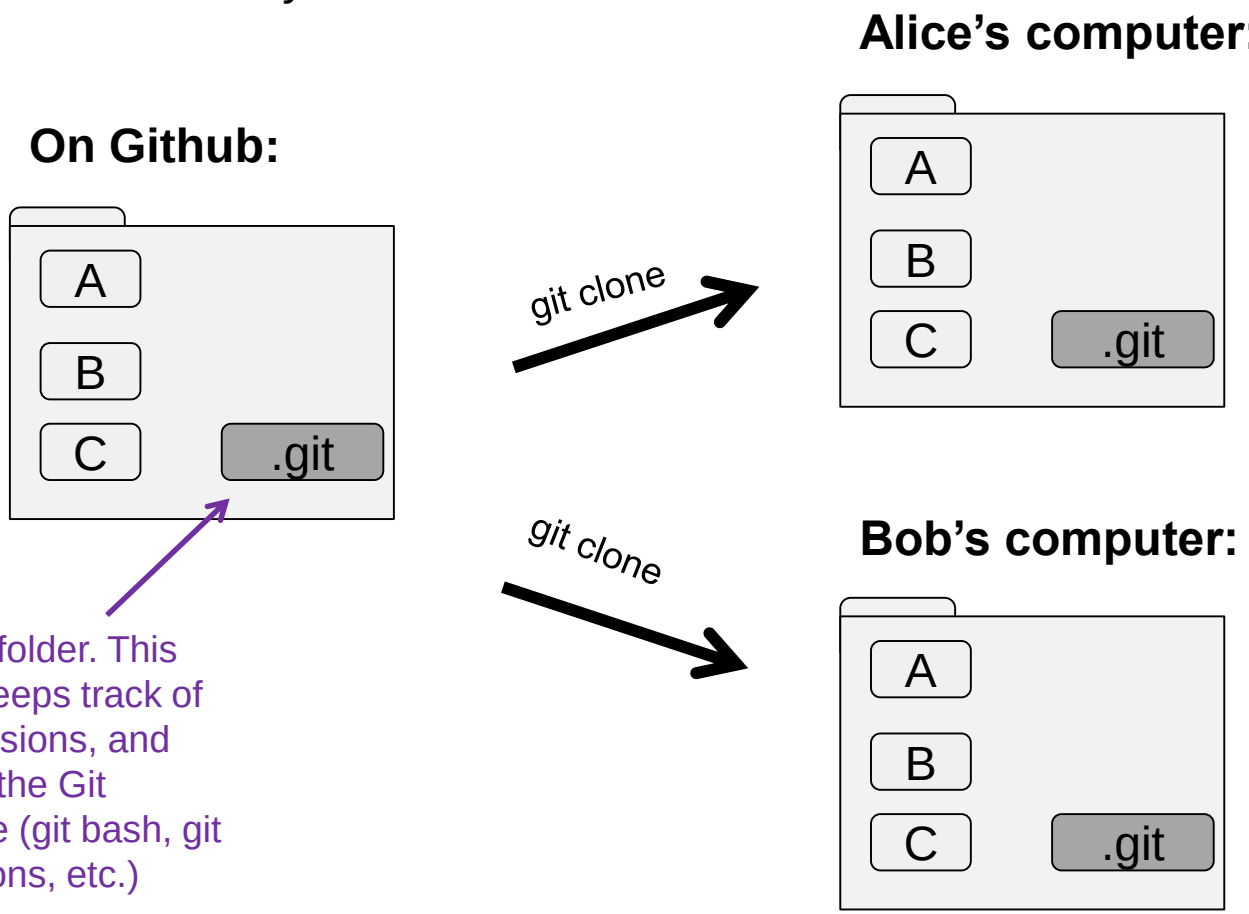
# Git and GitHub for Complete Beginners

**Neal Humphrey**

August 2014

# What is Git?

Version control system for a project (a collection of files i.e. a folder). Everybody gets a copy of everything – all current files, all past files, and a snapshot of how they are related to each other.

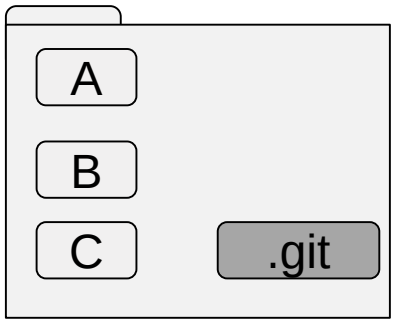


# What happens when you edit files?

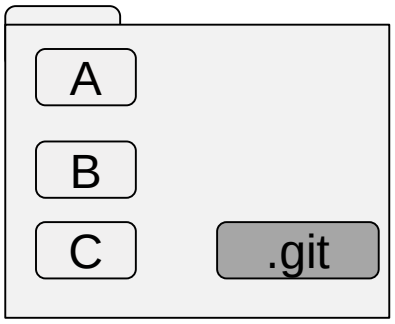
**Alice's changes and adds files. She clicks 'File-save' and on her hard drive they look like this:**


**Github and Bob don't know:**

**On Github:**

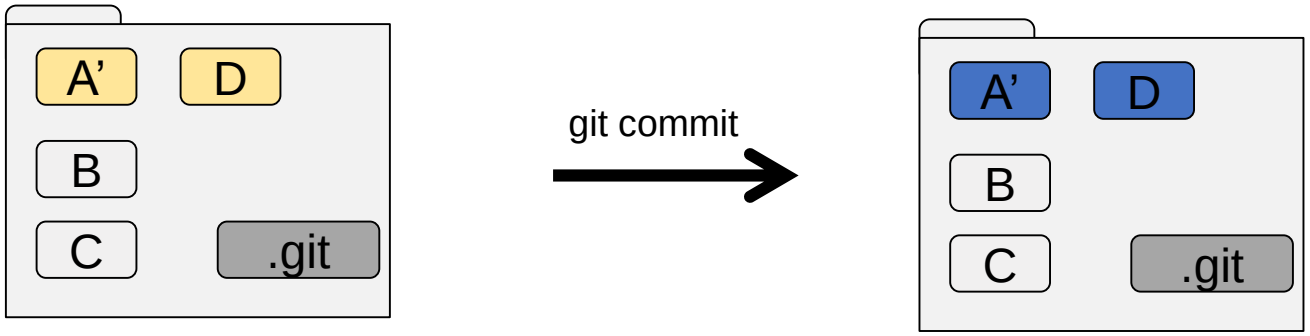



**Bob's computer:**



# What happens when you edit files?

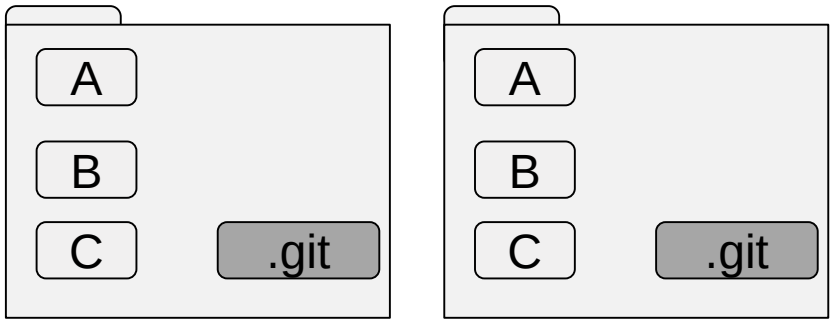
## Alice tells her git to store a snapshot




## Github and Bob still don’t know

**On Github:**

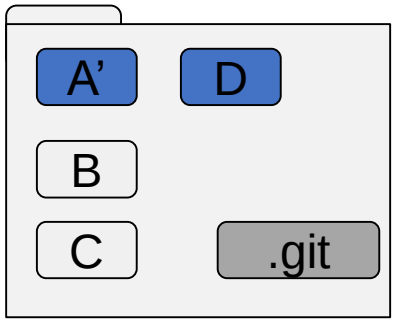
**Bob’s computer:**



# What happens when you edit files?

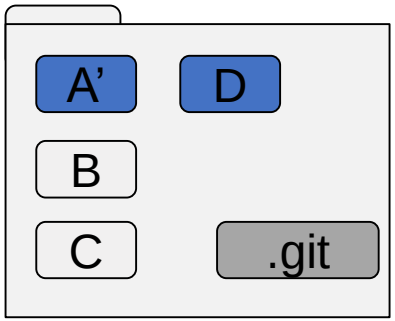
**Then, Alice tells Github about her new snapshot**

**Alice's computer:**



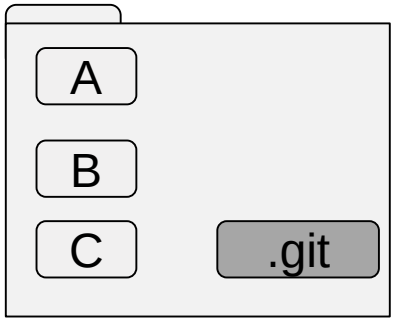

git push

**On Github:**



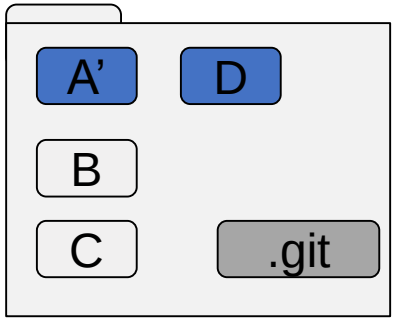

**Bob still doesn't know**

**Bob's computer:**

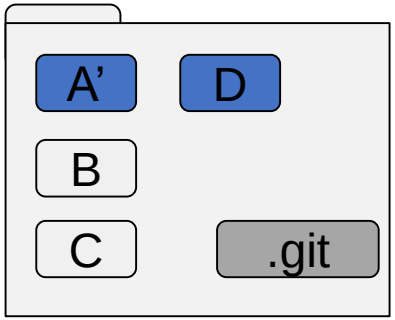


# What happens when you edit files?

**Alice's computer:**

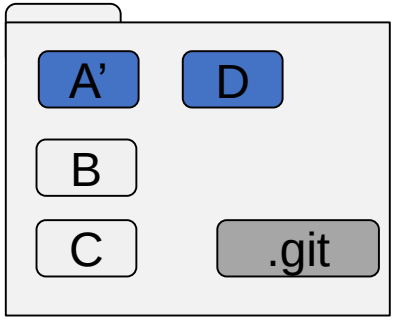


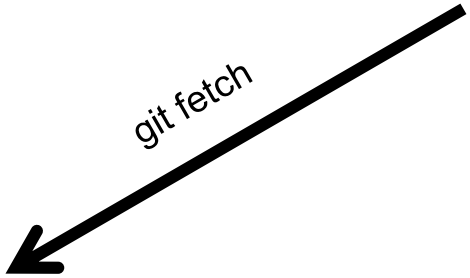

**On Github:**




**Bob asks Github what the most recent copy is**

**Bob's computer:**



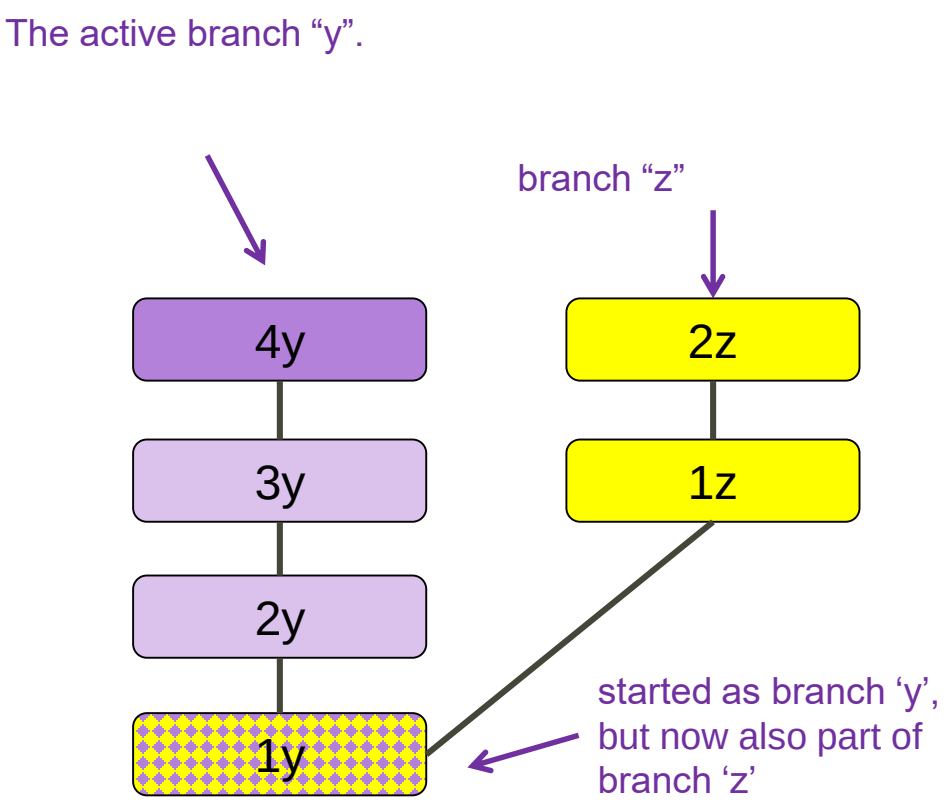



# Basic Git Concept: Repositories

- Git stores snapshots of a whole folder of files
- Everyone has a copy of the entire history of snapshots in their (hidden) .git folder, and the Git interface (Git Bash, Git Extensions, etc.) talks to this folder to manage changes.
- You have to tell git when to take a snapshot, when to tell other people about your new snapshot, and ask Github for the latest snapshot. These are the relevant commands:
  - git commit
  - git push
  - git fetch <-this gets the data about external branches, but doesn't actually merge the changes in.
  - git pull <- this merges external commits into your local branch

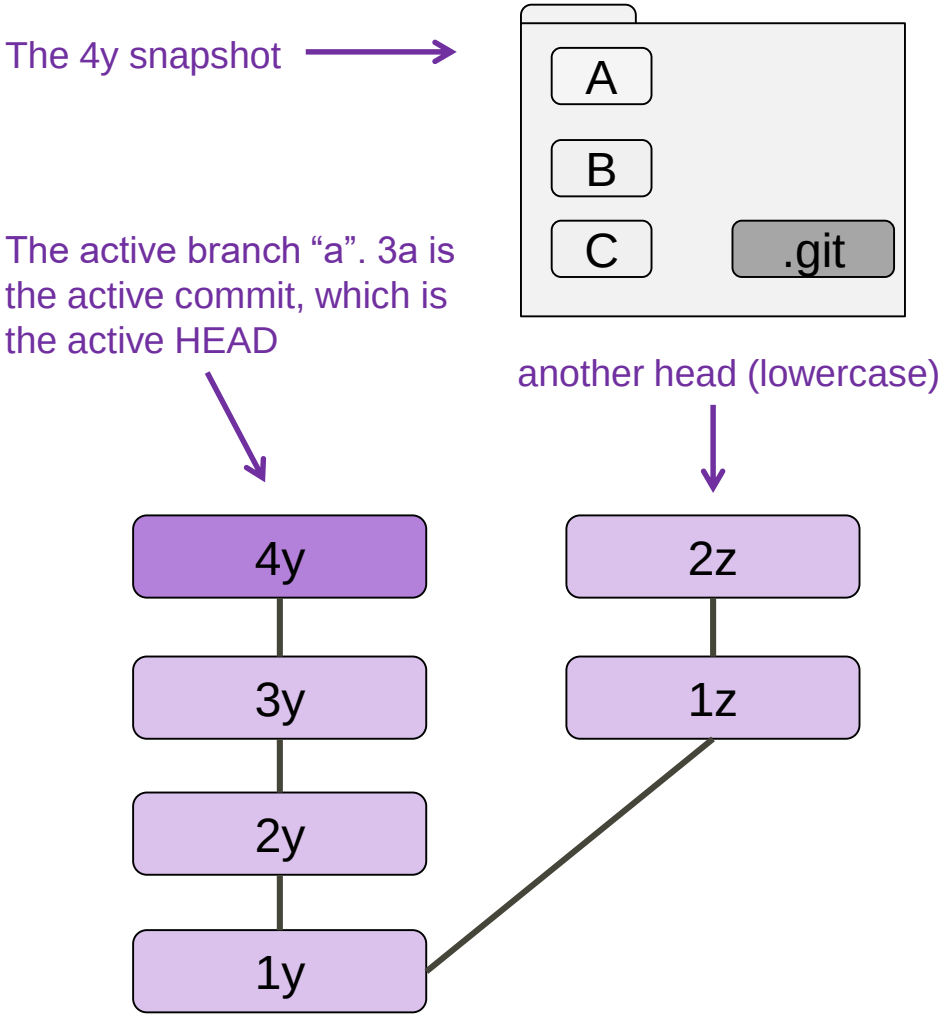
# So what's up with these snapshots?

- Storing a snapshot is 'to commit' (verb). You also refer to each snapshot as 'a commit' (noun).
- A string of commits is called a branch
- Each commit has a parent commit, but two commits can have the same parent (i.e. a new branch).
- Each commit has a hash (automatic unique identifier) and a message (description).



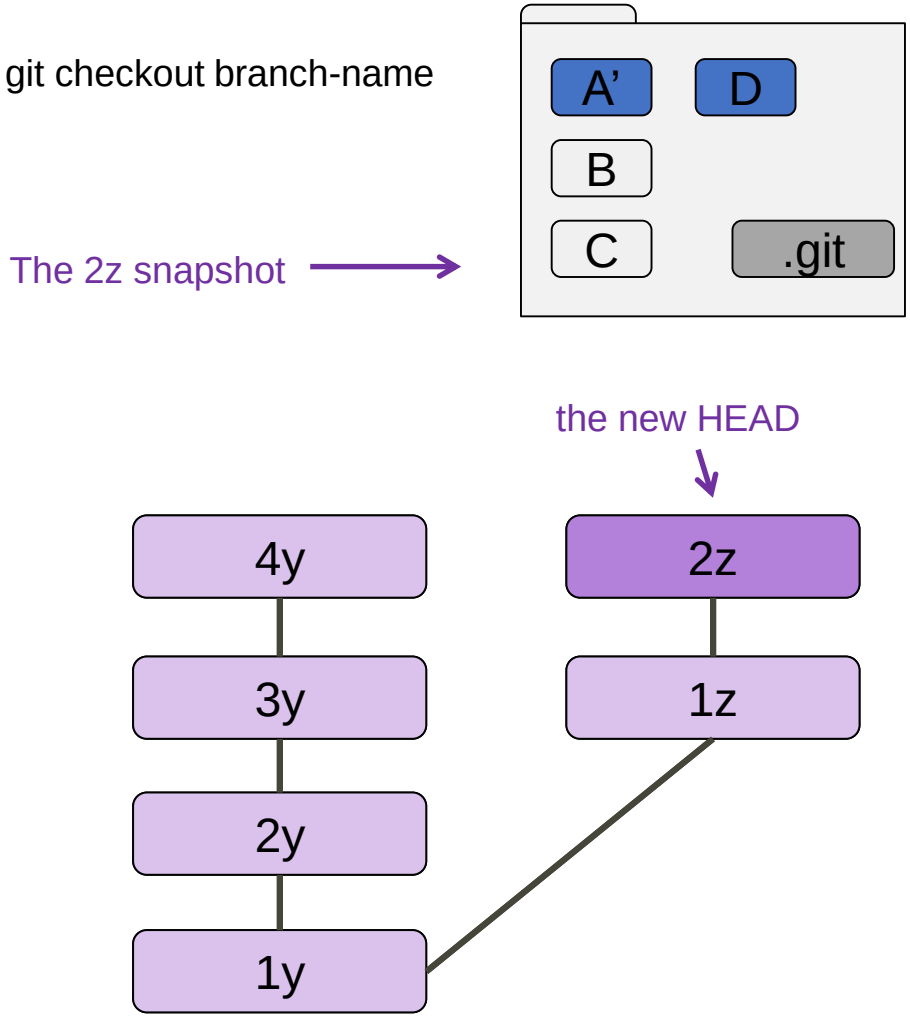
# So what's up with these snapshots?

- Each branch has a 'head'
- The active head (the HEAD, all caps) pulls the relevant content out of the .git file and replaces the contents of the rest of the folder with this snapshot.



# So what's up with these snapshots?

- Changing branches changes to that branch's head (the most recent commit)
- Git changes the actual content of the folder – deleting and adding files as needed. You can restore the old version by switching branches again.

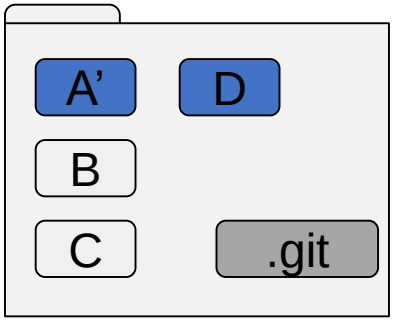


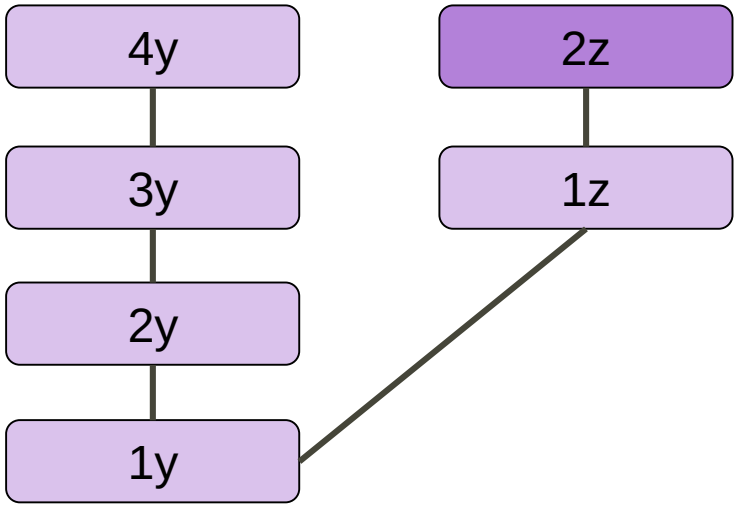
# Basic Git Concept: Branching

- Branches let multiple people work at the same time
  - Alice works on ‘cool-new-feature-branch’
  - Bob works on ‘urgent-bug-fix-branch’
- Commits let you go back to a previous version of the project
  - Alice commits when she gets one section of the feature working, e.g. ‘display email’
  - She tries to then add ‘edit email’, which breaks the tool
  - She reverts to the ‘display email’ commit, and tries again a new way
- Every repository has a ‘master’ branch – this should be clean at all times, i.e. this is the current ready-to-use version of the project.
- Whenever you want to do some development, make a branch first. In general, only one person should work on a branch (make a sub branch if two people need to work)

# But how do you get these changes back into the ‘master’ branch?

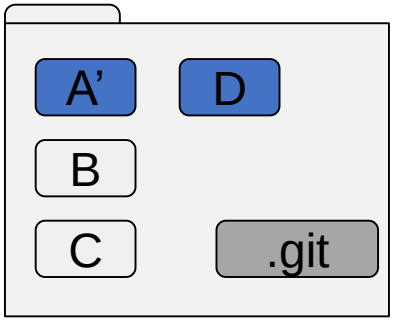
- merging brings the changes from one branch into another branch.
- With code (text files) git is pretty smart – it can merge changes to different sections, or let you manually pick which version to use line-by-line

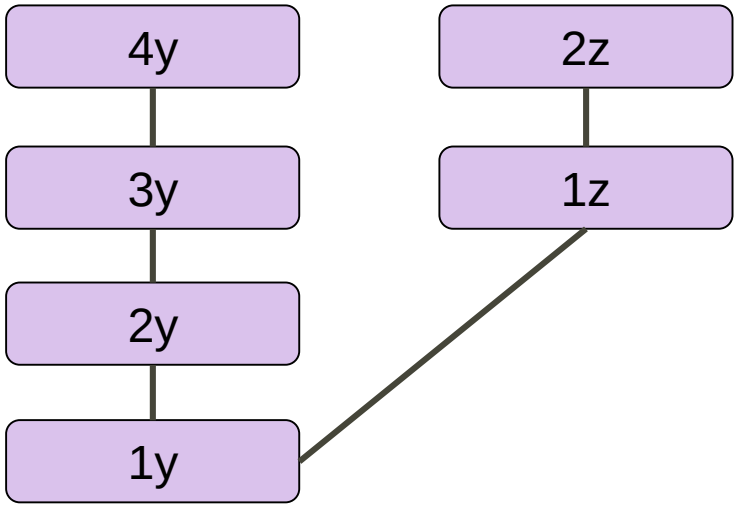


# But how do you get these changes into the ‘master’ branch?

- With binary files (pictures, .xlsx files, etc.) you have to tell git which version to use. If the file changed on both branches, you have to pick one or the other (or manually combine them).
- merging always edits the active branch
- if a file only changed in one branch, the changed version is used by default.

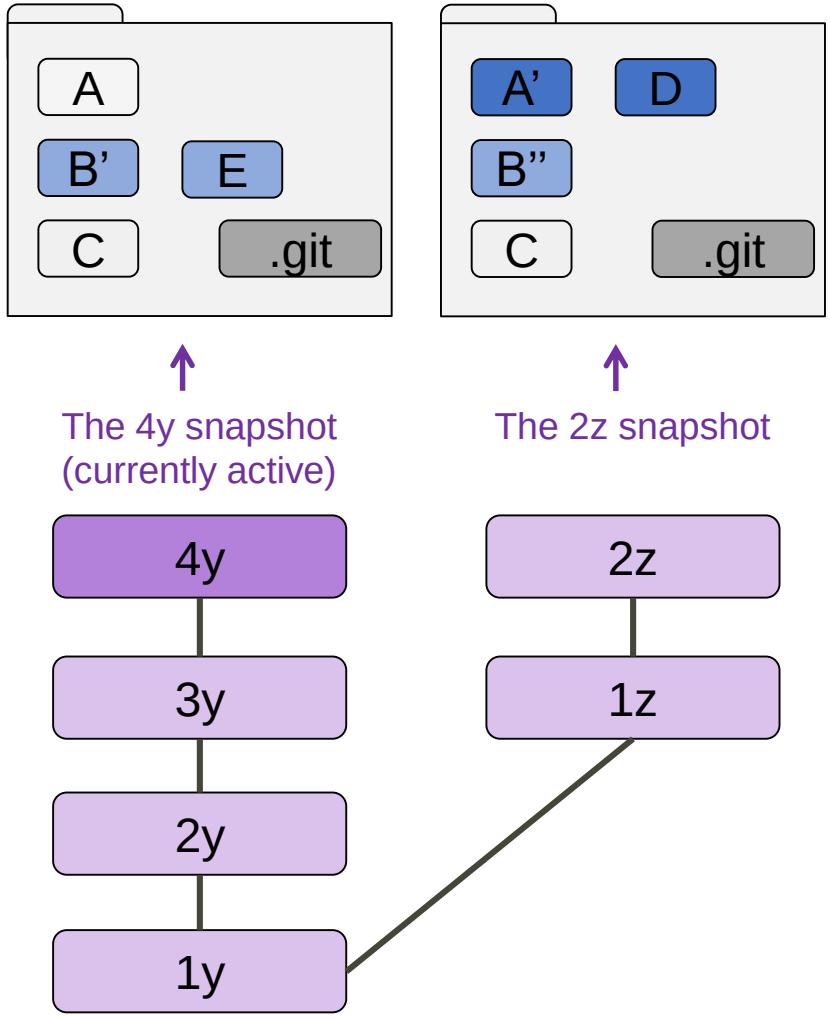


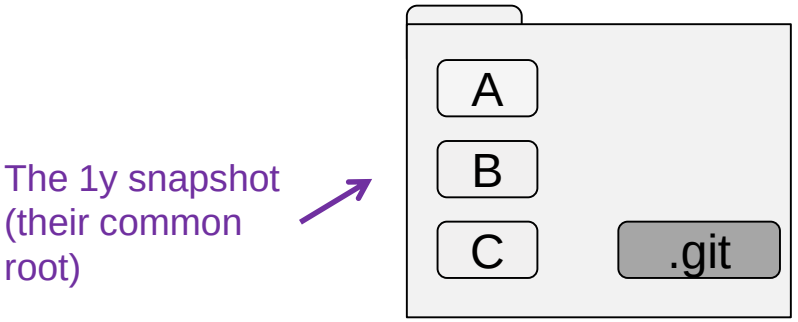



# But how do you get these changes into the ‘master’ branch?

Let’s look at an example. To merge branches y and z, we need to:

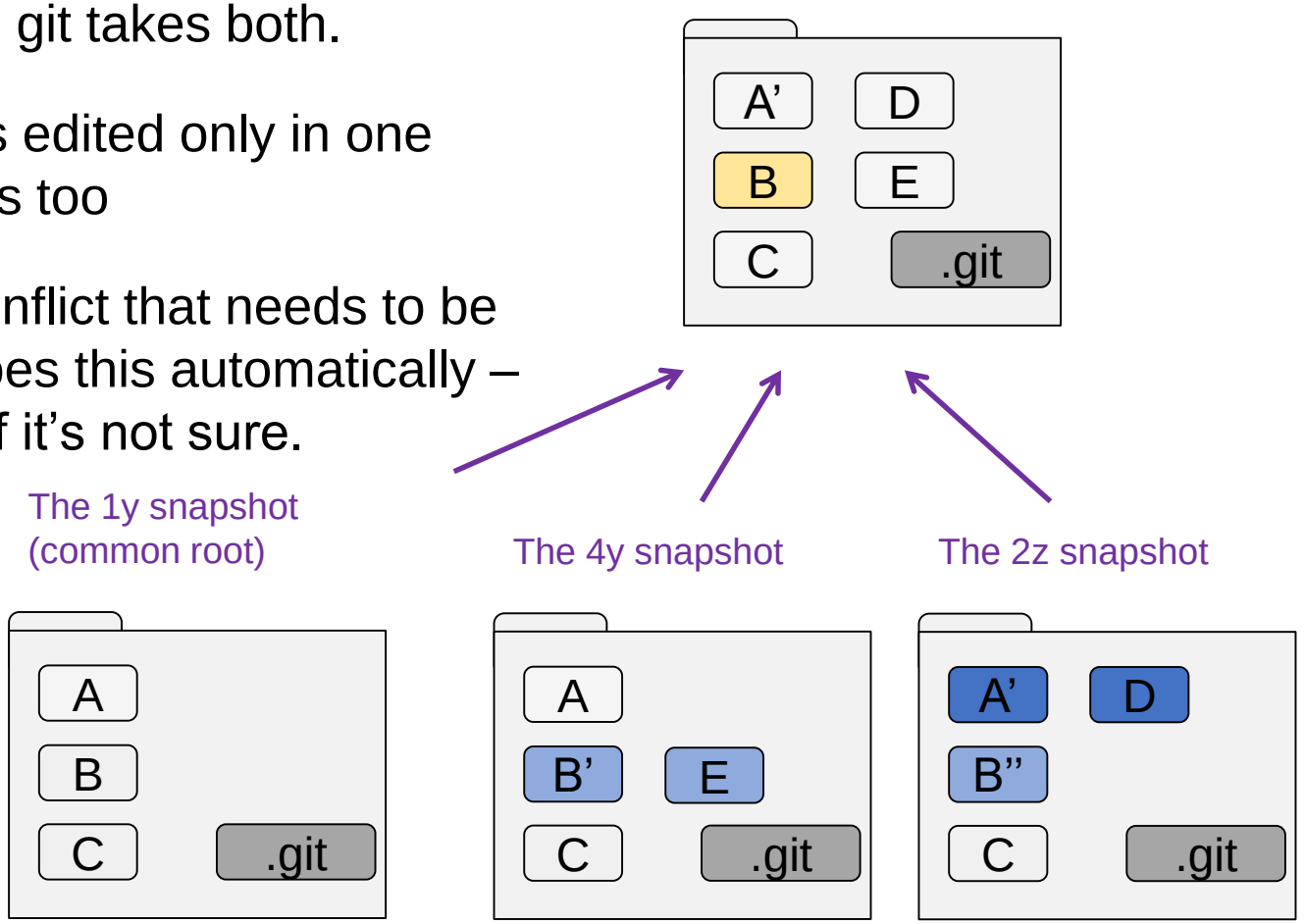
- checkout branch y (the head is commit 4y)
- merge in branch z (i.e. the head commit, 2z)



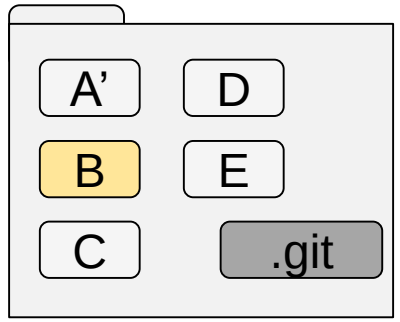
# But how do you get these changes into the ‘master’ branch?

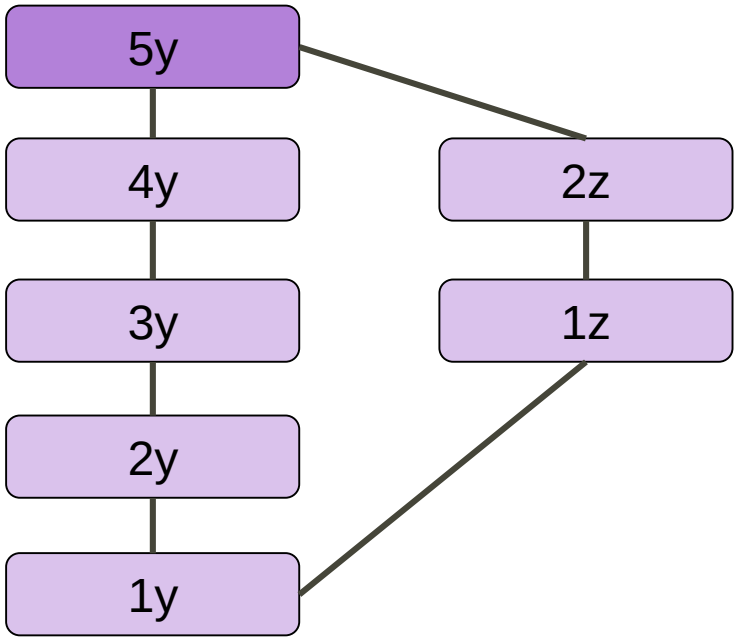
- Since file E was added in y, and file D was added in z, git takes both.
- Since file A was edited only in one branch, it comes too
- B is a merge conflict that needs to be resolved. Git does this automatically – and alerts you if it’s not sure.



# But how do you get these changes into the ‘master’ branch?

- Because branch y was checked out when we merged, the changes occur on branch y
- branch y now has a new commit, and the HEAD of y combines both versions
- branch z is unchanged



# Basic Git Concept: Merging

- Merges bring the changes from one branch into another easily
- If changes only occurred to a file in one of the branches, that version is used.
- If both branches changed a file, you should inspect the differences, and if necessary manually add the changes from one branch to the file on the other, create a new commit, and then merge.
- Before starting the merge, you should check out the branch you want to add the changes to. Merging brings those changes into the checked out branch (and does not affect the other branch)

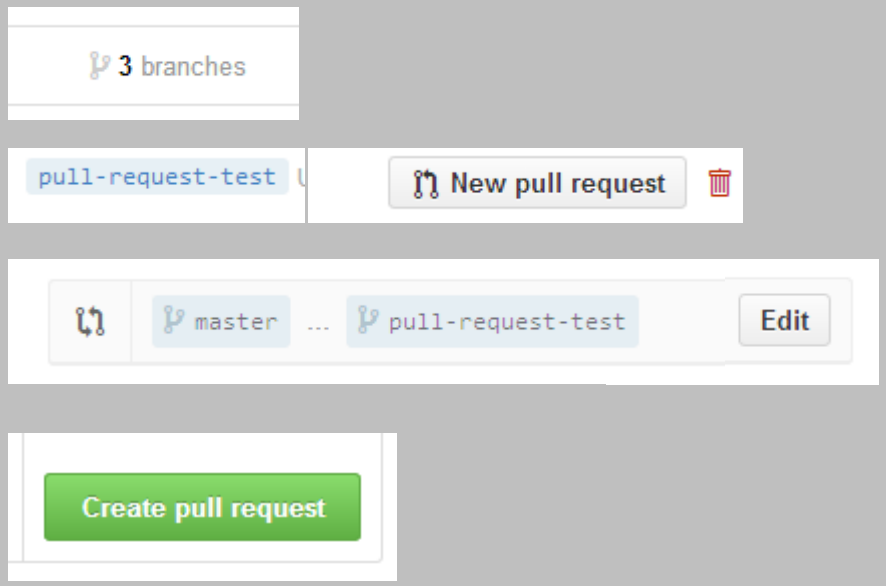
# What about a QA process??

Merging is all well and good, but you need some quality assurance before you are able to make something the new ‘official’ version.

- As a general rule, only merge into a branch that you created
- If you want something to be brought into someone else’s branch (or the master branch), you should use a pull request on Github.
- If the term ‘pull request’ is confusing, think of it instead as a ‘merge request’
  - Remember, pull = fetch + merge, so it’s a declaration to the other users to both get your changes and merge them in to their copy of the repository.

# What about a QA process??

- How to do it:




1) Go to the branch you have edited

2) New pull request

3) Make sure the target branch (i.e. master in this example) is correct, or click edit

4) Create pull request