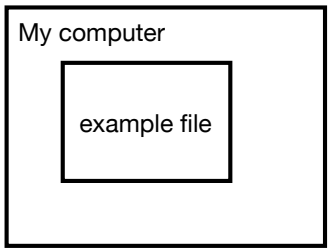
Home-baked Version Control ("low" tech)



This is the system in which you prefix your file names

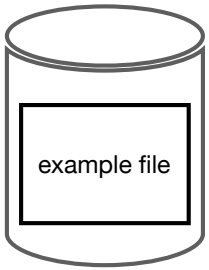
pros

lightweight
easy to establish
no merge conflicts
user has full control

cons

users are bad at consistency
humans are forgetful
accidental overwriting
All on one computer/ one location
scalability

Autocratic Version Control

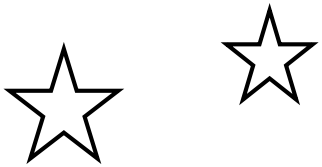


pros

access from remote
no merge conflicts

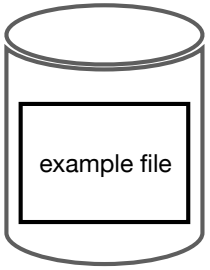
cons

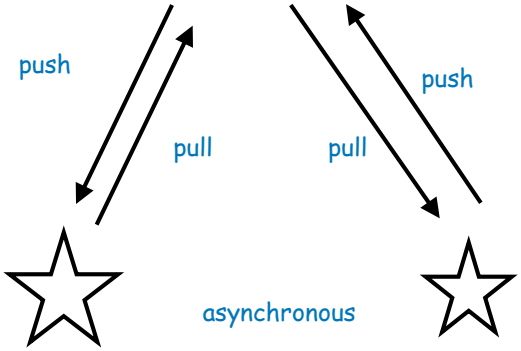
only one person has write access at a time



Centralized Version Control (subversion, svn, perforce)

one master remote repository





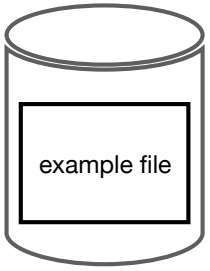
pros

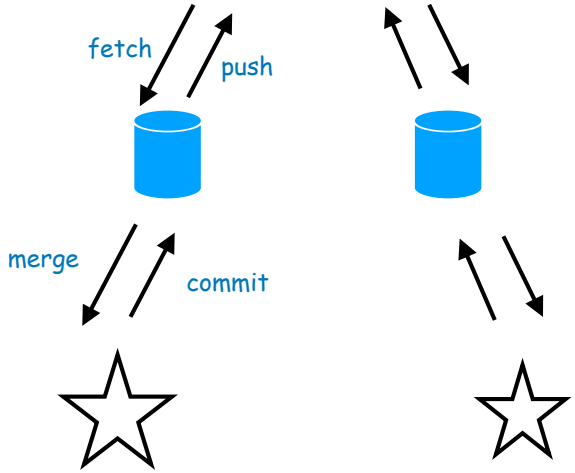
can checkout a subtree of the repository

cons

Distributed Version Control (git, mercurial, bazaar)

Users also maintain their own local repositories





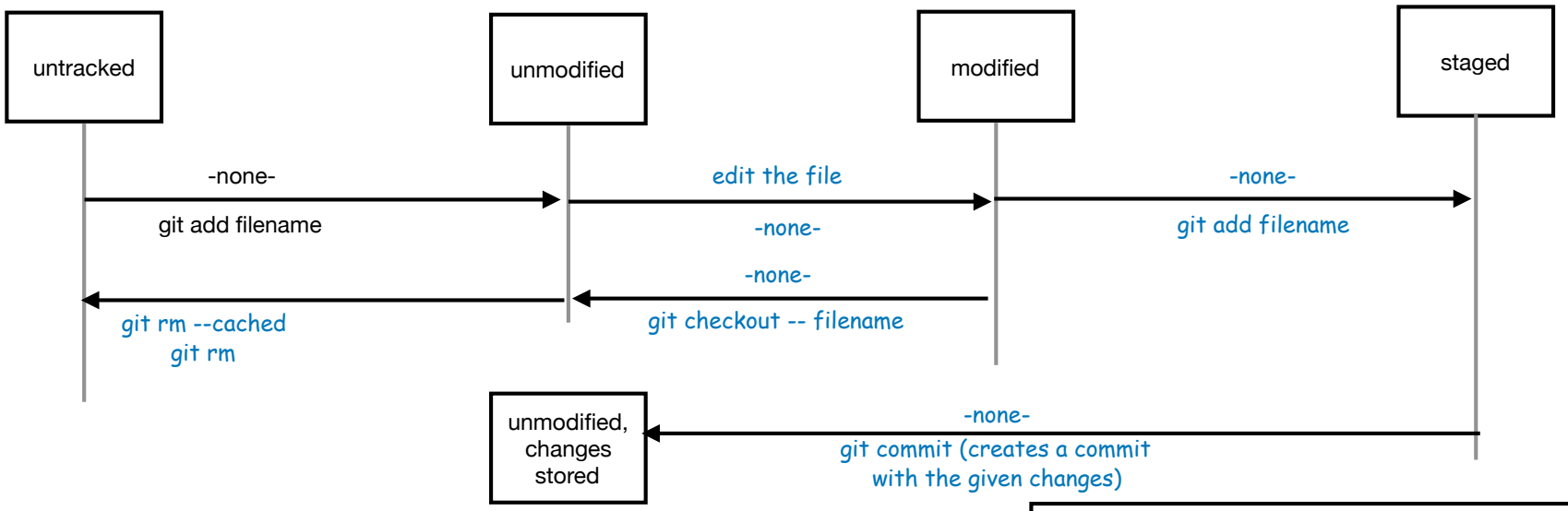
pros

all users have all parts of the repository

cons

merge conflicts
branching logic can become extremely complicated
requires users to have full copies of the repositories locally (bigger)

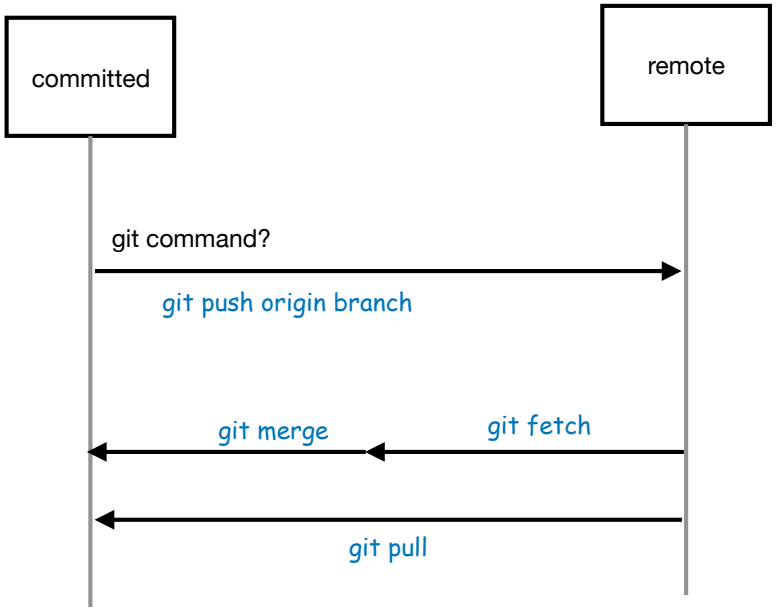
states of a file in git



What is the state of the files after commit?

files are indexed and a checkpoint has been created in the log (they are still local though)

Label these arrows are follows: how (file manipulation)? → git command?



What are the steps for fixing a merge conflict?

1. (before conflict you should have committed your changes) git pull
2. edit the files that are indicated in the failed git merge, putting the file in the state that you'd like it to ultimately be in
3. git add the modified files
4. git commit the modified files
5. git pull (make sure that there aren't more changes to merge)
6. git push