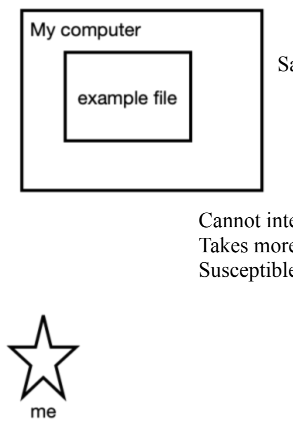
Felipe Lima

Question 1

1

Home-baked Version Control
("low" tech)



pros

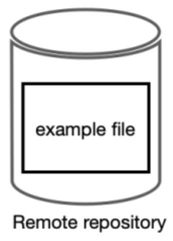
Safer from attacks

cons

Cannot interact with others well
Takes more memory
Susceptible to crashes

2

Autocratic Version Control



pros

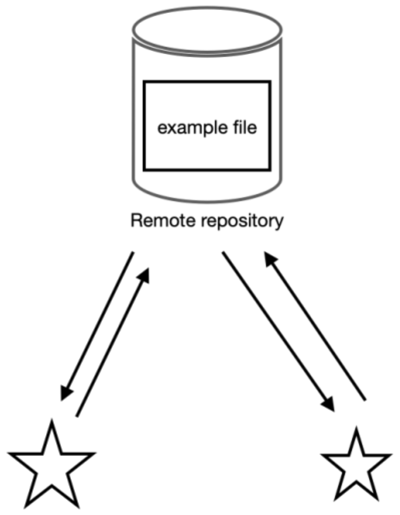
There is a backup for the files
No inconsistence or differences on the files

cons

Reading and writing are locked to one user

3

Centralized Version Control
(subversion, svn, perforce)



pros

Everyone has access to the file at the same.
Multiple levels of access.

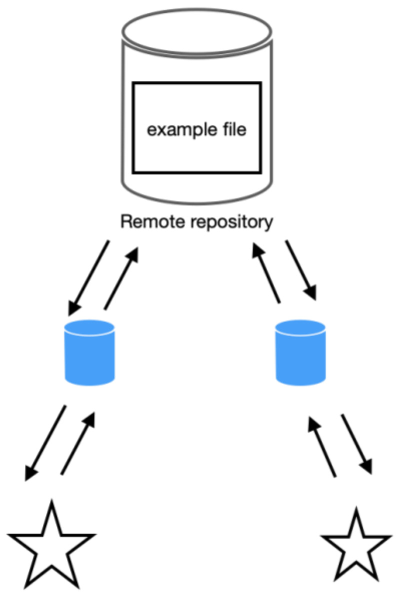
cons

If central repo goes down, no one has access to the file anywhere.
Need internet connection to commit

X

4

Distributed Version Control
(git, mercurial, bazaar)



pros

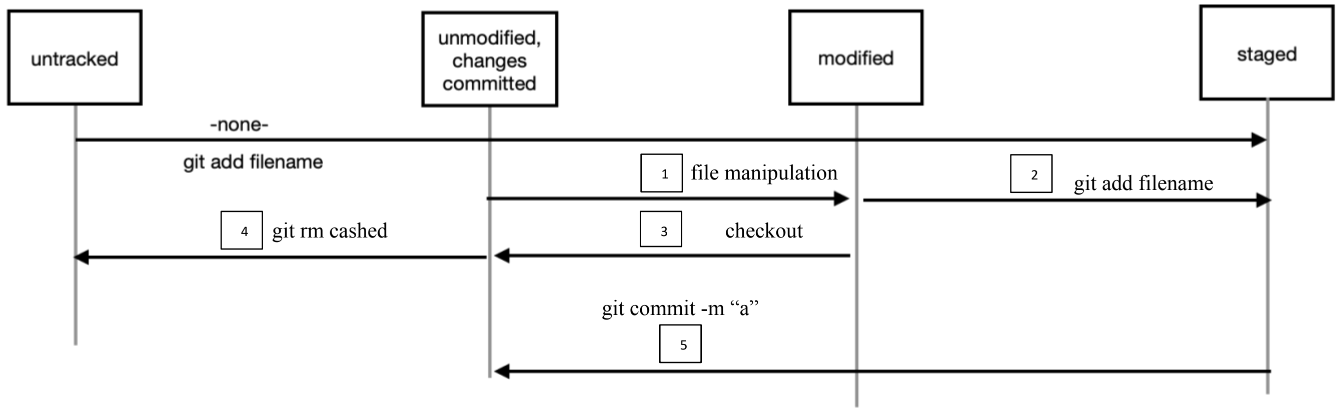
Can commit to local to then push to remote.
Your code is separate from others so you can change and not mess up anyone else'.

cons

Collaborators don't have access to your code all the time.
Can get complicated with branches and conflicts (more work).
If not continuously pulling from remote repo, your local can up outdated.
Does not have multiple levels of access - you have to clone the entire repo.

Question 2

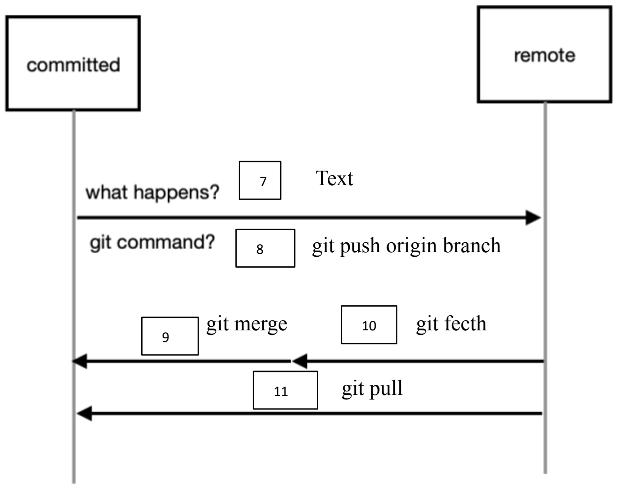
states of a file in git



Label these arrows are follows: how (file manipulation)? $\longrightarrow$ git command?

What is the state of the files after commit?

6 Index, cataloged and ready, local repo but not remote



What are the steps for fixing a merge conflict?

12
commit
pull
discuss
add
commit
pull
push the resolved

X