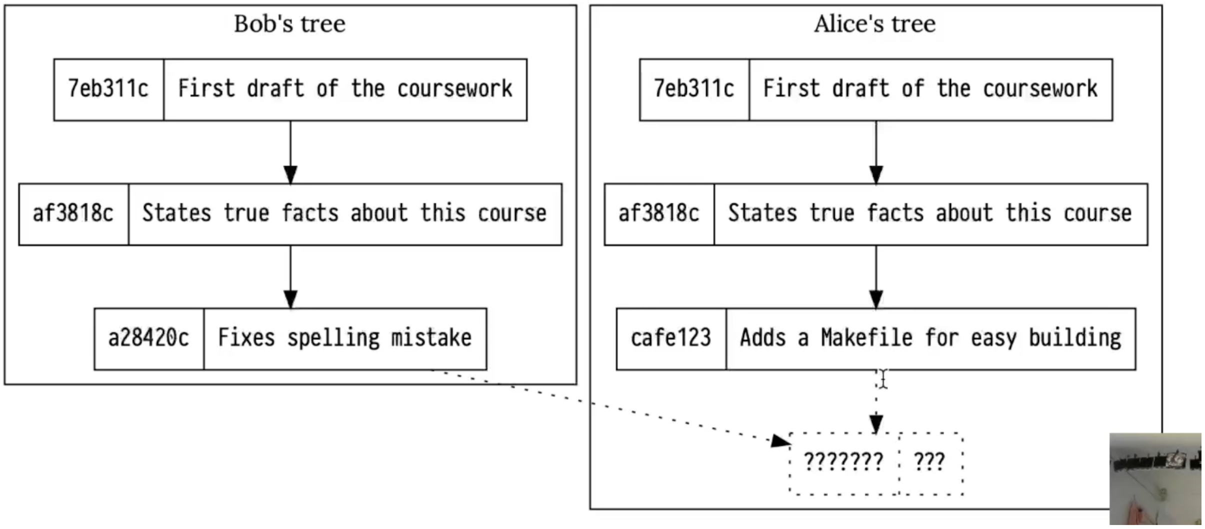
Git: Merging & Rebasing

What happens if things go wrong?

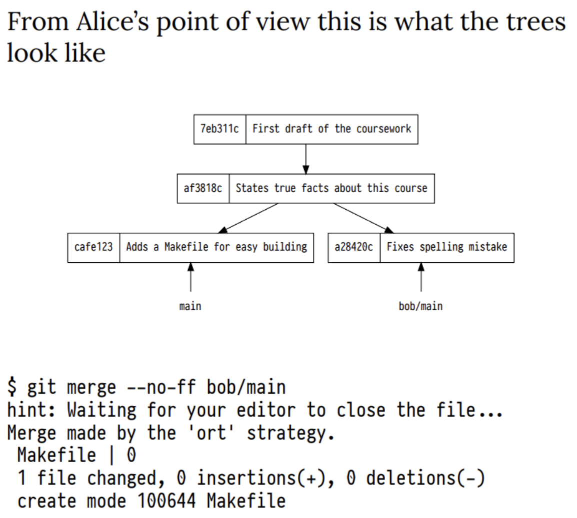
- Alice pulled Bob's changes earlier she could simply pull them across to her tree if she hasnt made any changes
- But say Alice had been making commits on her own tree whislt bob was alos making changes?
    - now the two trees have *diverged*
    - therefore they cant be **fast forwarded**

Fastforwarding - when the feature branch builds directly off the main branch, so to merge them you can just move the main branch pointer to the top of the new feature branch


Bob's tree
7eb311c
First draft of the coursework
af3818c
States true facts about this course
a28420c
Fixes spelling mistake
Alice's tree
7eb311c
First draft of the coursework
af3818c
States true facts about this course
cafe123
Adds a Makefile for easy building
???????
???

- e.g. Bob doesnt have the commit for the Makefile as his tree diverged earlier
    - one tree has the Makefile, the other has "fixes spelling mistake"

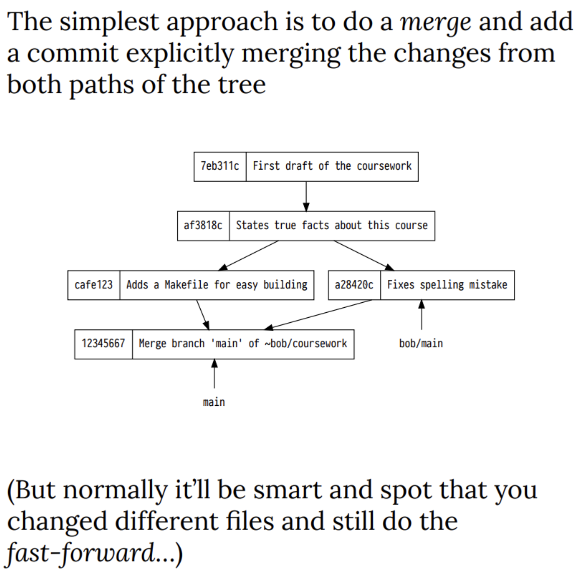
Merging



From Alice's point of view this is what the trees look like

7eb311c
First draft of the coursework
af3818c
States true facts about this course
cafe123
Adds a Makefile for easy building
a28420c
Fixes spelling mistake
main
bob/main

```
$ git merge --no-ff bob/main
hint: Waiting for your editor to close the file...
Merge made by the 'ort' strategy.
 Makefile | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Makefile
```



The simplest approach is to do a *merge* and add a commit explicitly merging the changes from both paths of the tree

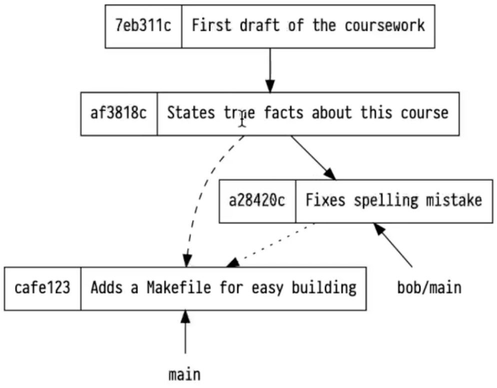
7eb311c
First draft of the coursework
af3818c
States true facts about this course
cafe123
Adds a Makefile for easy building
a28420c
Fixes spelling mistake
12345667
Merge branch 'main' of ~bob/coursework
bob/main
main

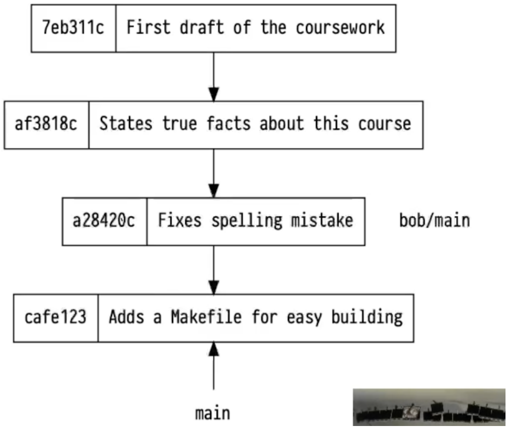
(But normally it'll be smart and spot that you changed different files and still do the *fast-forward*...)

- this merge will be fine as long as the two files dont interact and break each other

Rebase

The alternative is to say that as long as Alice's changes do not affect Bob's changes at all, then we do some time travel and rewrite the order of commits - i.e. that Alice merged bob's changes before she began her work...


7eb311c
First draft of the coursework
af3818c
States true facts about this course
a28420c
Fixes spelling mistake
bob/main
cafe123
Adds a Makefile for easy building
main


7eb311c
First draft of the coursework
af3818c
States true facts about this course
a28420c
Fixes spelling mistake
bob/main
cafe123
Adds a Makefile for easy building
main

- you want the left diagram to become the right one

use the git rebase command

```
git rebase bob/main
```

- so if Alice runs that command and rebases onto bob main, it will take all of her work up until the point where their trees diverged, and move it onto the **HEAD** of bobs main
- git rebase has the potential to go wrong

Merging vs Rebasing

- Merging is simpler conceptually
    - however it is slightly messier due to more merge commits
- Rebasing is neater but more failure prone and complicated