VCS FTW

Timofey Vasenin

Senior Software Engineer @ Xored,
MMD NSU graduate

VCS WTF

Timofey Vasenin

Senior Software Engineer @ Xored,
MMD NSU graduate

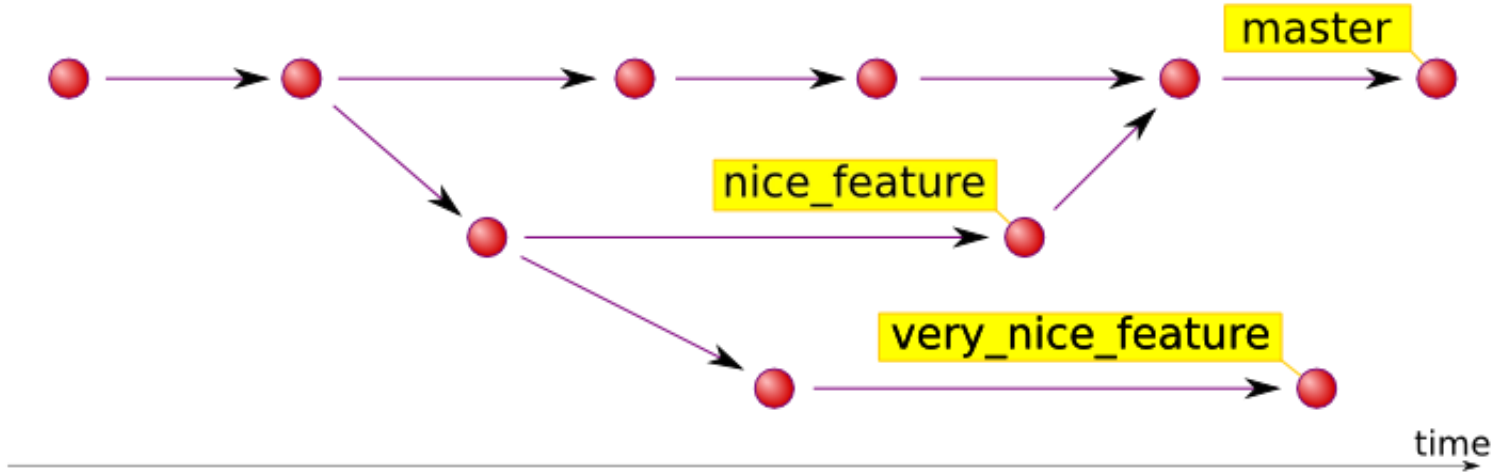
Contents

- Common terms and definitions
- VCS Evolution: **Local** -> **Centralized** -> **Distributed**
- A long journey: **ZIP** -> **CVS** -> **SVN** -> **GIT**
- GIT fundamentals

Linear history

Branching

- Commit (revision)
- Branch
- Merge

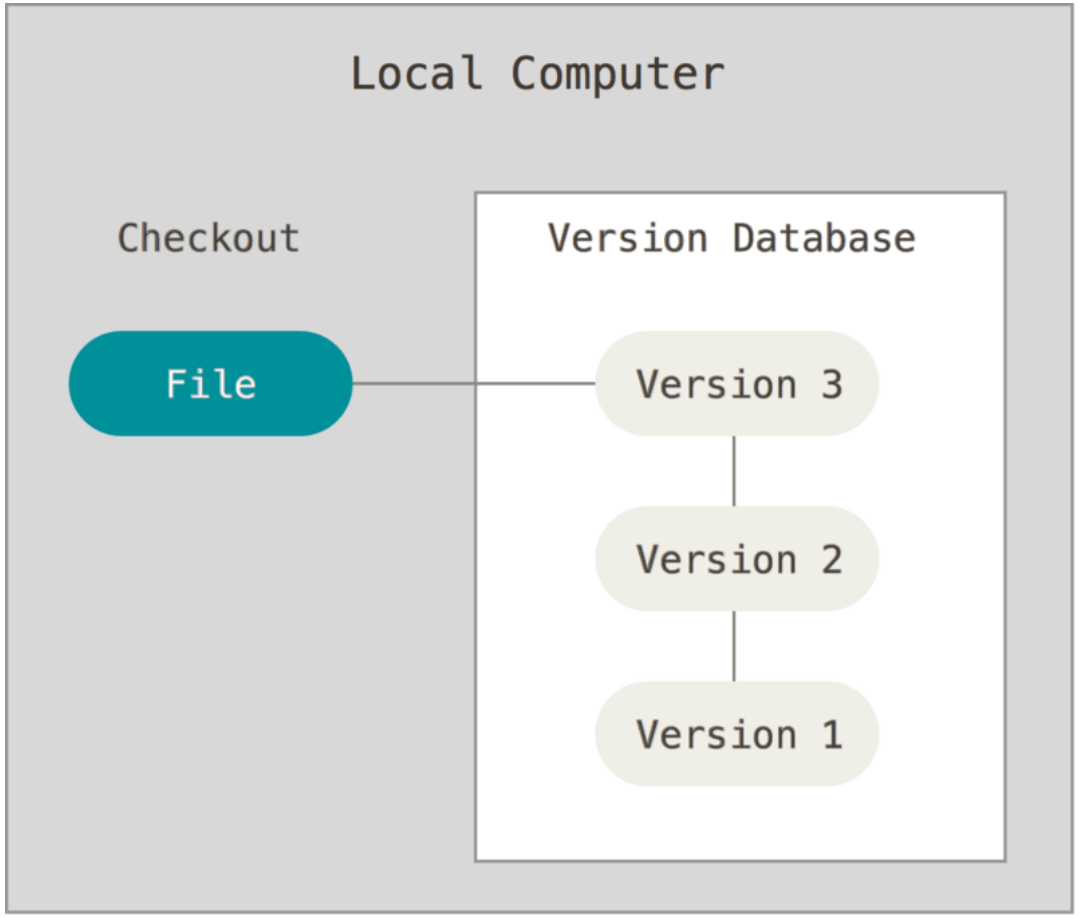


Version control system (VCS)

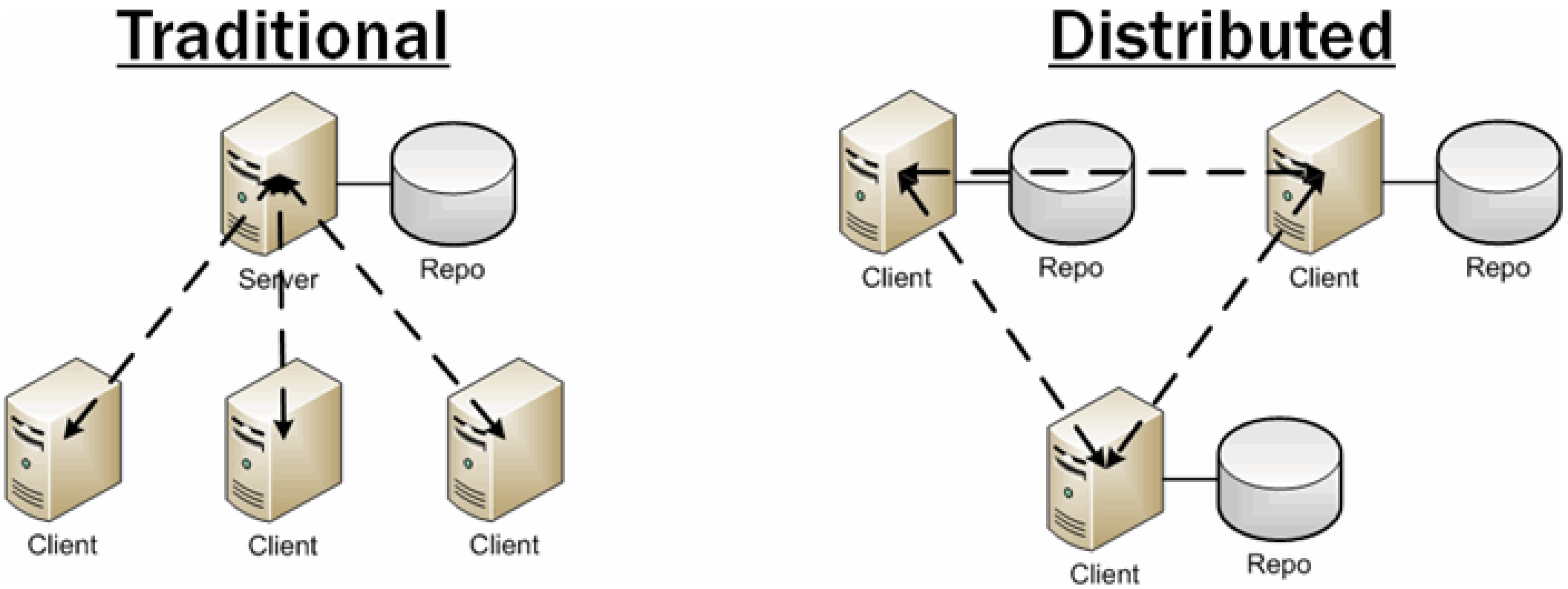
- Keep development history
- Enable concurrent development
- Show changes between versions
- Foundation for CI

- Can be used for anything, not only for code

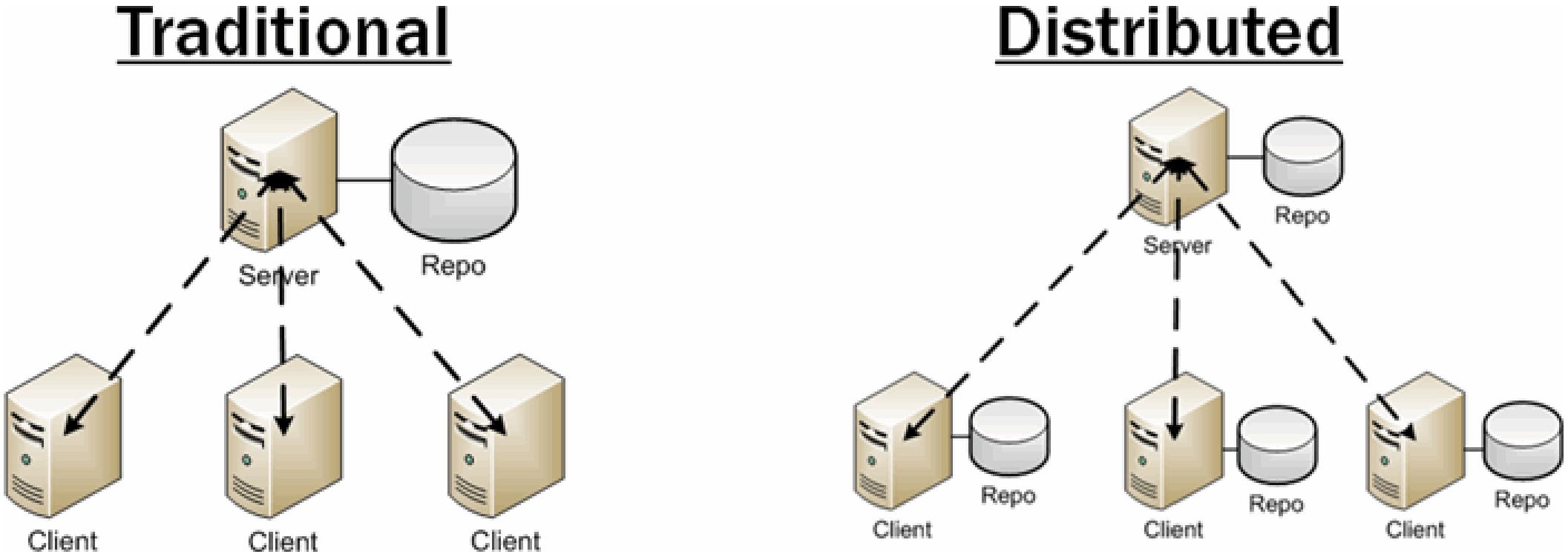
Evolution – Local VCS



Evolution – CVCS vs DVCS



Evolution – CVCS vs DVCS



ZIP + patches

- Snapshots + diffs
- Good for manual exchange
- Not space-efficient
- No way to easily combine patches

- Not obsolete!

CVS

I have seen

TERRIBLE things

SVN

- **/trunk**
- **/branches/featureX**
- **/tags/vX.XX**

- Cheap copy/move (incl. branches/tags)
- One SVN server can host multiple repos

SVN

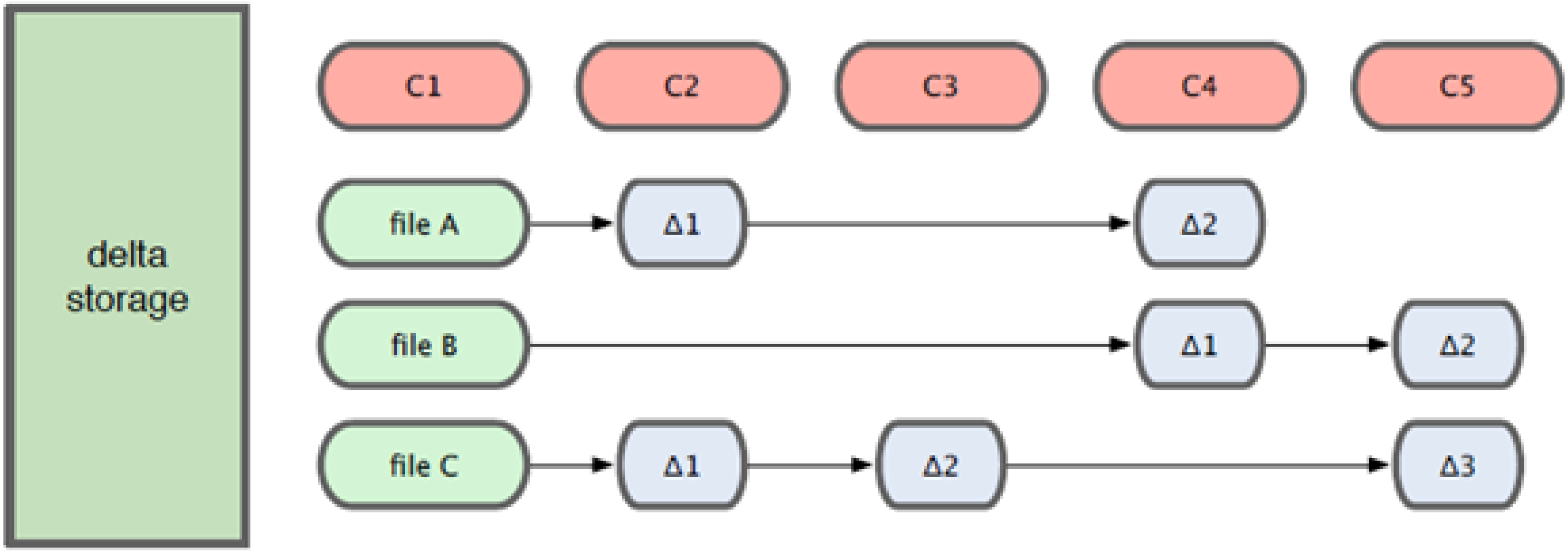
- Everything is about **naming conventions**
- No real branches/tags (just different folders)
- Easy to abuse
- SLOW

- Not obsolete!

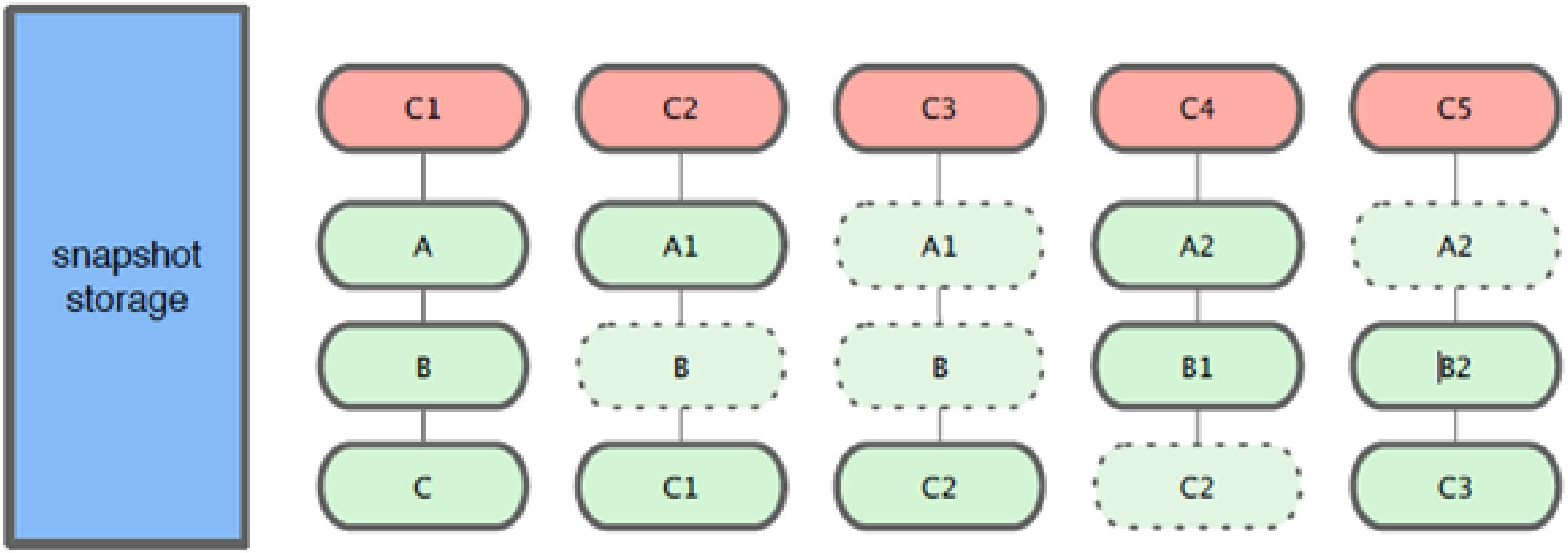
Git – design goals

- Patching should take no more than three seconds
- Take CVS as an example of what *not* to do; if in doubt, make the exact opposite decision
- Support a distributed, BitKeeper-like workflow
- Include very strong safeguards against corruption, either accidental or malicious

List of **deltas**



Stream of **snapshots**

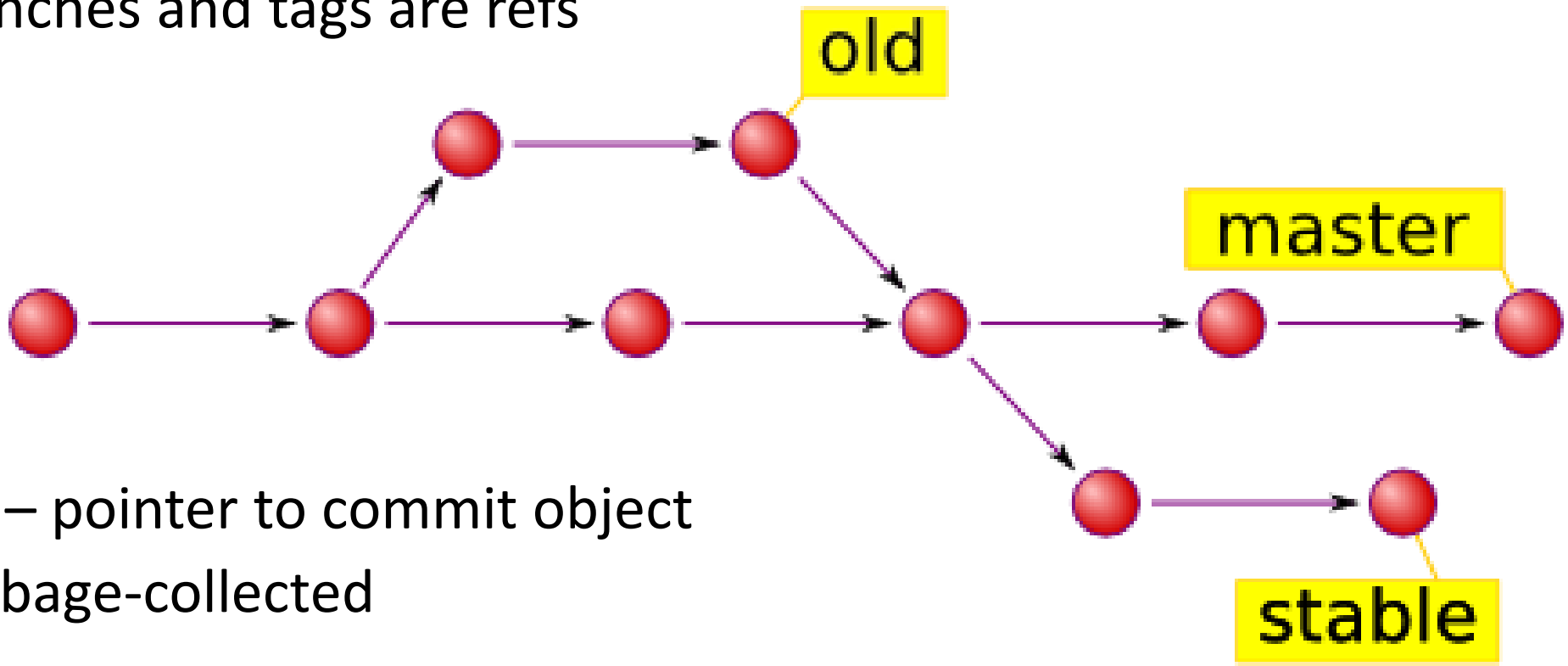


Git – overview

- FAST
- Distributed
- Simple data model
- Most operations are local
- Branching development strategy
- Almost no overhead for branches/tags
- Each commit is a snapshot of the entire project
- It’s **important** to know how it works under the hood

Git – commits and refs

- Commits store their parent IDs
- Branches and tags are refs



- Ref – pointer to commit object
- Garbage-collected

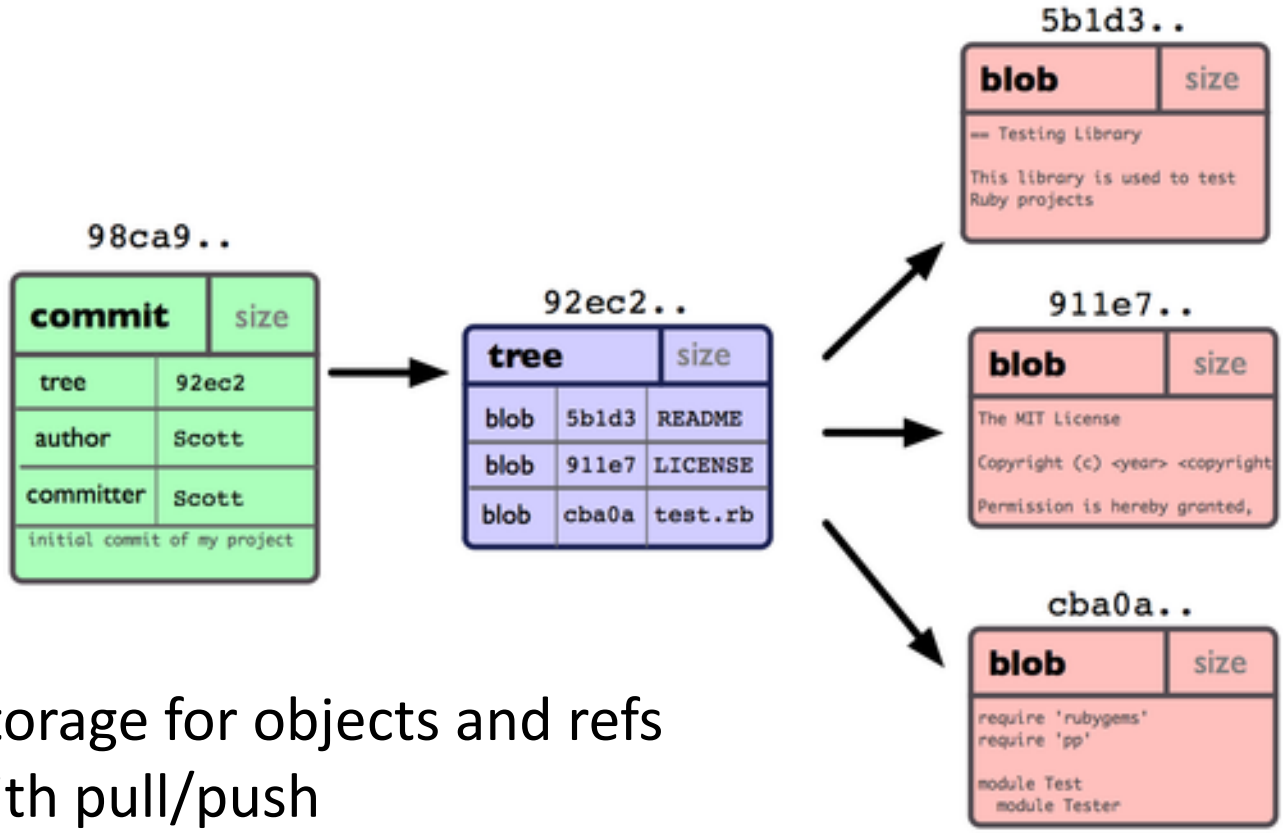
Git – commit anatomy

Blob

Tree

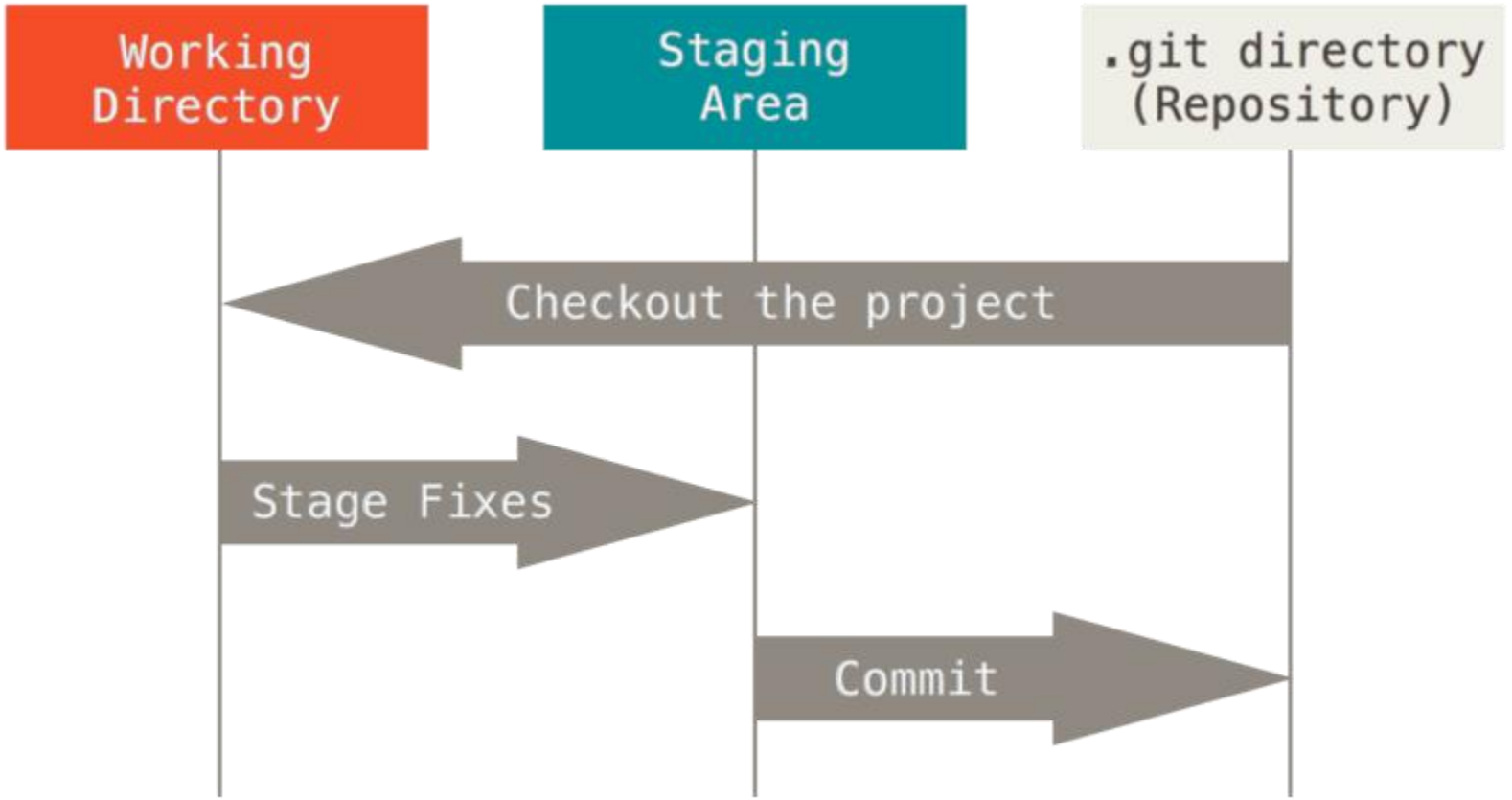
Commit

Tag



- Repository – storage for objects and refs
- Synchronize with pull/push

Git – 3 states


Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

Git – limitations

- Can’t rewrite history without changing IDs of all commits involved
- One repo – one project
- Not easy to sync multiple repos
- No native way to effectively store large binaries

Git – code of conduct

- Config files (**.gitignore**, **.gitattributes**)
- Branch/tag naming conventions
- Commit message rules (!)
- Branching strategy (e.g. GitFlow)

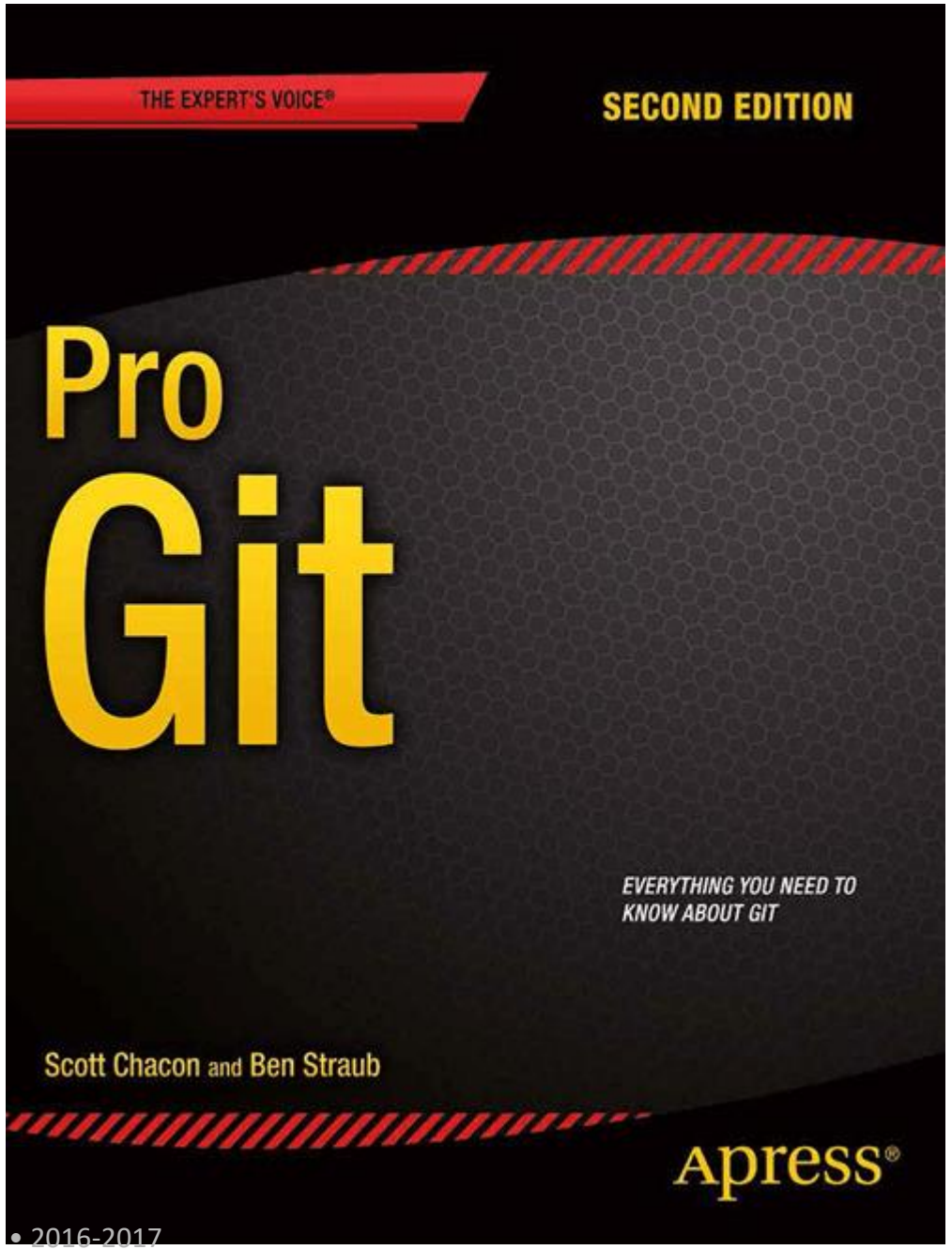
Git – learning

- Start off with GUI clients
  - **SourceTree**
  - GitK
  - IDE plugins

- Try to imagine the result before doing actual work

- Experiment!

https://git-scm.com/book



Links

- http://chris.beams.io/posts/git-commit/
- http://www.slideshare.net/DrupalForumZP2012/vcs-git
- http://nvie.com/posts/a-successful-git-branching-model/
- http://www.slideshare.net/tarkasteve/understanding-git-goto-london-2015
- https://illustrated-git.readthedocs.io/en/latest/