Git

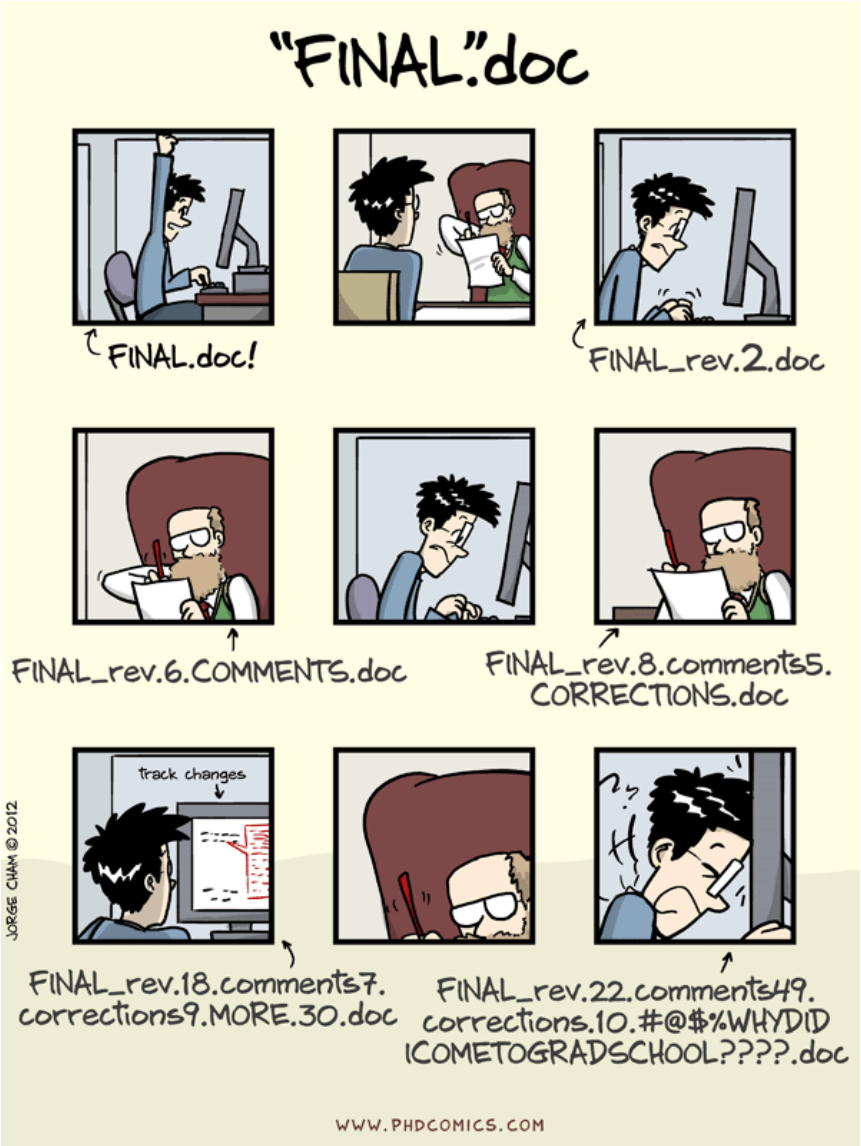
Amir Masoumzadeh

CSI 333 – System Fundamentals

UNIVERSITY
AT ALBANY
State University of New York

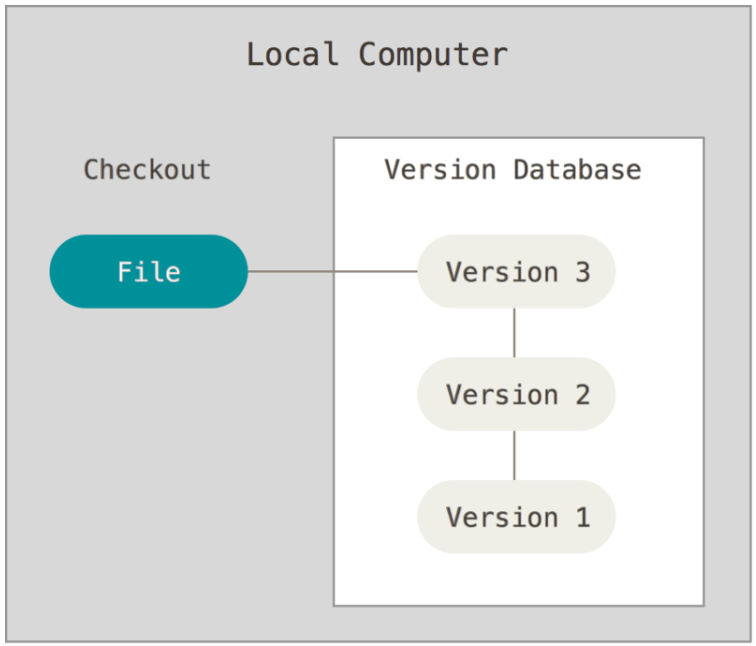
Aug. 24, 2023

No Version Control



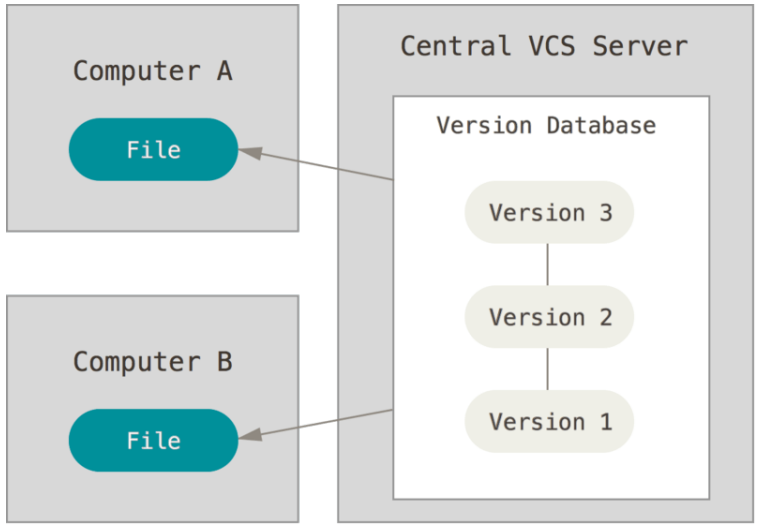
Local version control

- Simple database keeps all changes to files
- Example: RCS (Revision Control System)
  - Keeps patch sets (difference between files)
  - Can reproduce a file at different time points



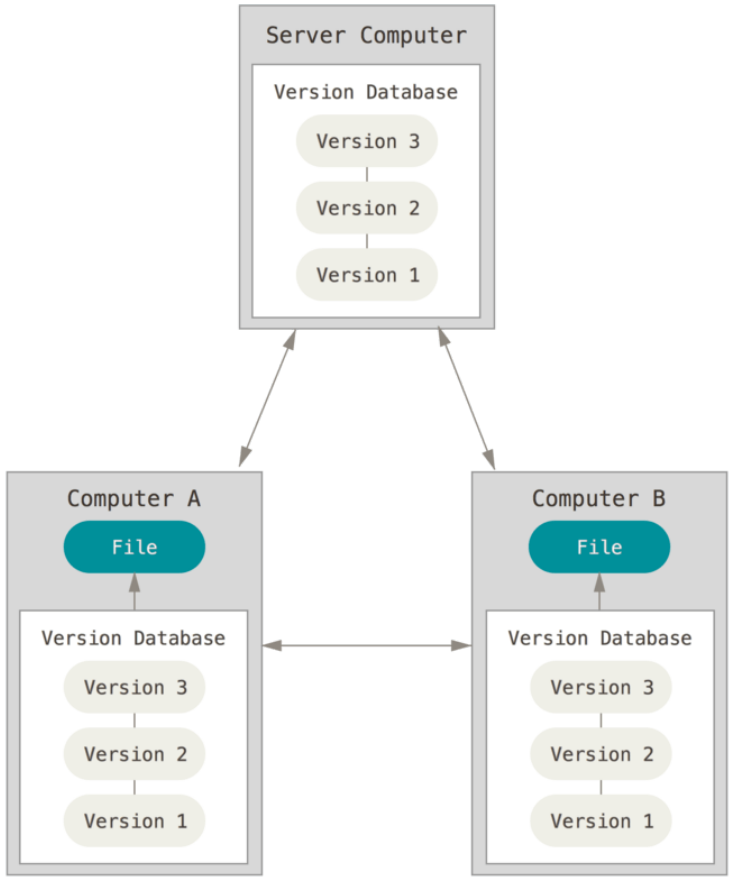
Centralized version control

- Examples: CVS, Subversion, Perforce
- Central server repo holds "official copy"
  - Also sole version history of the repo
- You make "checkouts" of it to your local copy
  - You make local modifications
    - Your changes are not versioned
- When done, you "check in" back to the server
  - Your checkin increments repo's version
  - What if someone has already submitted a new version?



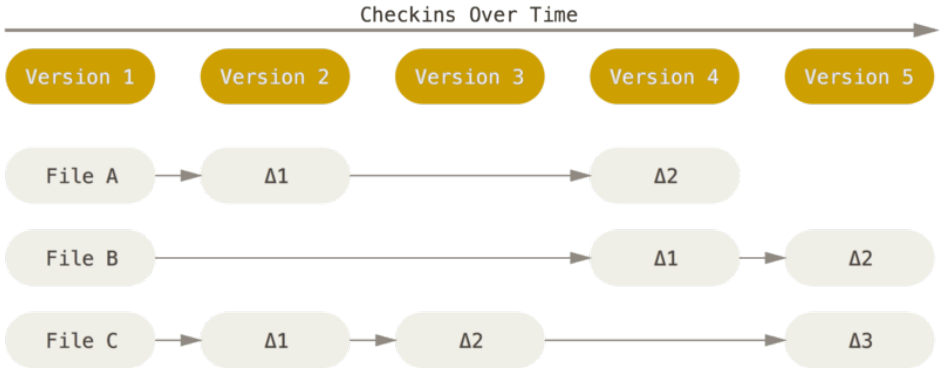
Distributed version control

- Examples: Git, Mercurial, ...
- You "clone" from a remote repo and "pull" changes from it
- Your local repo is a complete copy of everything on the remote machine
  - Yours is just as good as theirs
- Many operations are local:
  - Check in/out from local repo
  - Commit changes to local repo
  - Local repo keeps version history
- When you're ready, you can "push" changes back to remote machine

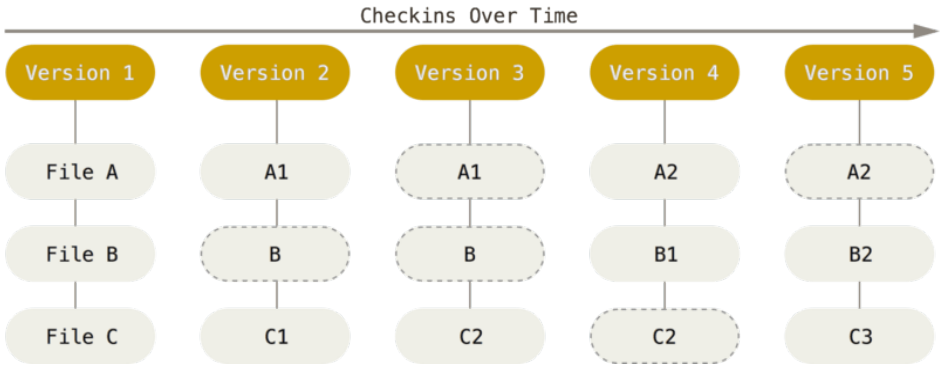


Git snapshots

- Centralized VCS's track version data on each individual file



- Git keeps "snapshots" of entire state of project
  - Each checkin version of overall code has a copy of each file in it
  - Some files change on a given checkin, some do not
  - More redundancy, but faster



Git: local, integrity-checked, immutable history

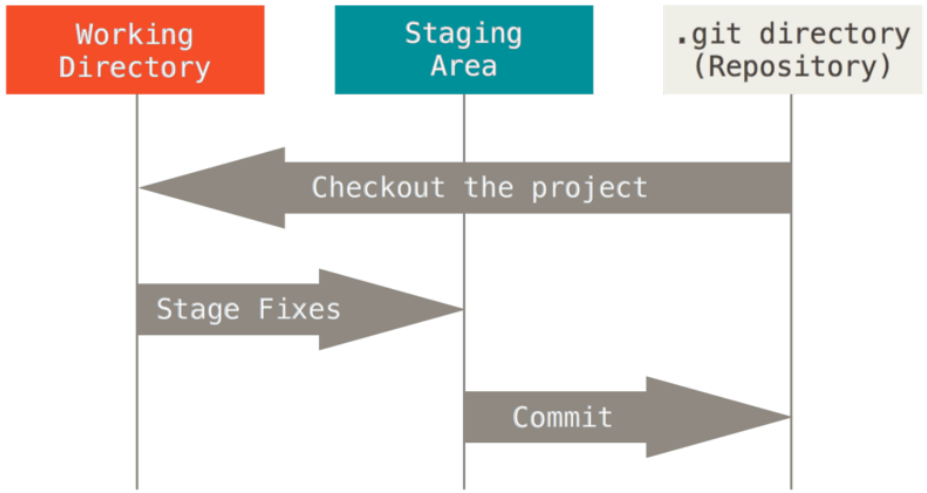
UNIVERSITY AT ALBANY
State University of New York

- You have all version history in your local repo
  - You can retrieve and modify it independently
  - No need to communicate to another server
- Everything is checksummed before storage
  - Checksum numbers used for reference
  - Git uses SHA-1 hash for checksumming
- Git generally only adds data

Git states

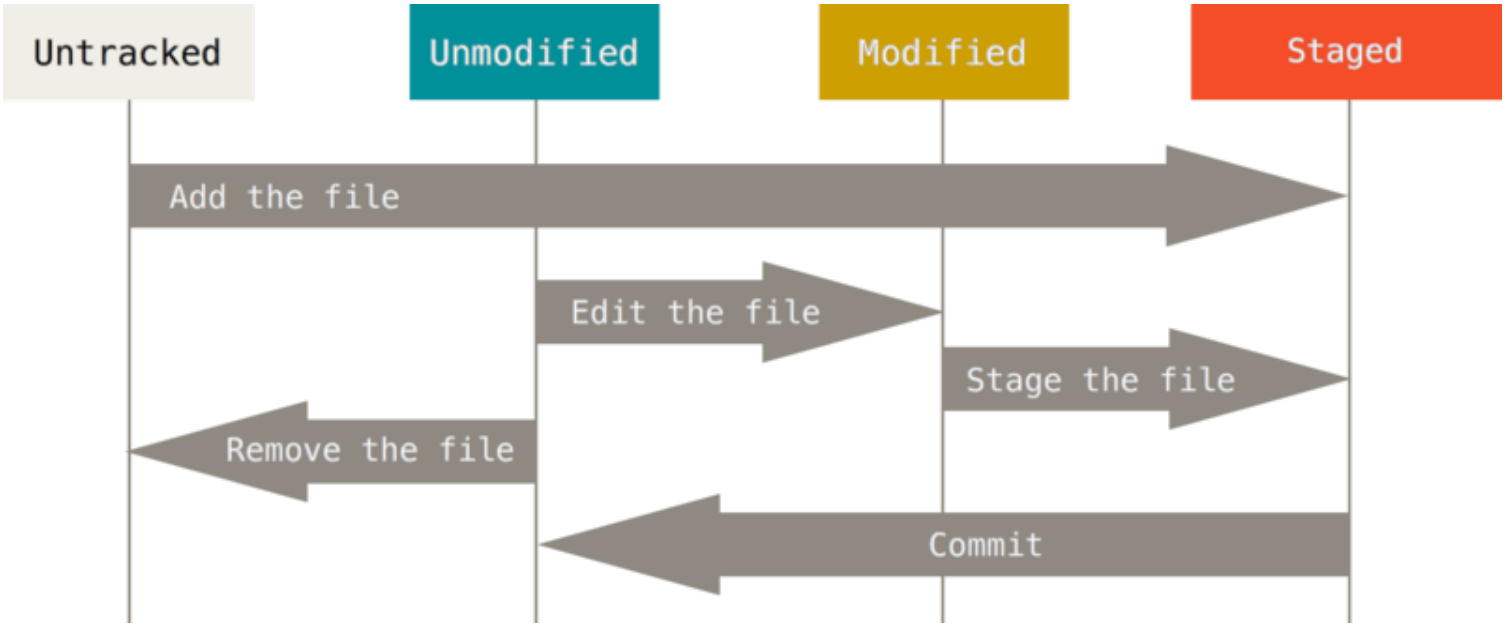
On your local machine files can be:

- In your local repo (committed)
- Checked out and modified, but not yet committed (working copy)
- In-between, in a "staging" area
  - Staged files are ready to be committed
  - A commit saves a snapshot of all staged state



Basic Git life cycle

- Modify files in your working tree
- Stage files (selectively), adding snapshots of them to your staging area
  - Also, track files in your working directory
- Commit, which takes staged files and stores that snapshot permanently to your Git directory



Basic Git life cycle: commands

Note: These are git sub-commands

- `init`: create an empty Git repo
- `status`: show working tree status
- `add`: stage a file
- `commit`: record staged changes to repo

Other commands you may use less frequently:

- `log`: show commit logs
- `diff`: show changes between commits, commit & working tree, etc.
- `rm`: remove files from working tree and index
- `checkout`: switch branches or restore working tree files

Git: working with remote repositories

UNIVERSITY AT ALBANY
State University of New York

- `clone`: clone a repo (create a local duplicate)
- `remote`: manage set of tracked repos
- `fetch`: download objects from remote repo
- `pull`: fetch and merge with local repo
- `push`: update remote repo

Exercise ex2

UNIVERSITY AT ALBANY
State University of New York

- Retrieve repo from:
  `https://github.com/ualbany-csi333-f23/inclass-git`
- In which snapshot *random.txt* contains random number 25083?
- Record the first 6 letters of SHA-1 hash code for the commit
- Hint: See section on "diff formatting" within `git log` man page
- Questions:
  1. Partial hash value indicated above?
  2. Commands you need to run to get to the answer (starting from retrieving the repo)?

Using Git for lecture slides

UNIVERSITY AT ALBANY
State University of New York

- Suggestion: create a folder for this course (e.g., `~/csi333-repos`) and clone/create repos within that folder
- Clone our lectures repo: `https://github.com/ualbany-csi333-f21/lectures`
  - Then, just pull it each time to retrieve the new lectures!

Using Git for homework assignments

UNIVERSITY AT ALBANY
State University of New York

- On release date of homework
  1. Check the repository of the homework
  2. Follow the link there to create a private repo for you for that specific homework: e.g., `lab01-azerrima`
- For working on your homework, you
  1. Clone your homework repo (e.g., `lab01-azerrima`, not `lab01`)
  2. Create/edit assignment files
  3. Add changes
  4. Commit
  5. Push (and go to 2 if needed)
- At the deadline, we
  1. Clone your repo to our machine
  2. Grade it
  3. Push it as a separate "graded" branch in your repository that contains feedback/grade