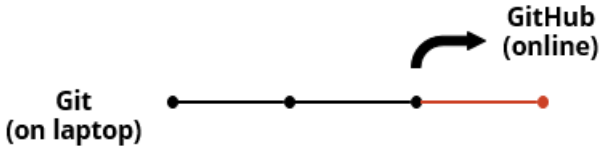
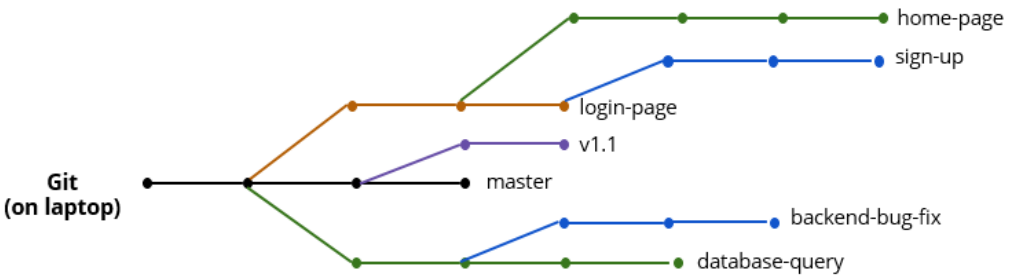
Lecture 1: Git

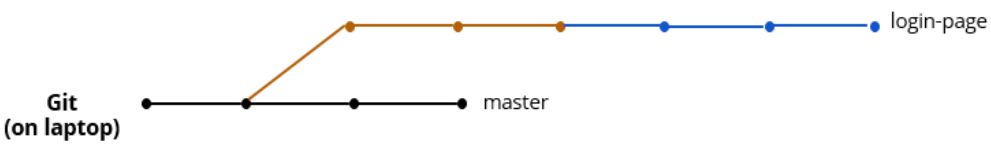
- GitHub [browser]:
  - A website
  - A visual interface to interact with the code that you write and the collaborators of the repositories that you contribute to, also allows you to backup files online
- Git [terminal]:
  - A version control system
  - The powerhouse behind Github
  - Something that happens locally, on your laptop, in the terminal
  - Manage the history of your git repositories.
- For each codebase (repository) I own, I want to write code where:
  1. **Progress Loss is minimized**
     - "Saving your work" and "backing up your work".
     - Create commits define specific checkpoints for saving your work
     - Upload those commits to Github so you have a local structure of a bunch of commits or checkpoints
     - If your laptop crashes, you still have the history for commits uploaded on Github



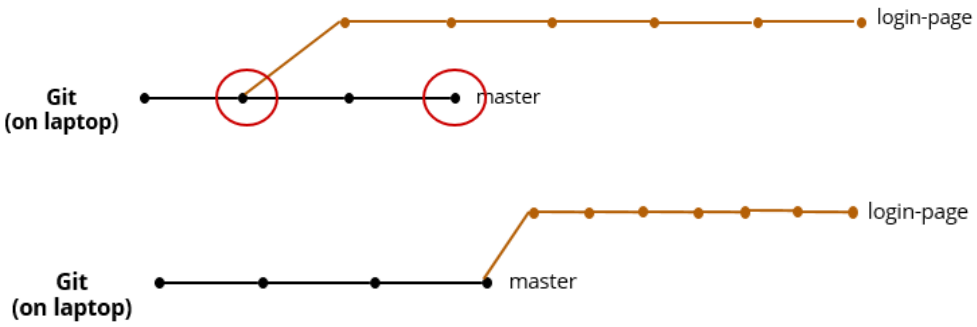
     -
  2. **Iterating on different versions of the code is easy**
     - The ease or difficulty of adding a new feature to the code base may depend on the state / version of the codebase.
     - We need to:
       1. Preserve both versions of history
       2. Overwrite history if we choose
     - So we branch off of that particular commit, push commits per branch, keep the primary branch as "master" branch and name other branches as the that is being developed on or the major or minor version of the software / product



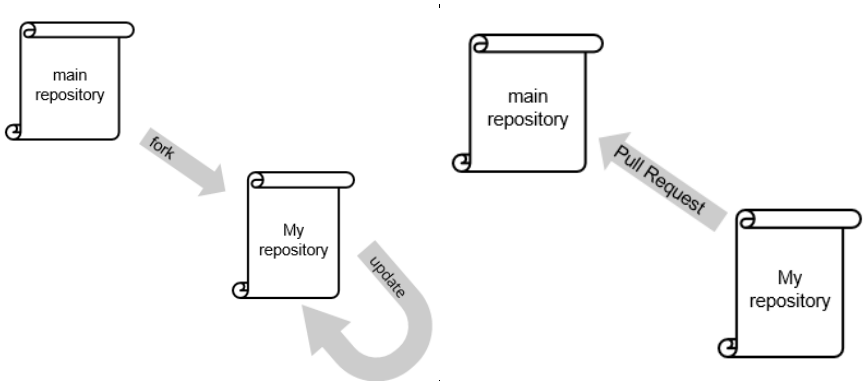
     - If the base of one branch is the head of the other, append the changes for merging



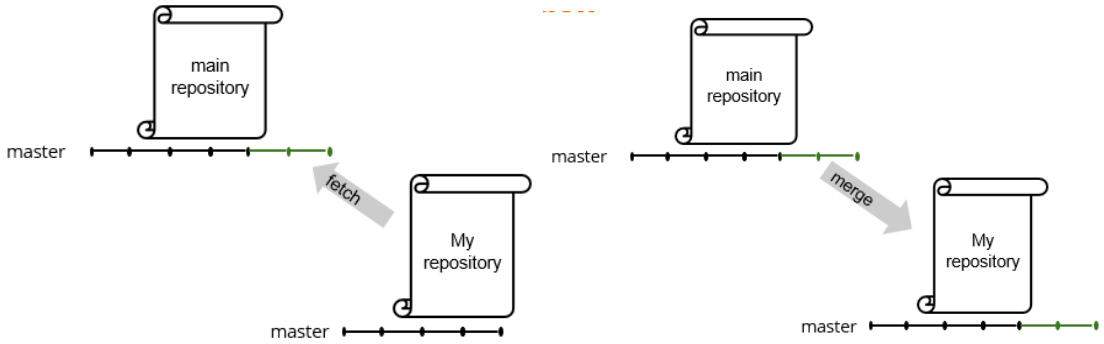
     - Else, rebase to be at the head of the master branch (often require manual intervention to resolve the conflicts)



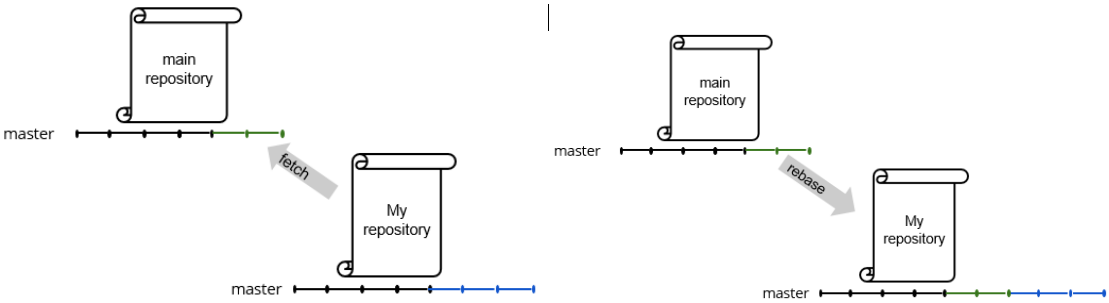
3. **Collaboration is productive**
   - Possible to collaborate on a single repository by having each collaborator create new branches for development and merging their branch into master, which is done by Pull Request (PR)
   - If repository owners doesn't want to manage collaborator permissions:
     1. Make a copy of (fork) the main repository
     2. Make all the changes they want to this copy
     3. Request that part of their copy be merged into the main repository via a Pull Request (PR)



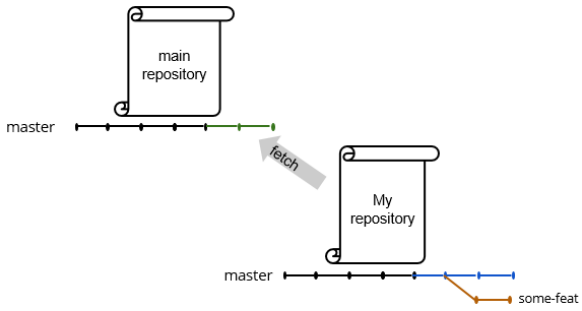
   - While making changes on your copy of the repository, the main repository might change
   - But we only need to keep the master branch up to date

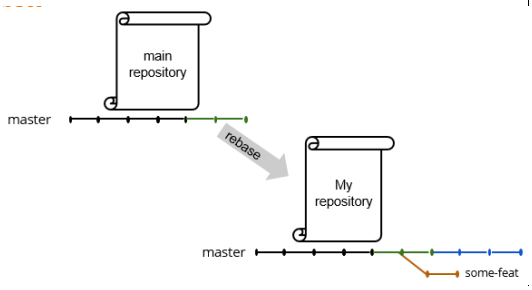


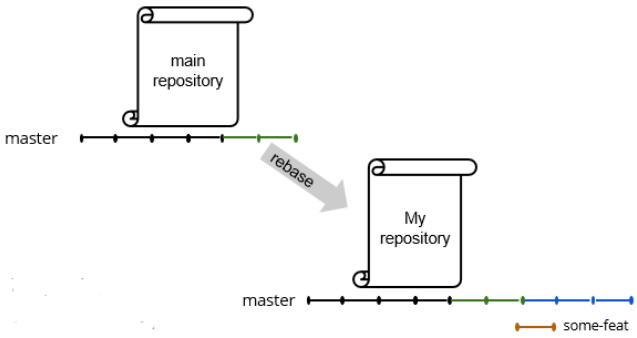
   - Trivial when the base of one branch matches the head of the other, just extend



- For other branches, Git doesn't know how to handle when commits are ordered differently, so some-feat branch is now stranded







- Don't commit anything to the master branch directly: create a new branch for updates, make your changes in separate branch so you can come back to master unless they completely changed which should never happen because your branches should always be in sync or at least able to merge