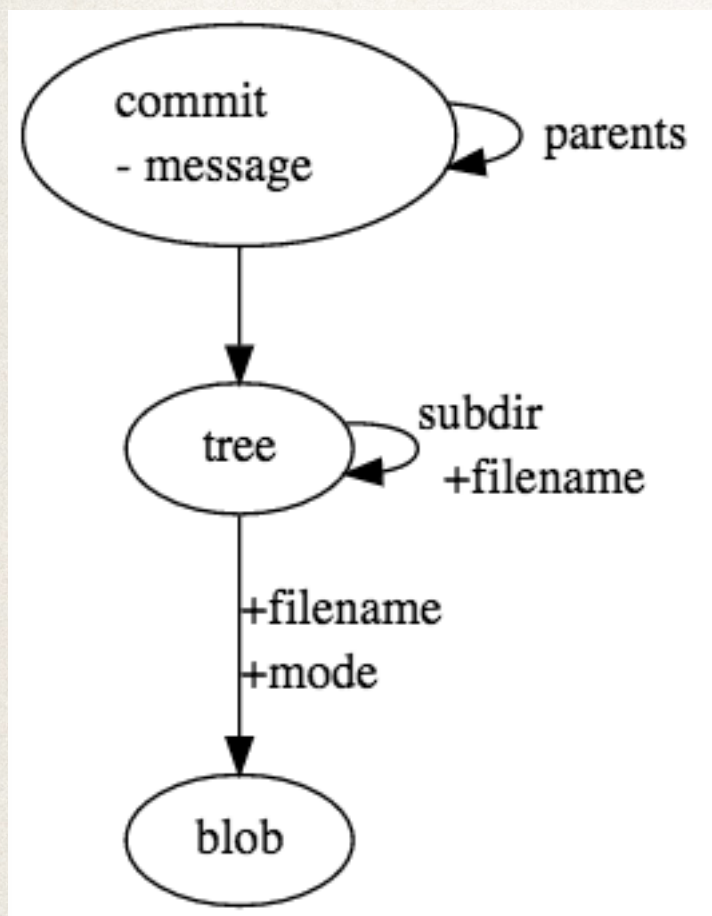
  if !File.exists?(path)
    content = Zlib::Deflate.deflate(store)

    FileUtils.mkdir_p(@directory+'/' + sha1[0...2])
    File.open(path, 'w') do |f|
      f.write content
    end
  end
  return sha1
end
```


Git data: commit, tree, blob

<http://eagain.net/articles/git-for-computer-scientists>

<http://arokem.github.io/2013-09-16-ISI/lessons/git-notebook/git-for-scientists.slides.html>



Git data: repository

