Marco Corvo
CNRS and INFN

THE GIT VCS SYSTEM

Outline

- Collaboration needs wrt a VCS system
- Why Subversion
  - with some experience
- Why Git

Requirements

- Distributed system (no single point of failure)
- Wide community of developers/contributors
- Reliable, strong and well supported
- ldap auth system
- Write once/Read many tagging
- Api for different languages interfaces

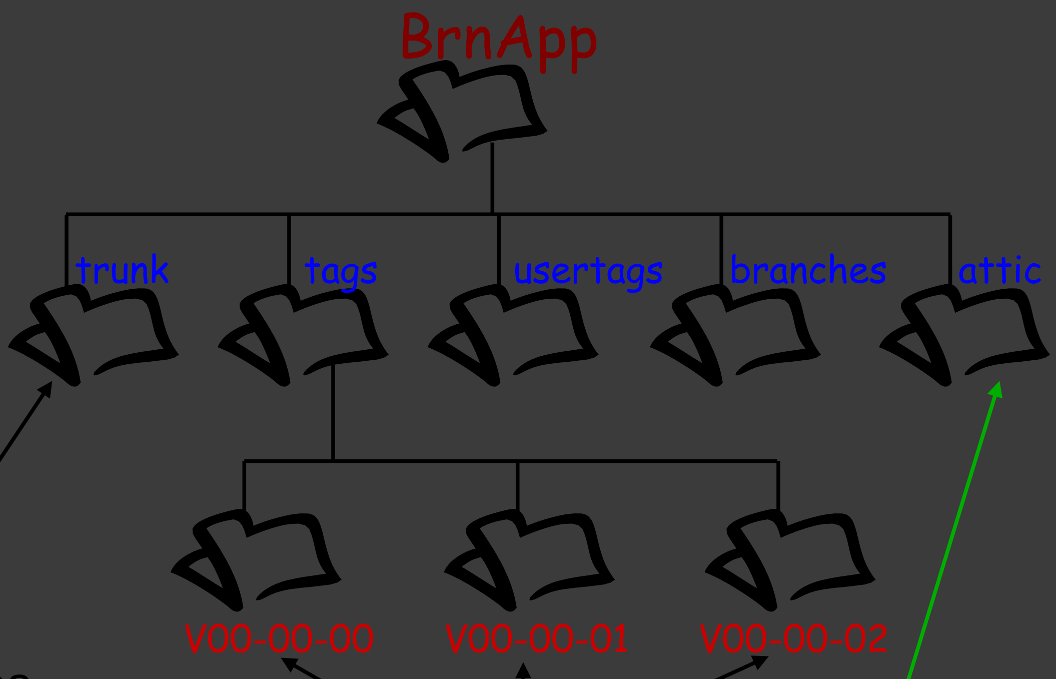
SuperB Subversion service

- Hosted in Padova (sbrepo.pd.infn.it)
  - Ldap auth service
  - Accessible via https with a WebDAV Apache module
- Structured with many repositories each hosting many packages
  - Critical issue: on one end we keep more control over granularity and avoid the explosion of revision numbers, on the other end there’s no single point of access when navigating the code and the user must know in advance which repo he needs to reach a given package

SuperB Subversion service

- “Write once/read many” tagging managed with a pre-commit hook
- Mail alerts
- CLI to svn developed in Perl
  - sbnewrel
    - Basic setup of the working directory
    - Checkout some useful packages and the list of all packages, along with their Tag, which make up the release
  - sbaddpkg
    - Checkout of all packages requested by the user

Subversion repo organization



development area

freezed code
(versions)

deleted code

subversion repository

tag

`Core/BaBar/tags/V00-00-00`

package name

Subversion basics

- Directories structure
  - much like a real file system
  - main code development follows the trunk line
  - creating a branch or a tag means copying a full snapshot of your project into an svn directory (../branches or ../tags)
- Network is accessed for most operations

Git in a nutshell

- Git is a DVCS (Distributed Version Control System)
  - Main diff with other VCS, like svn, is that client fully mirrors the repo
  - Every client contains all information, nice to solve servers failure
  - Every commit is a full snapshot of the project, not just a “diff” between versions

Git access protocols

- git://
  - fastest as it’s optimized for git itself
- http|s://
  - commonly used for “readonly” repositories. Needs WebDAV as svn for “write” ones.
- ssh://
  - Best choice, provides both fast and secure commits
- How to use ldap auth too?
  - As of version 1.6.6 there’s a Smart HTTP cgi script which let git client push over HTTP w/o WebDAV services

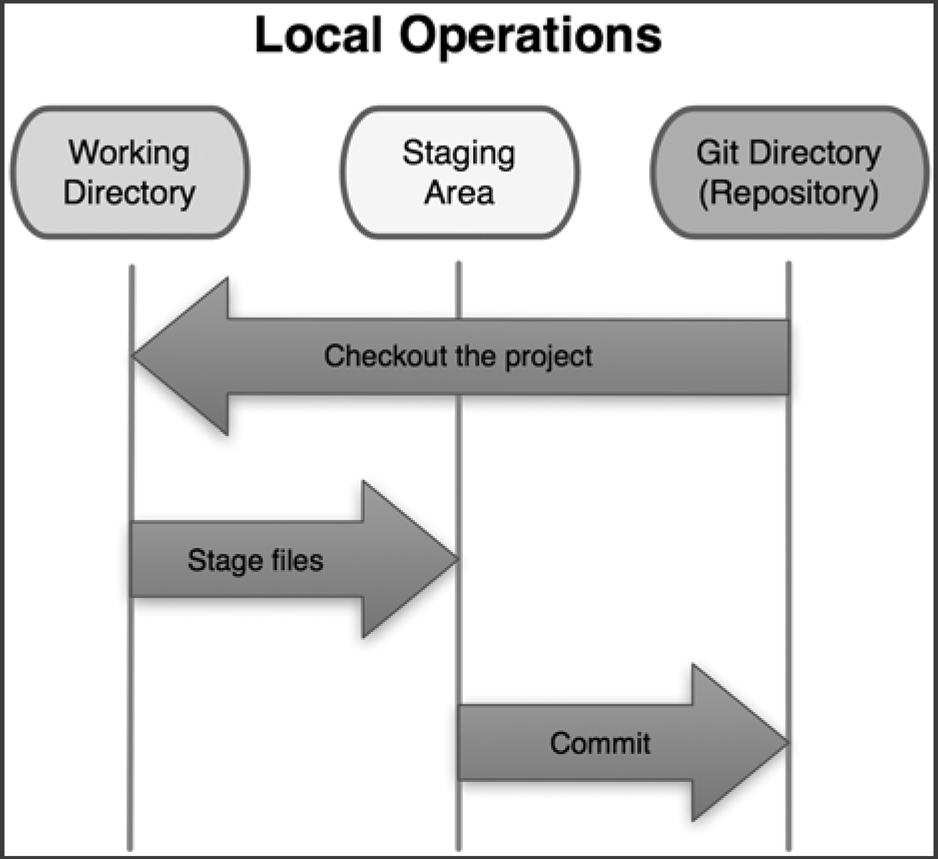
Git basics

- Nearly every operation is local
  - Diff, history navigation…there’s little that you can’t do even w/o connection
  - This also means that the equivalent as “svn list” or “svn cat” are not available
- Git has three states
  - Committed, modified, staged
  - Other VCS lack the last state “staged”

Git basics

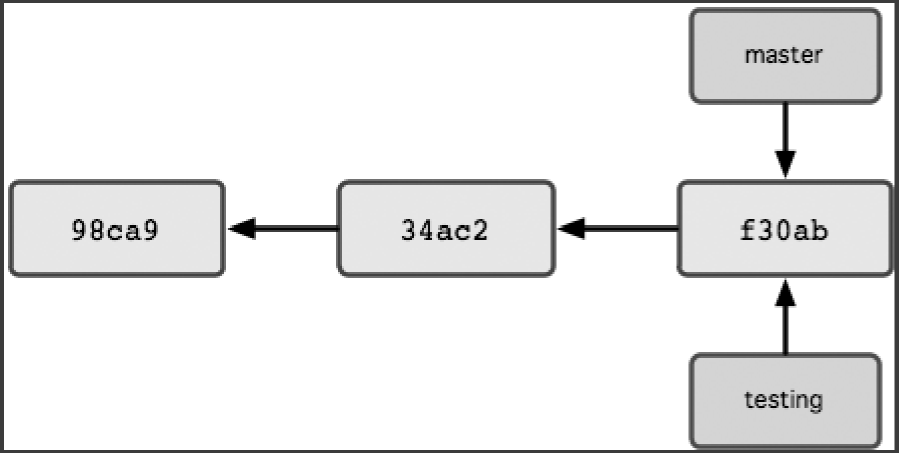
- You work always in the same directory, unlike svn or other VCS where the HEAD (trunk), branches and tags live in separate dirs
- Branches and tags are easier to manage
  - Basically they’re pointers to a given snapshot

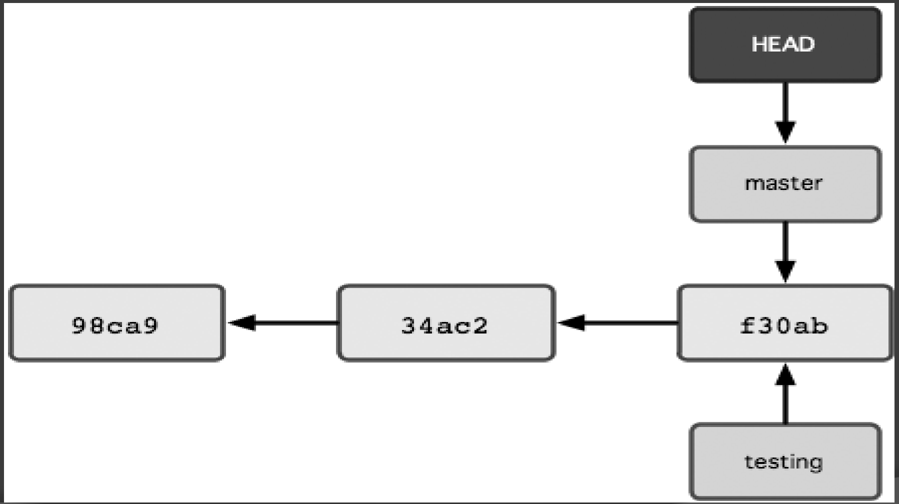
Git three states



- Unlike other VCS, Git has an intermediate “staging” area, where changes go before commit

Git branches

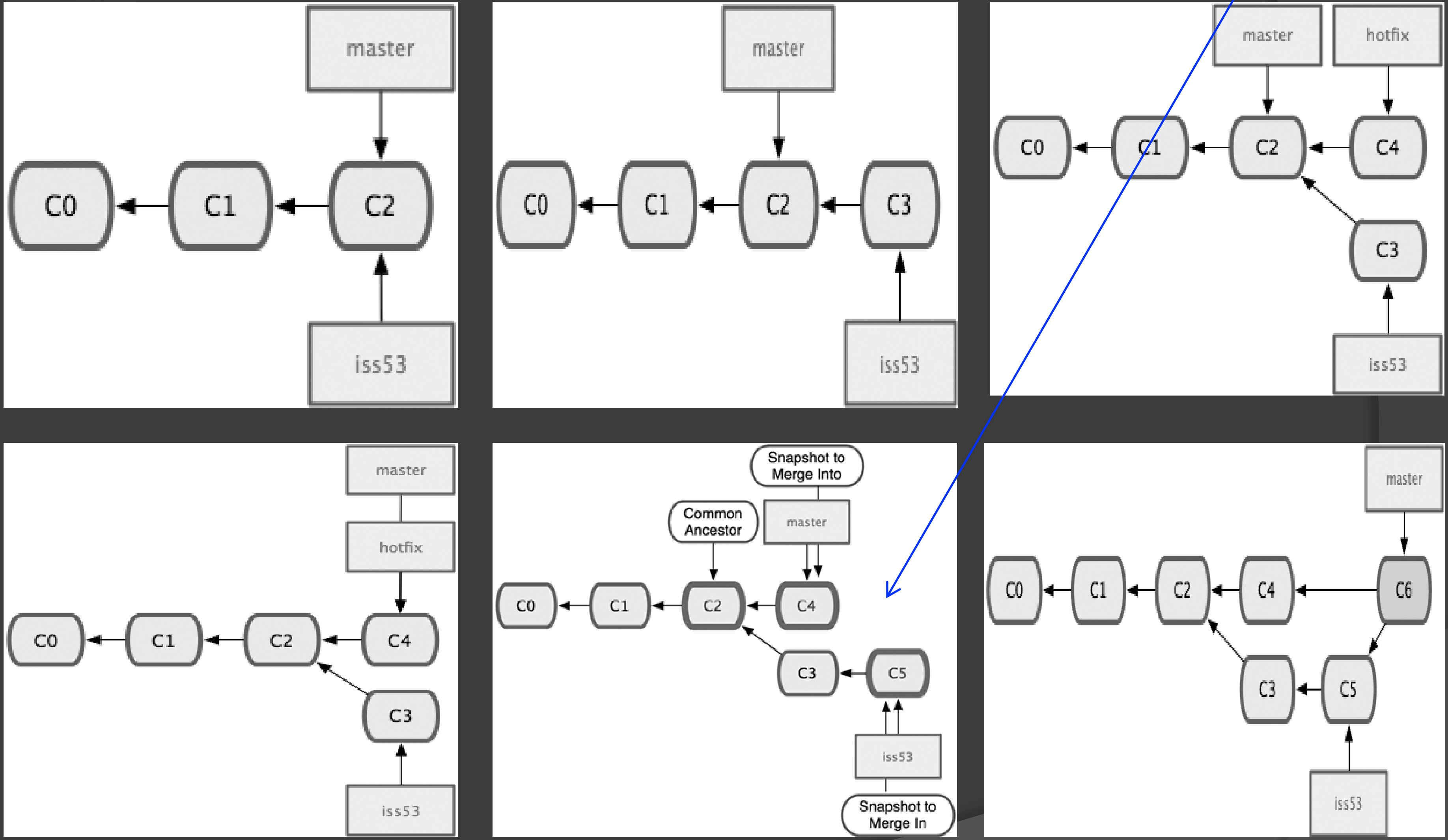




- Two branches point to the same snapshot
- HEAD points to the “current” branch

Typical Git brach workflow

Three way merge



GitProjects

- https://git.wiki.kernel.org/index.php/GitProjects
- A couple for all: Linux kernel and Android os

Conclusions

- This is not supposed to be an exaustive presentation
  - I started to play wit Git a couple of weeks ago
- The idea is to start a real and serious evaluation of Git and possibly take a decision in one year from now
  - We can start migrating a FastSim repo to perform some tests
- It’s nevertheless clear, at least to me, that Git philosophy is quite different form Cvs or Svn ones and needs some “user behaviour” adjustments.