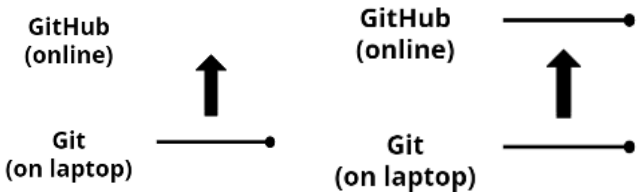
Lecture 0: Git

1) Github vs Git

a) Github(browser): website to backup files online

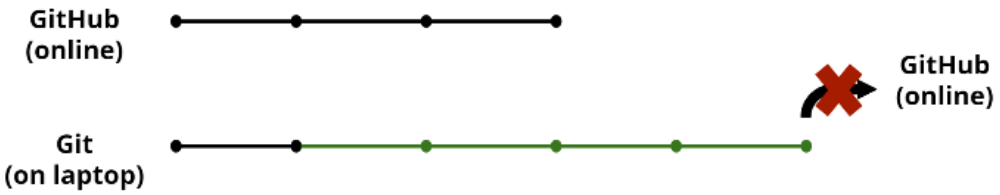
b) Git(terminal): version control system

- Backup work: Creating regular commits(save points) and pushing them to Github
  - Laptop destroyed- > only lose things not committed to Github

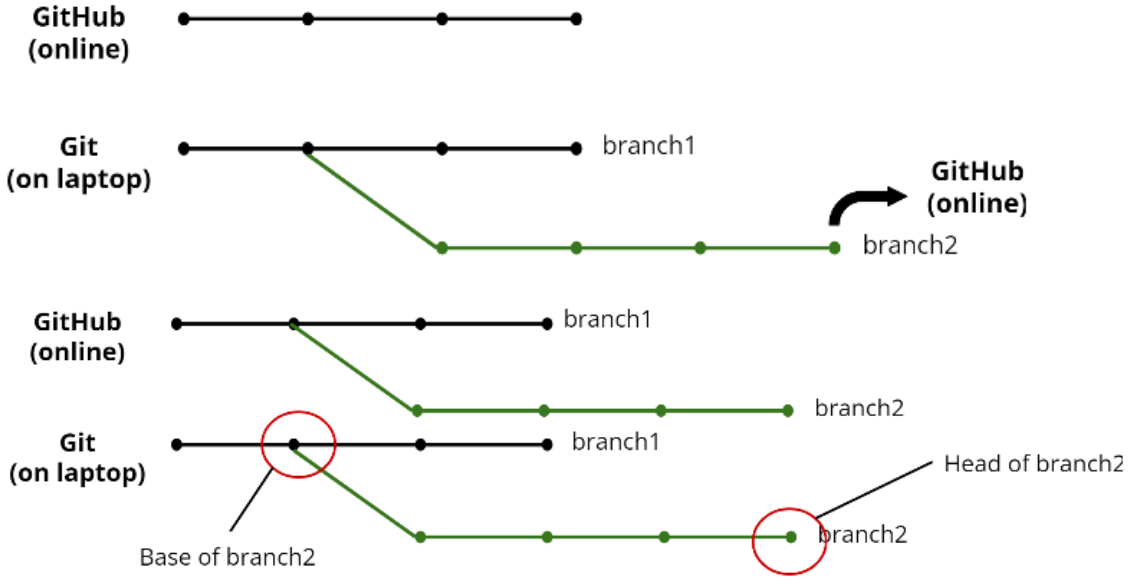


2) Github vs Git

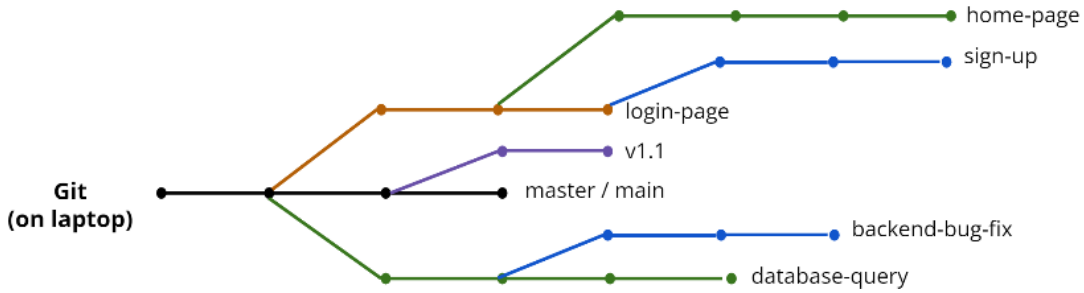
a) Want to preserve both versions of history and overwrite if we wanted to



i) Branch off from particular commit and push commits per branch

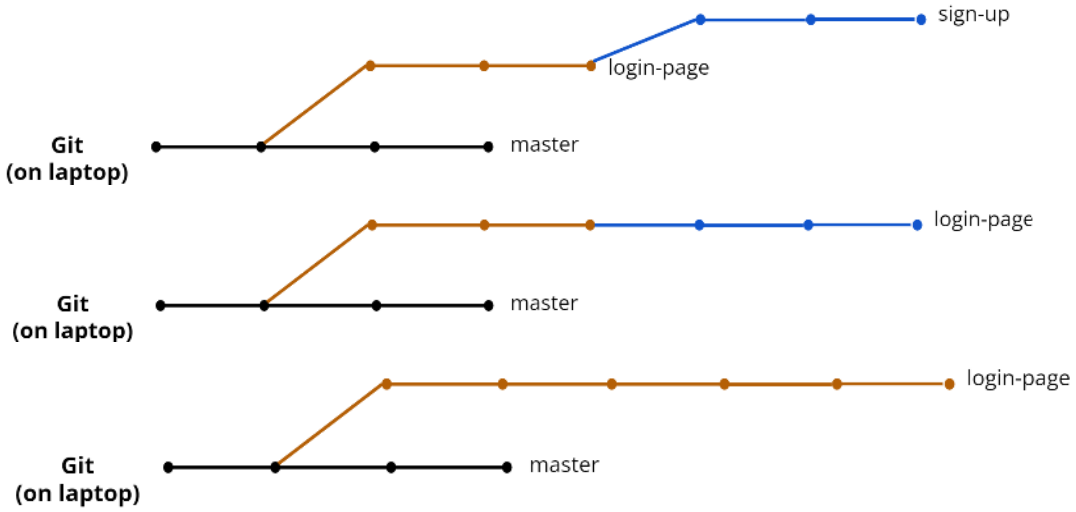


ii) Can have multiple branches, but one of them has to be the primary one: "master" or "main" branch and can name others branches anything

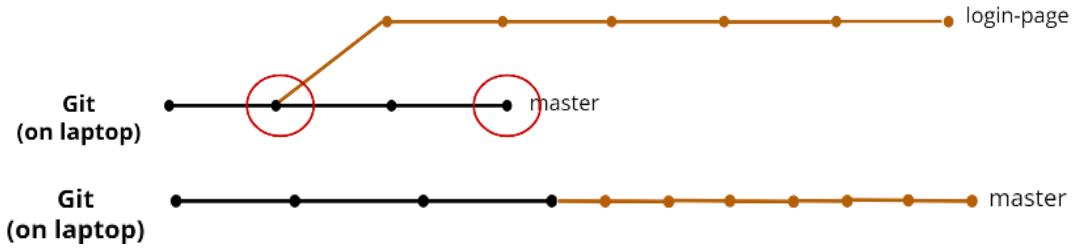


b) Clean up branches by merging them with master/main branch or each other

i) If the base of one branch is the head of another, simply append



ii) If not the head of another branch, commits can conflict with each other. Need to change base of login-page branch(rebase) to head of master branch.



(1) Often manual intervention is needed to resolve conflicts

3) Collaboration

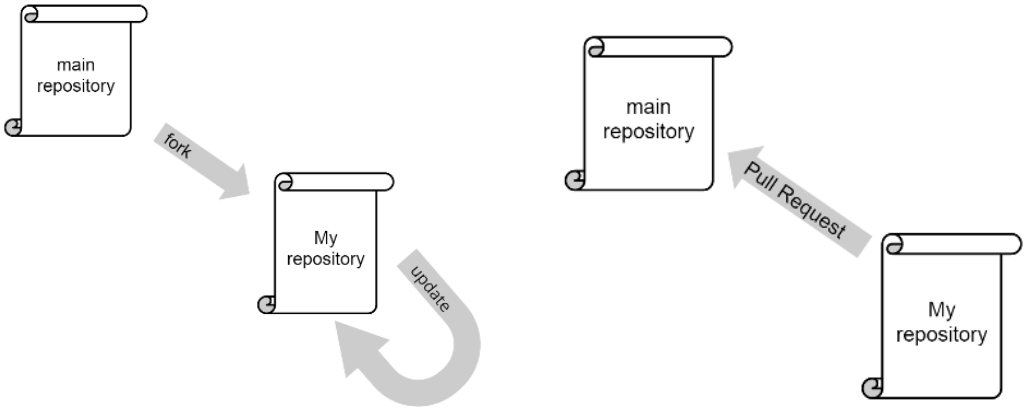
a) Steps:

i) Make a copy of main repository -> fork

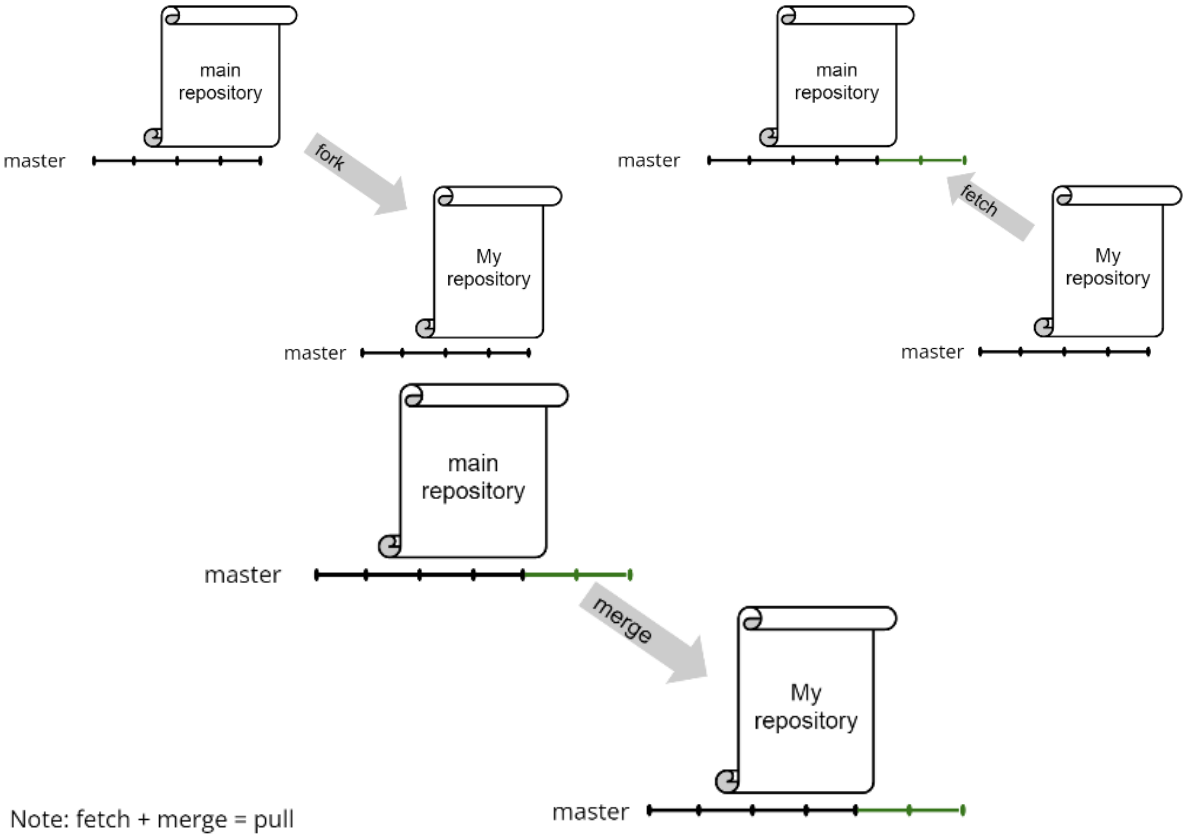
ii) Make all changes wanted on copy

iii) Request part of copy to be merged into main repository via Pull Request(PR)

b) Base of one branch match head of another branch

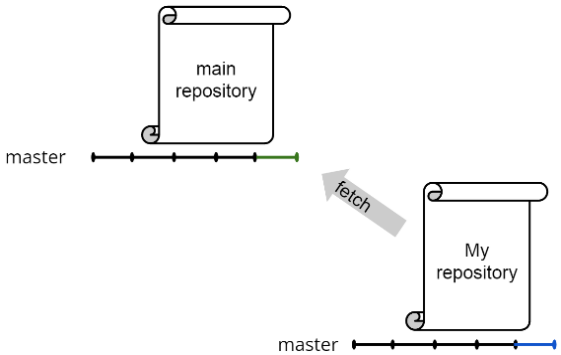


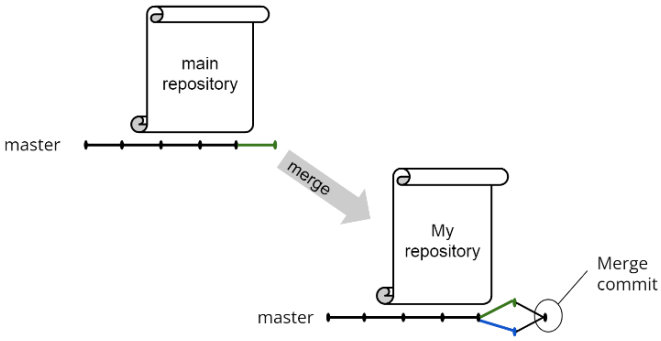
c) Keeping copy up to date



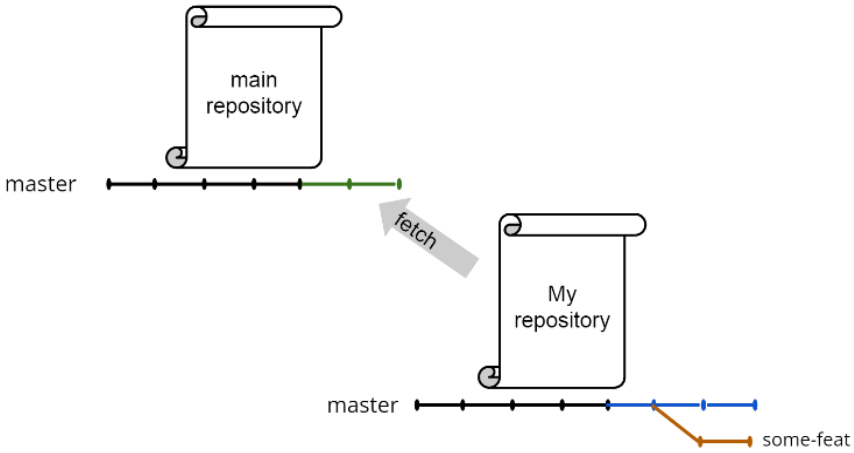
Note: fetch + merge = pull

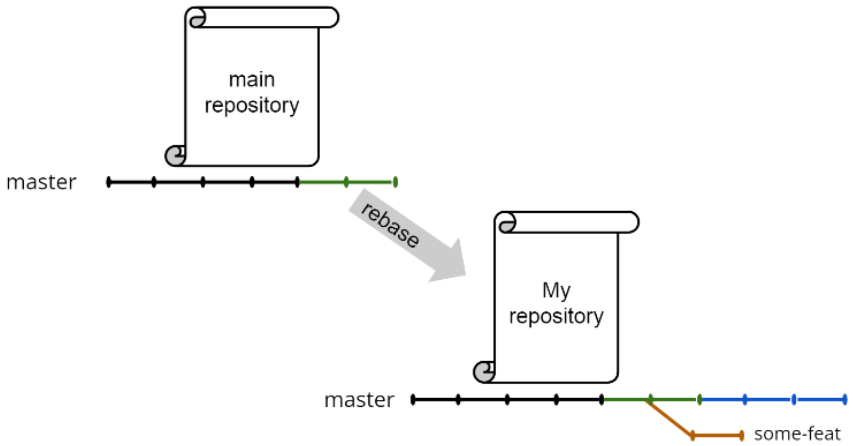
i) Trivial when base of one branch match head of another branch

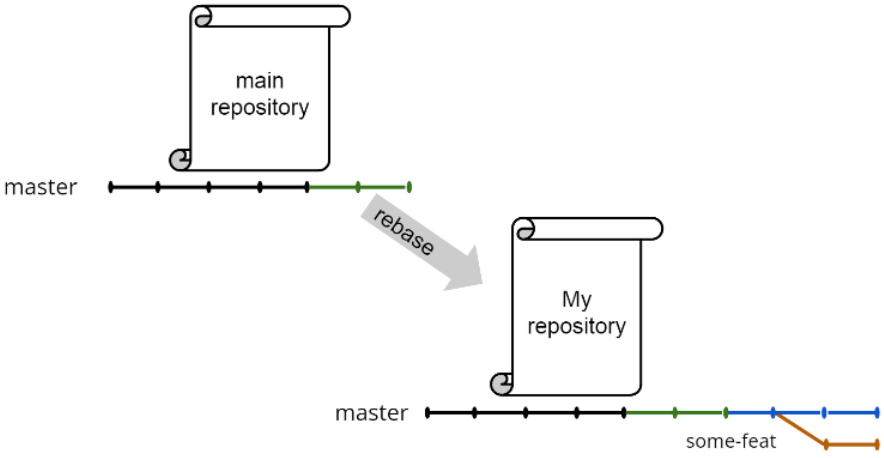


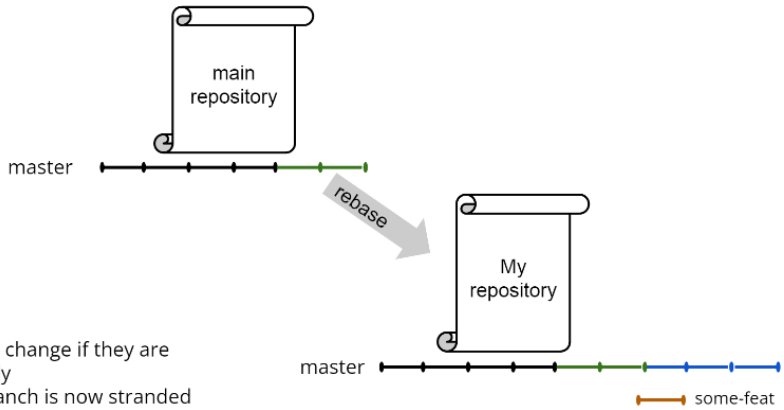


d) Branches when rebasing









commit IDs (shas) change if they are ordered differently
The some-feat branch is now stranded

Rule: Always create new branch when developing and never commit directly to master branch