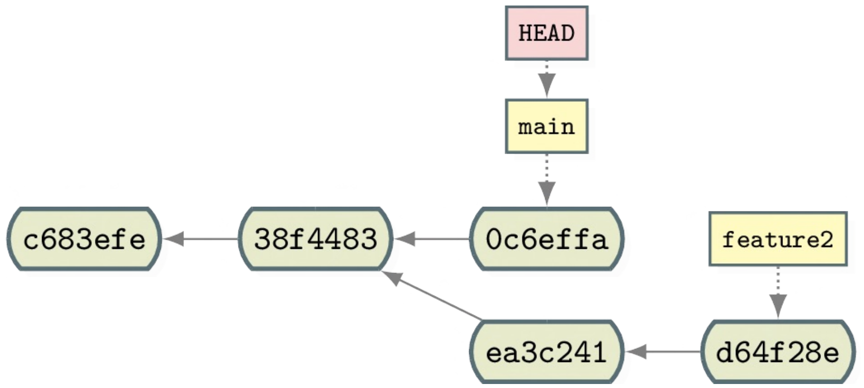
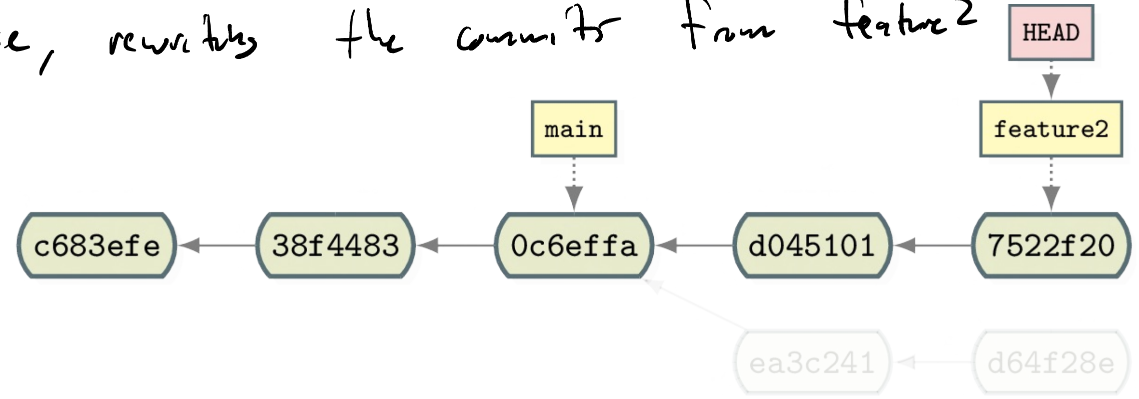
Git Rebase

- Like a merge, a rebase can combine commits from multiple branches
- Merging
    - Preserves past commits
    - Fast-forwards if the branches being merged do not have divergent histories
    - Creates a merge commit otherwise

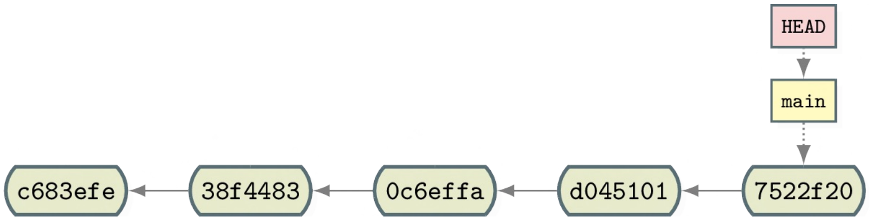
- Rebasing
    - Takes changes committed to one branch and replays them on a different branch
    - Rewrites history
        - Past commits are altered to appear as though they happened in a branch they did not happen in
        - Should be done with <u>extreme caution</u>



Rebase, rewriting the commits from feature2



Merge feature2 into main, delete feature2



- Merging vs rebasing
    - Merging preserves the record of what happened
    - Rebasing allows for construction of a simpler story of what happened
    - Rebasing can screw things up if you rebase commits that other developers have based their work on
    - For a "normal" workflow, merging avoids more issues and is probably preferable

- Reserve use of rebasing for cleaning up history between the last contribution and current work

- Squashing
    - Using rebase to combine multiple commits into one
    - Can be useful to combine multiple local commits into one before merging

- Use -i for interactive mode, use HEAD~n to work with the last n commits

```
git rebase -i HEAD~3
```

interactively work with the last 3 commits