CSC 7003 : Basics of Software Engineering

J Paul Gibson, D311

paul.gibson@telecom-sudparis.eu

http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC7003/

Version Control

…/~gibson/Teaching/CSC7003/L12-VersionControl.pdf

Version control : background

Version control is also known as *resource control* or *source control*

It is the management of changes to documents, programs, and other information stored as computer files.

It is mostly used in software development, where a team of people may change the same files, and it is an important part of software configuration management

Changes – known as revisions - are usually identified by a unique ID - the *revision number*,

Each revision is usually associated with a timestamp and the person making the change.

Revisions can be compared, restored, and with some types of files, merged.

Version control : a (selected) history

Local

- 1972 SCCS
- 1982 RCS

Client-Server

- 1990 CVS (Concurrent Versioning System)
- 2000 Subversion

Distributed

- 2001 GNU arch
- 2000 DCVS
- 2003 SVK
- 2005 Bazaar
- 2005 Git
- 2007 Fossil

Distributed Systems

1. No canonical, reference copy of the code base exists by default; only working copies.

2. Common operations such as commits, viewing history, and reverting changes are fast, because there is no need to communicate with a central server.

3. Each working copy is effectively a remote backup of the code base and change history, providing natural security against data loss.

Version control : some key articles

The Source Code Control System , Marc J Rochkind, 1975

Design, implementation, and evaluation of a Revision Control System , Walter F Tichy, 1982

On Optimistic Methods for Concurrency Control , H.T. Kung and John T. Robinson, 1981

Version control : why?

Reversion: If you make a change, and discover it's not viable, how can you revert to a code version that is known to be good?

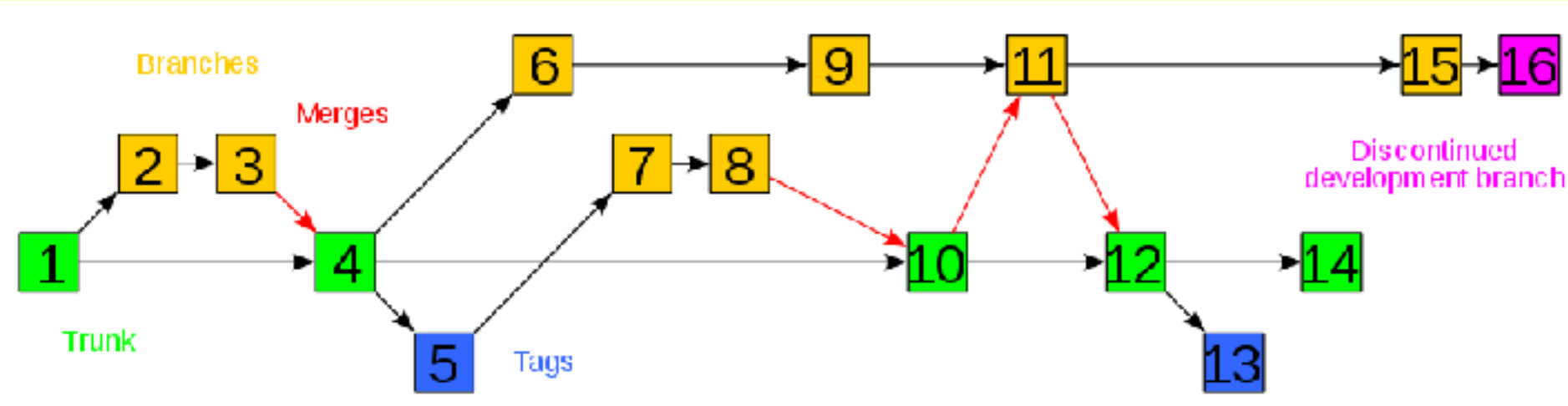
Change/Bug Tracking: You know your code has changed; do you know who, when and why? (When and where the new bug was introduced?)

Branches: How to introduce a completely new feature or concept and not mess up the working code?

Merging branches: If I divide up the code, how to merge new code with old code?

Parallel Development: How to manage independent developers making different changes to the same code?

Version Control: fundamental concepts



Tags (Baselines/Labels) – important snapshot of a project

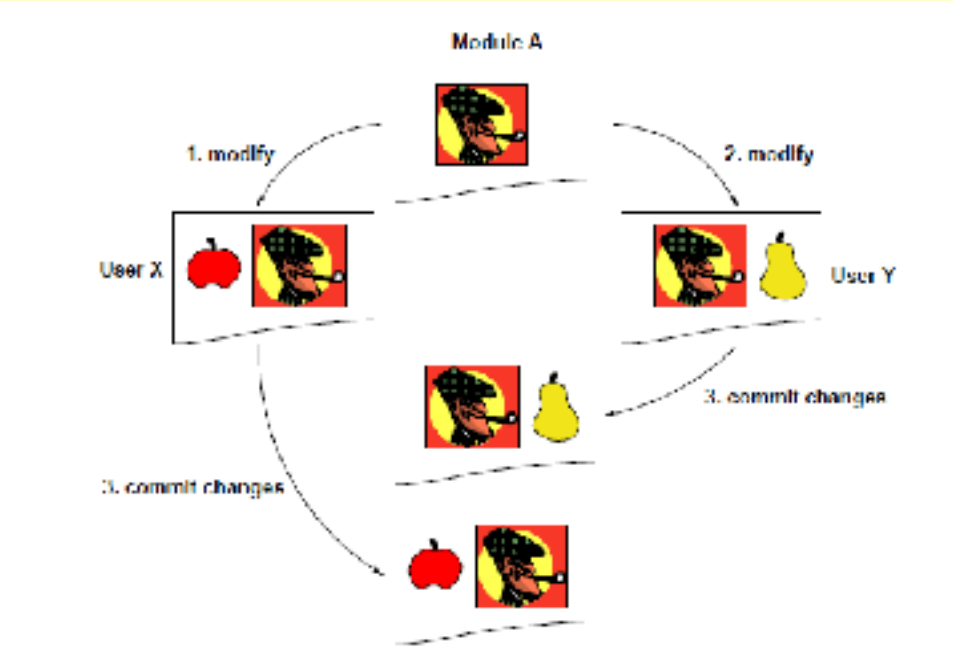
Branch - two (or more) copies of a project that may develop at different speeds or in different ways independently of each other.

Trunk (Baseline/Mainline) - The unique line of development that is not a branch

Merge - an operation in which two sets of changes are applied to a file or set of files or branches.

Version Control: fundamental concepts

Parallel Development: How to manage independent developers making different changes to the same code?



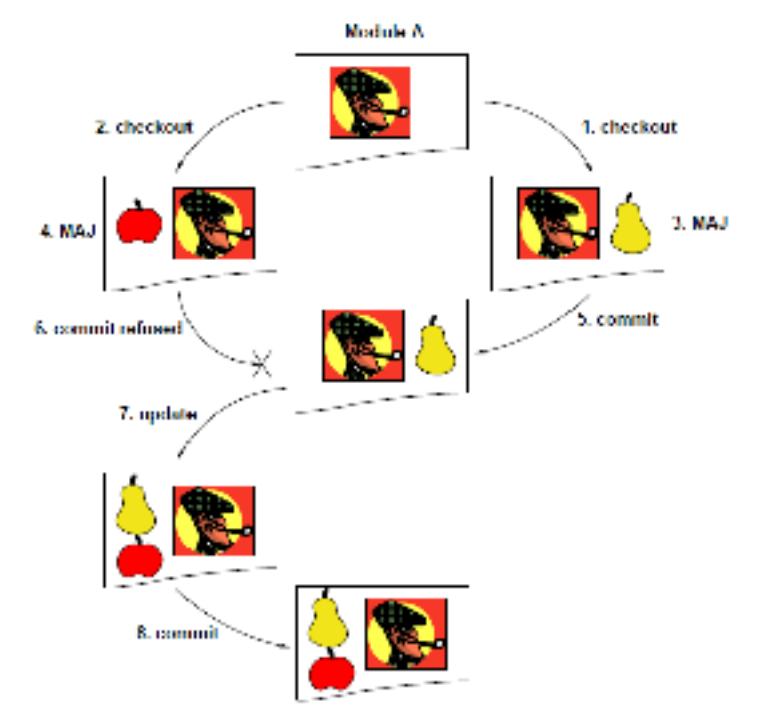
Solution: an *access protocol*

Most VCSs follow 1 of 2 approaches:

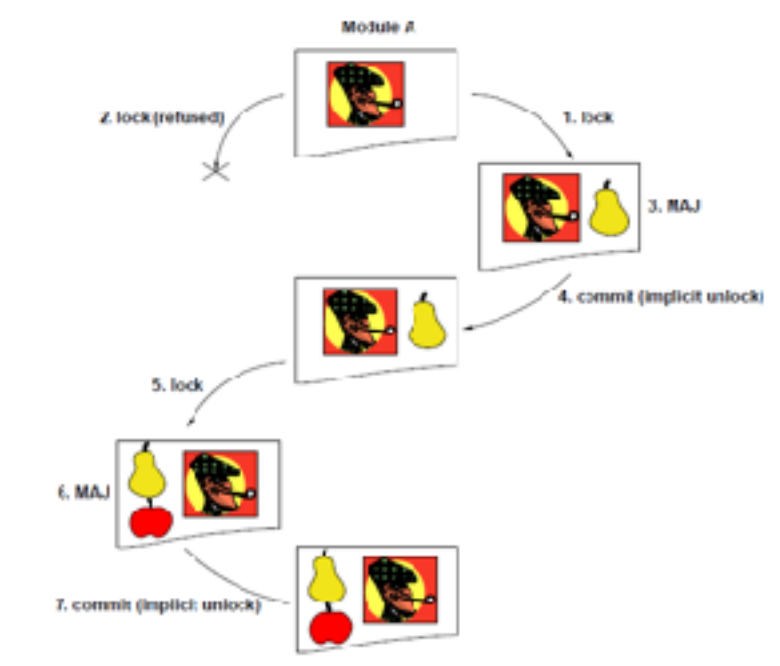
- Copy-Modify-Merge
- Lock-Modify-Unlock

Version Control: fundamental concepts

Parallel Development: How to manage independent developers making different changes to the same code?



Copy-Modify-Merge



Lock-Modify-Unlock

“Always Use Source Code Control”

From “The pragmatic programmer” by Andrew Hunt and David Thomas, 1999, an excellent advanced programming book:

> *“Always. Even if you are a single-person team on a one-week project. Even if it’s a “throw-away” prototype. Even if the stuff you’re working on isn’t source code. Make sure that everything is under the source code control — documentation, phone number list, memos to vendors, makefiles, build and release procedures, that little shell script that burns the CD master — everything. … Even if we’re not working on a project, our day-to-day work is secured in a repository.”*

Basic svn commands

svn checkout/co
svn add
svn delete
svn status
svn update/up
svn commit/ci
svn diff
svn move

Basic CVS commands

cvs checkout/co
cvs add
cvs remove
cvs log
cvs update
cvs commit
cvs diff
cvs tag
cvs release

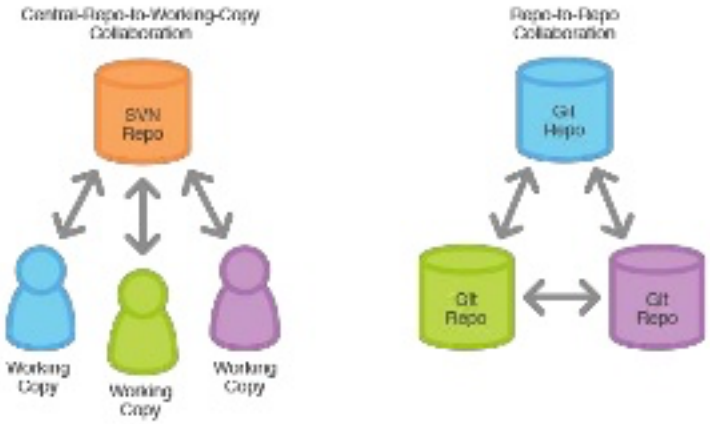
Warning: like languages with common syntax, do not assume a common semantics

svn or git?

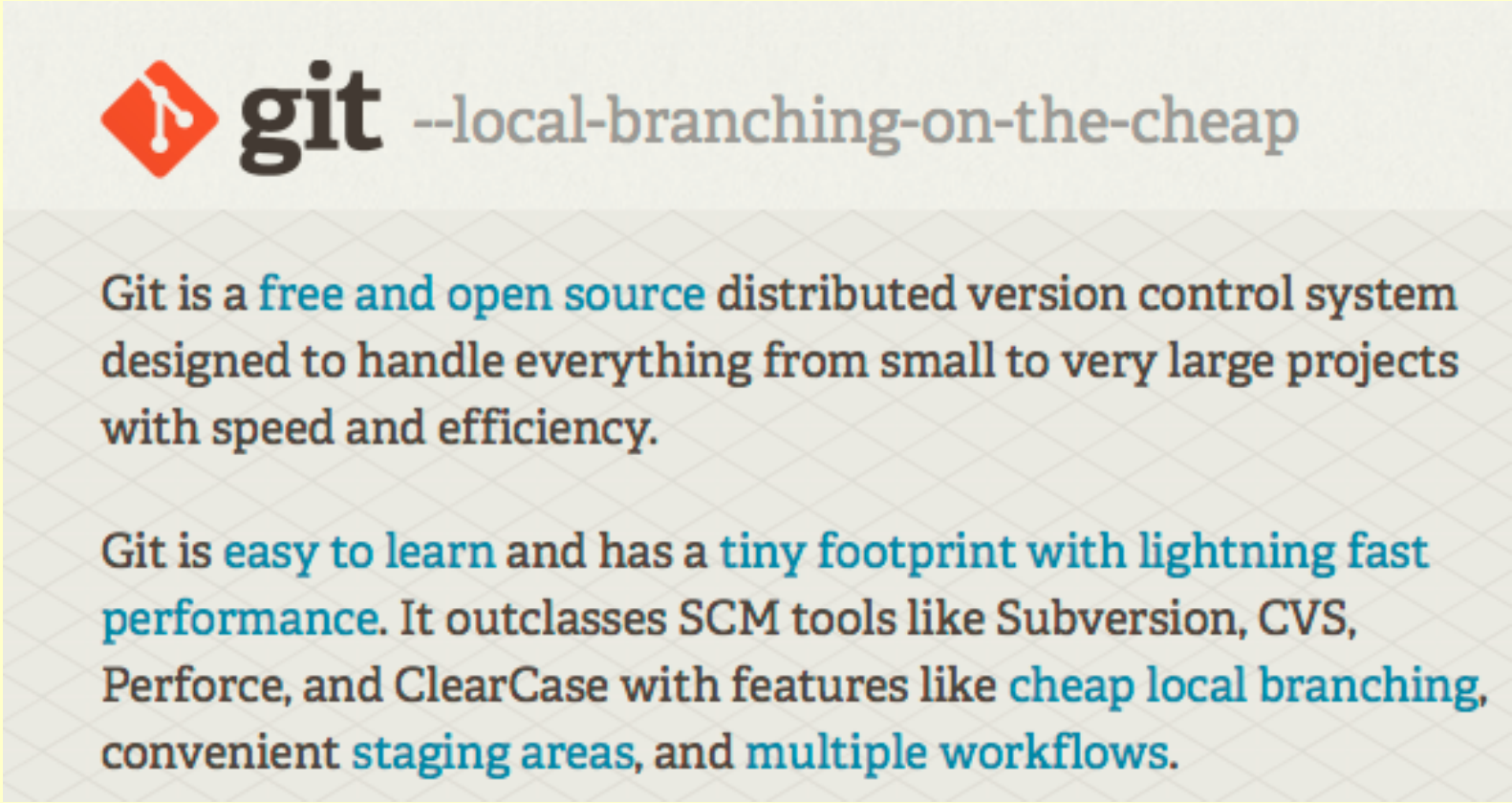
svn = remote

git = local

While with svn, everything routes through the remote repository, git introduces the notion of a “local” repository

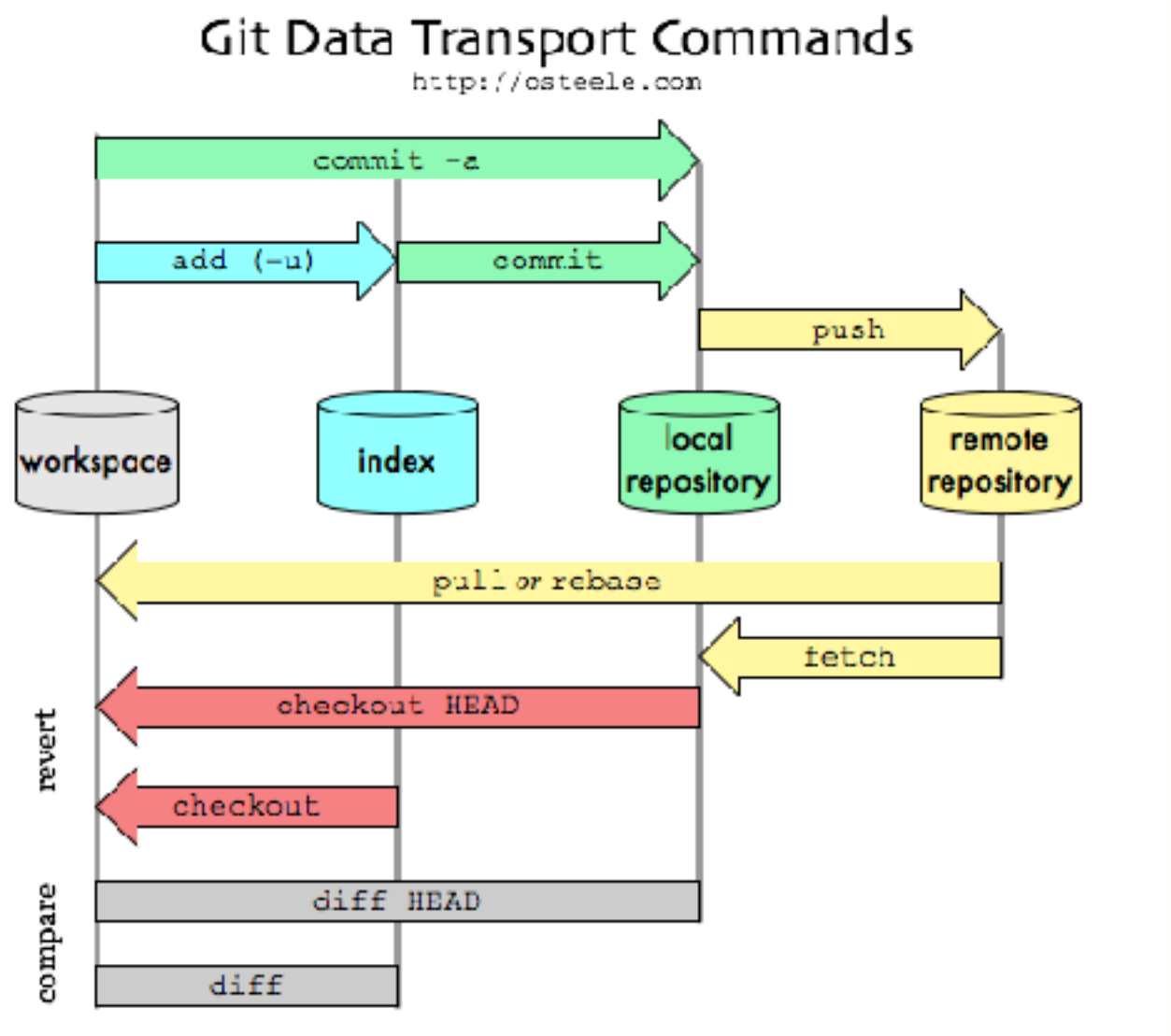


My preferred system: git

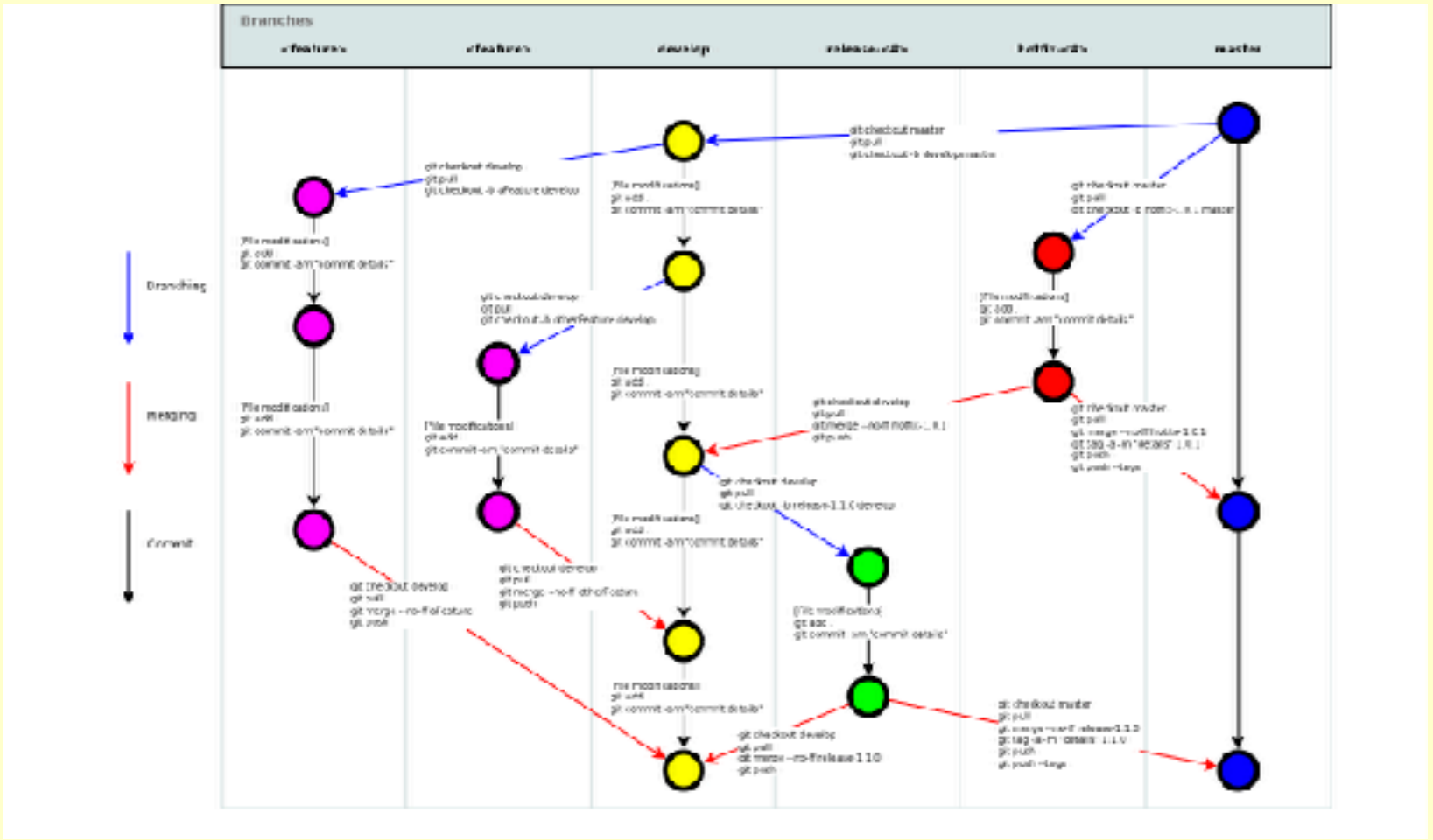


https://git-scm.com

The 3-stage data transport in git



A recommended git workflow



http://www.geekgumbo.com/2011/08/04/git-a-simple-workflow-commands-usage/

An on-line tutorial

http://rogerdudler.github.io/git-guide/