Open Source Version 관리 Git2

구명완교수

Office: K관 239호

Email: mwkoo9@gmail.com

Local Repository A -> GitHub Repository

```
$ mkdir  local
$ echo “my test” > test.txt
$ git add test.txt
$ git commit –m “Uploading”      (1)
$ GitHub Repository 를 만든다.
     - 초기에 readme.md 를 안 만든다.
$ git remote add githubSite
https://githubcom/mwkoo/KOO.git   (2)
$ git remote -v
$ git push –u githubSite master        (3)
$ git status                              (4)
```

(1) test.txt 를 local Repository 에 저장한다.

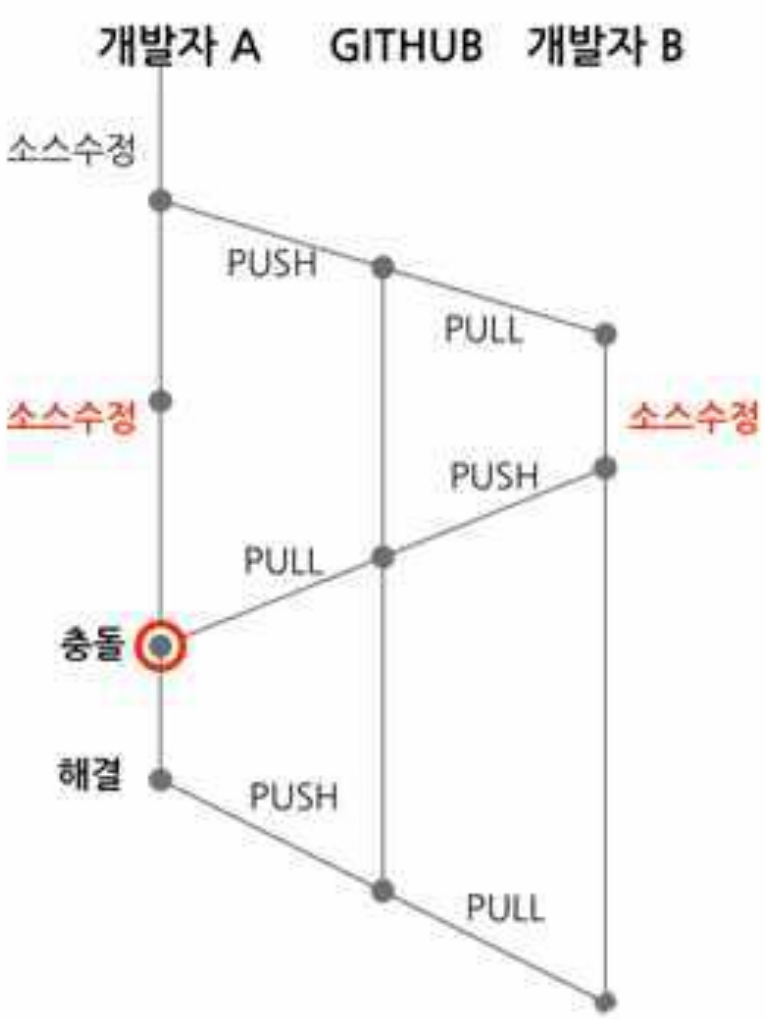
(2) 서버에 저장소를 githubSite local 이름으로 만든다.

(3) 서버에 파일 test.txt를 Uploading 한다.

Changes in GitHub Repository

- Make Changes in GitHub Repository (Local B)
  - Changes in test.txt on GitHub Reposity
    - printf “ kk”
- Local A
  - git pull githubSite master
  - git remote –v
  - git status

Conflict between Local and GitHub Repository

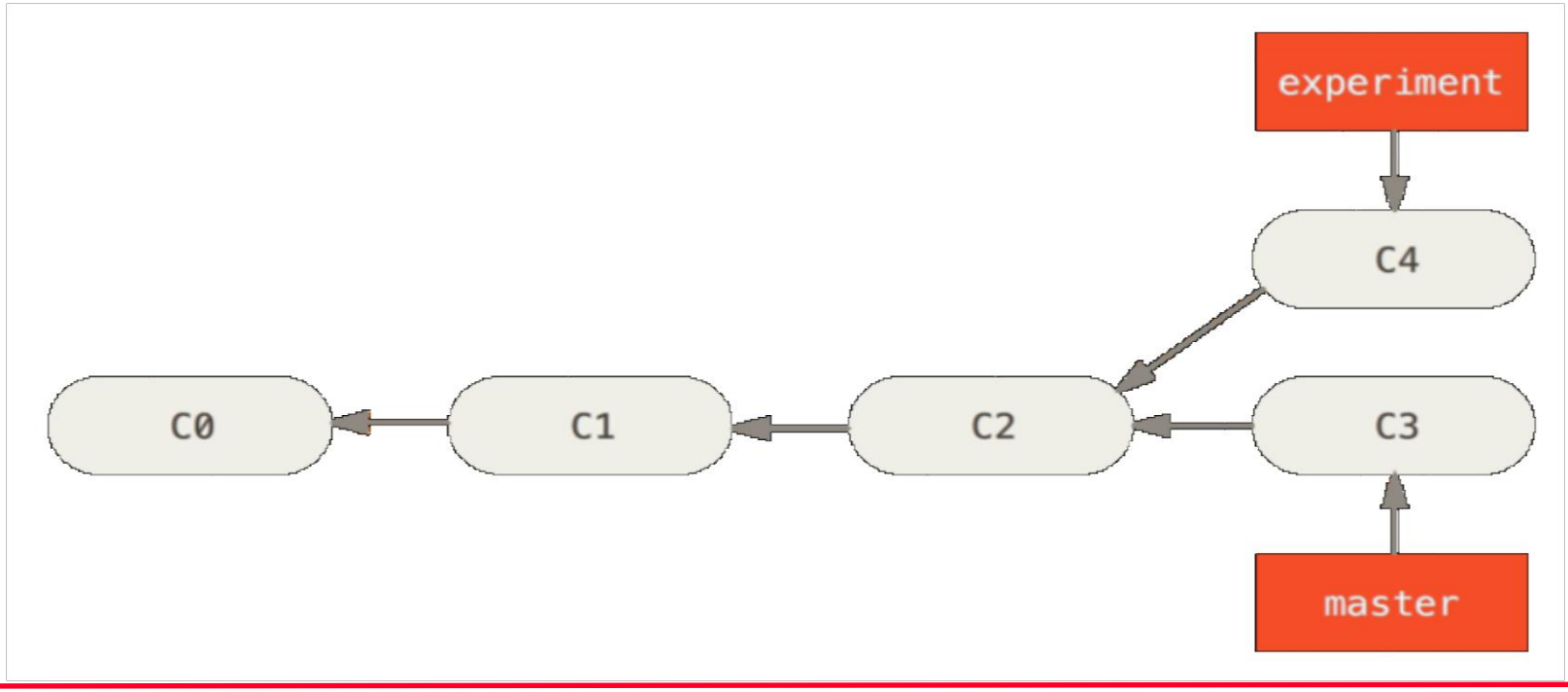


Conflict between Local and GitHub Repository

- Make Changes in GitHub Repository
  - Changes in any file on GitHub Reposity
    - printf " kk" in test.txt
- Local
  - vi test.txt
    - printf " KOO " in test.txt
  - git commit –am " koo changed "
  - git pull githubSite master
    - Conflict error → vi test.txt 변경
    - <<<< , ======, >>>> 제거
  - git commit –am " Conflict resolved "
  - git push --set-upstream githubSite master

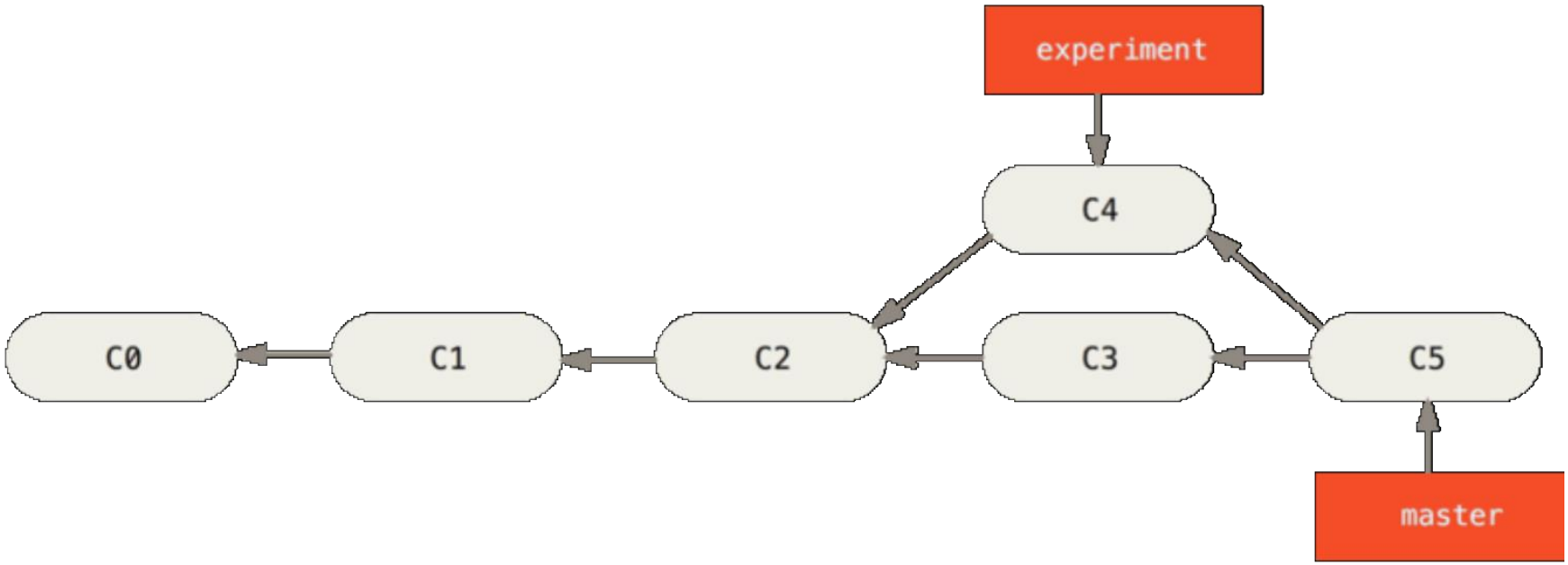
Rebasing

- two main ways to integrate changes from one branch into another
  - Merge
  - Rebase
- Merge



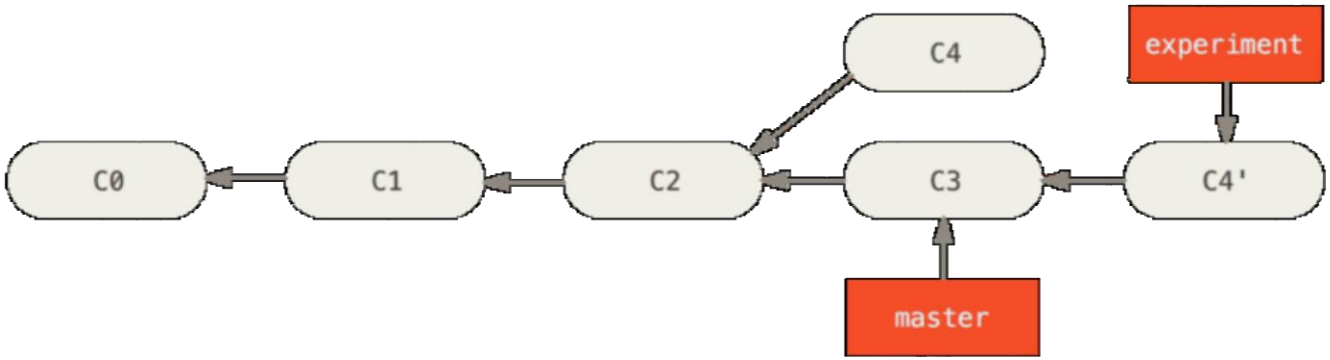
Rebasing

- $ git checkout experiment
- $ git merge master

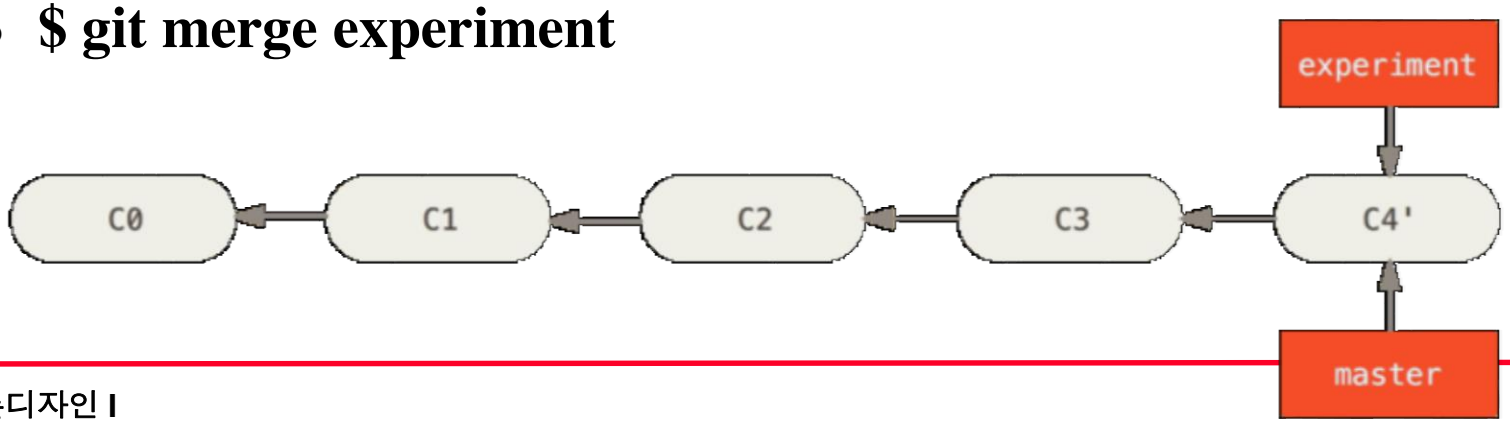


Rebasing

- Rebase
  - $ git checkout experiment
  - $ git rebase master

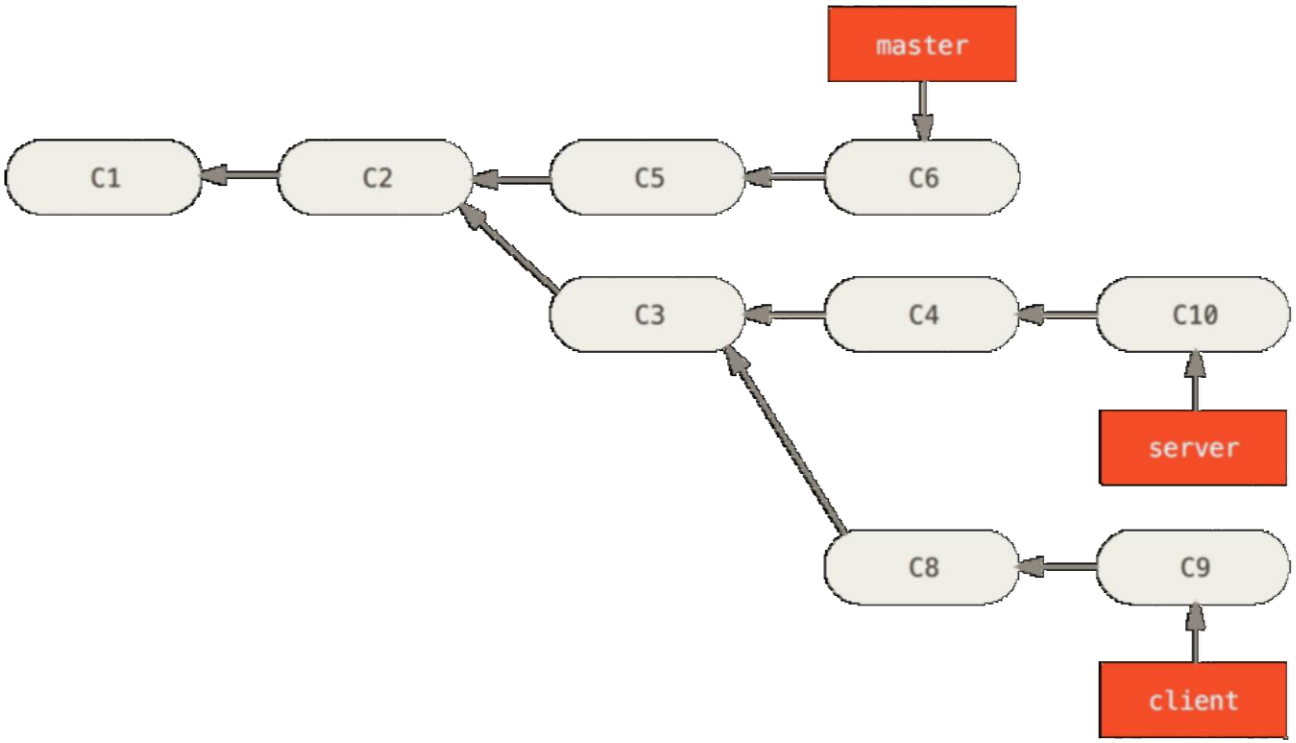


  - $ git checkout master
  - $ git merge experiment



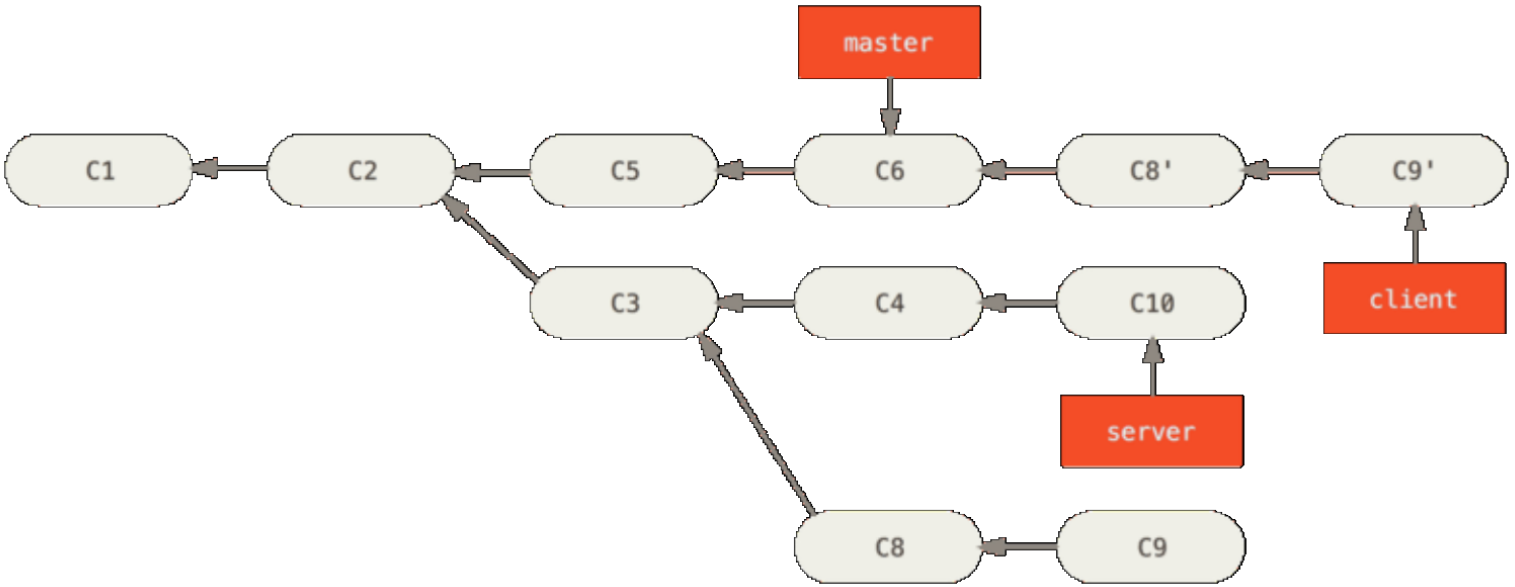
More Interesting Rebasing

- Goal
  - to merge your client-side changes into your mainline for a release
  - to hold off on the server-side changes until it's tested further



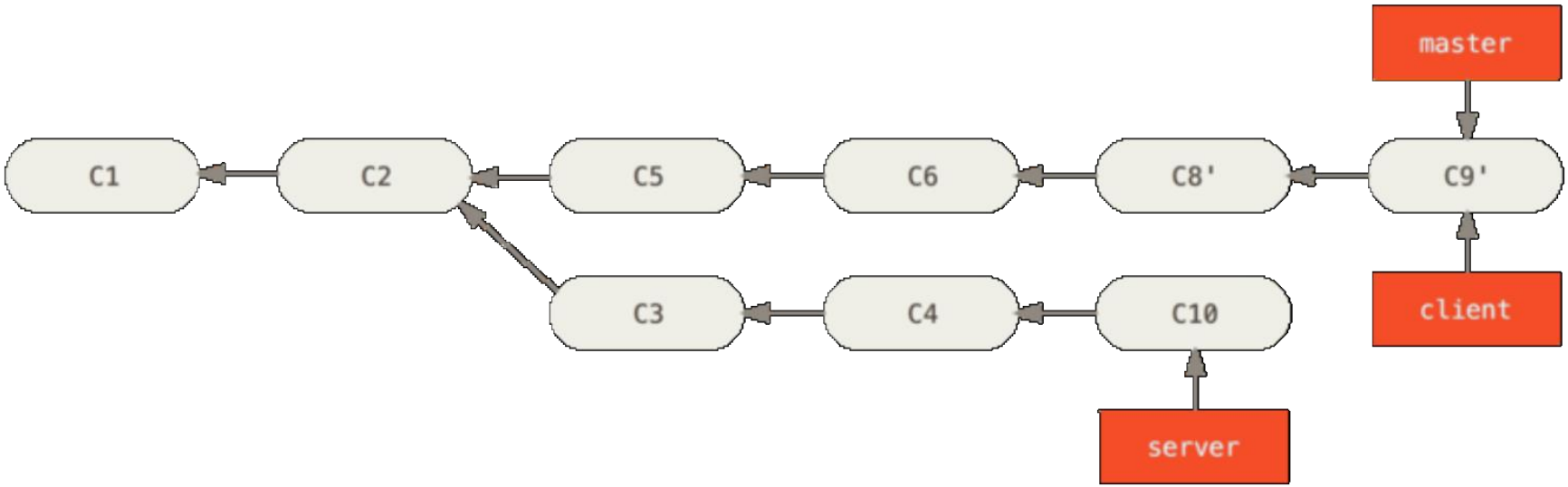
More Interesting Rebasing

- $ git rebase --onto master server client
  - Check out the client branch,
  - figure out the patches from the common ancestor of the client and server branches,
  - and then replay them onto master



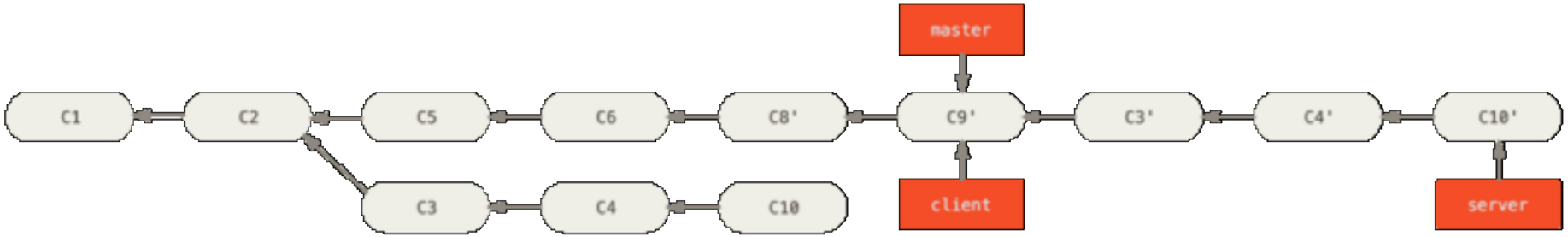
More Interesting Rebasing

- $ git checkout master
- $ git merge client
  - fast-forward your master branch



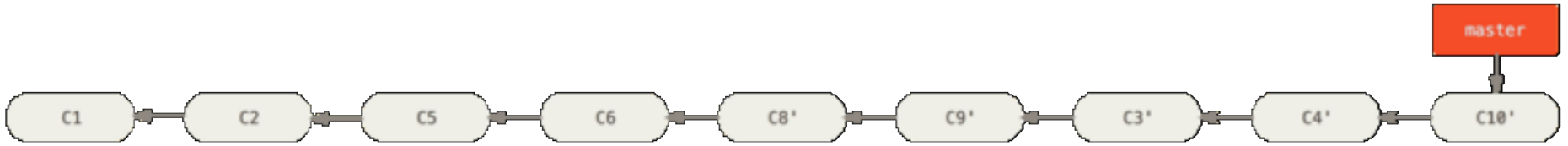
More Interesting Rebasing

- git rebase [basebranch] [topicbranch]
- $ git rebase master server
  - checks out the topic branch (in this case, server)
  - replays it onto the base branch (master):



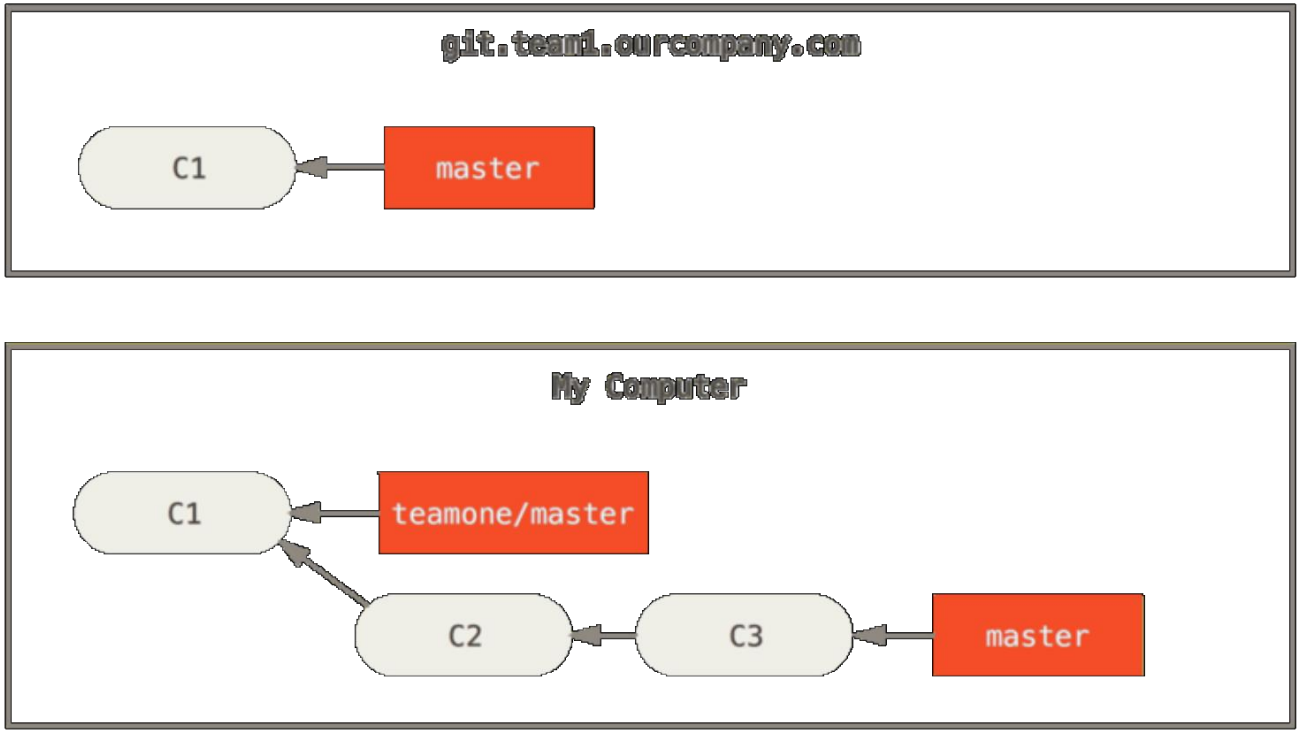
More Interesting Rebasing

- $ git checkout master
- $ git merge server
  - fast-forward the base branch (master):
- $ git branch -d client
- $ git branch -d server



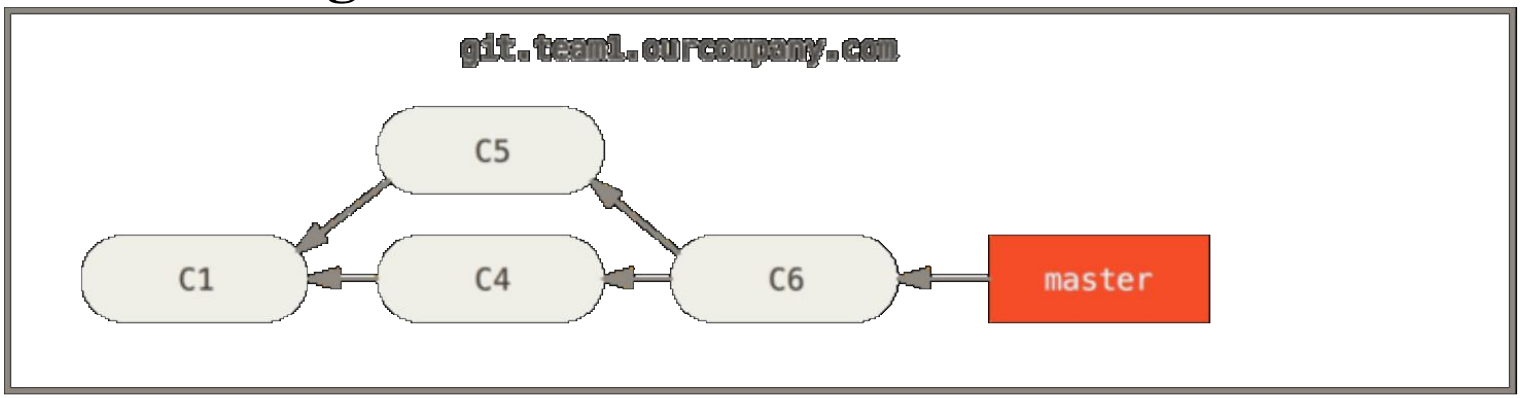
The Perils of Rebasing

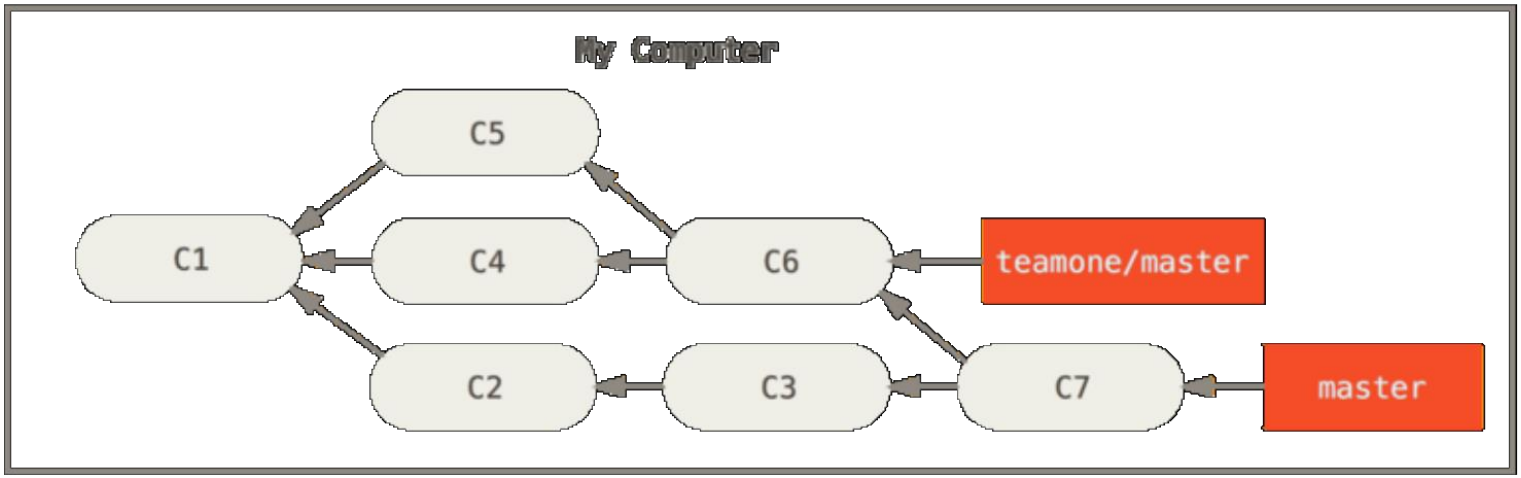
- Do not rebase commits that exist outside your repository
- Suppose you clone from a central server
  - and then do some work off that



The Perils of Rebasing

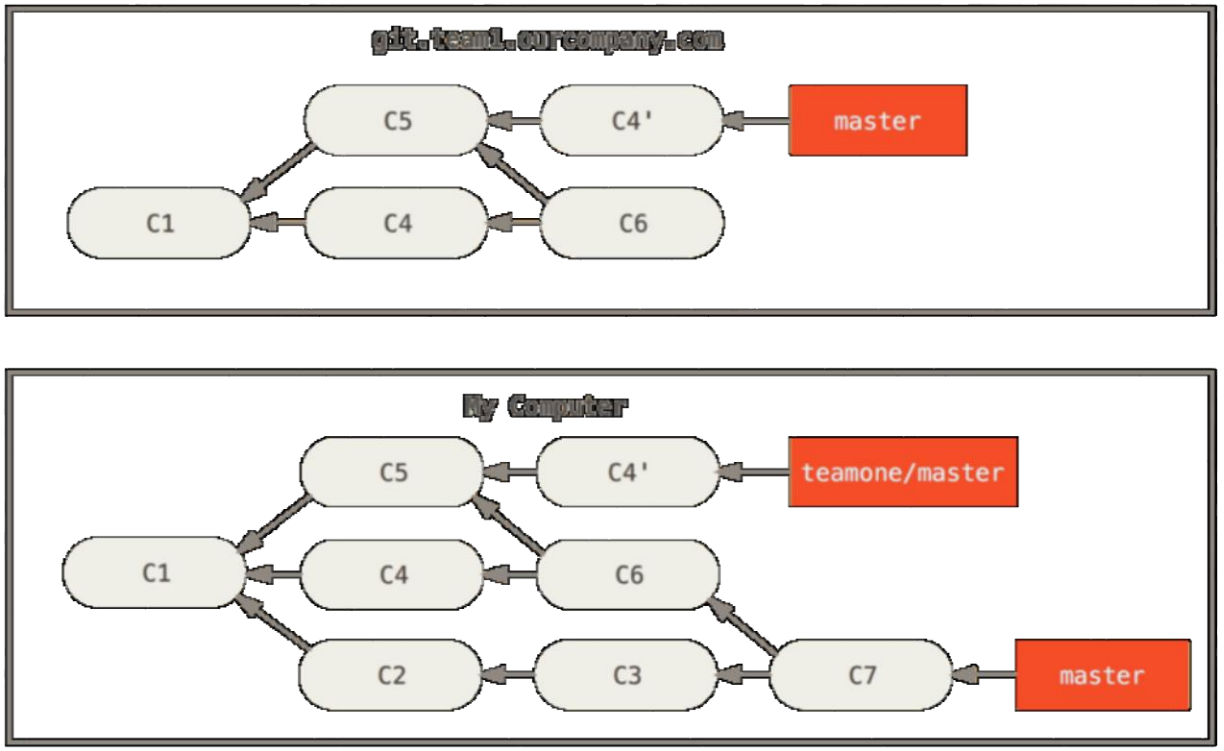
- a merge, and pushes that work to the central server.
- fetch it and merge the new remote branch





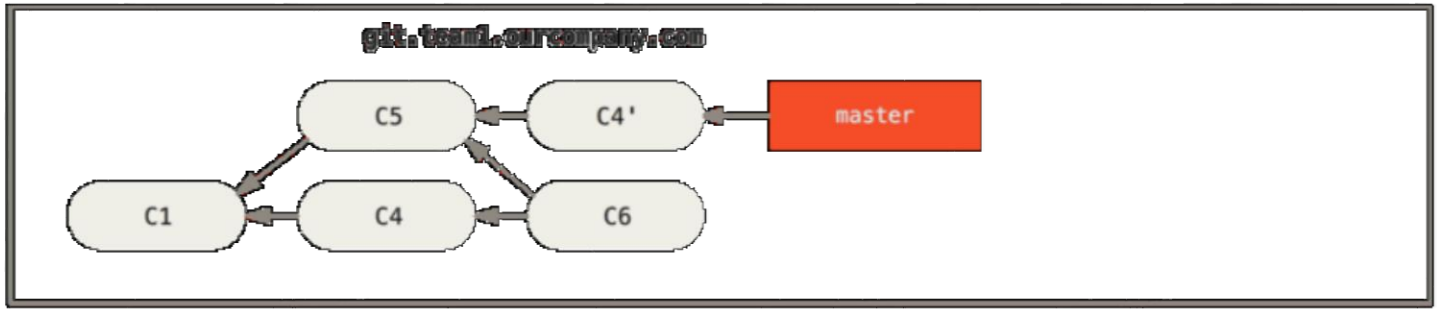
The Perils of Rebasing

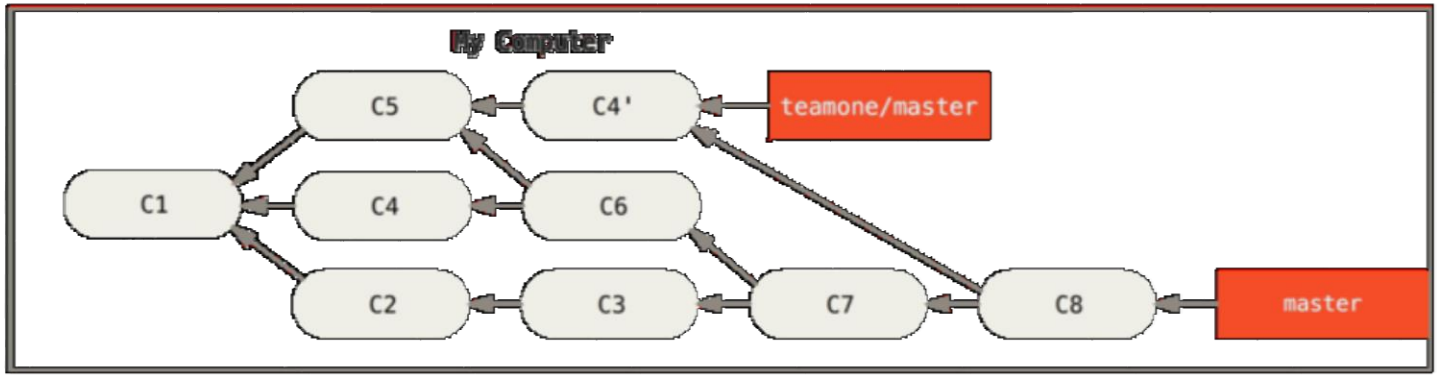
- decides to go back and rebase their work instead
  - a git push –force to overwrite the history on the server
- You then fetch from that server, bringing down the new commits



The Perils of Rebasing

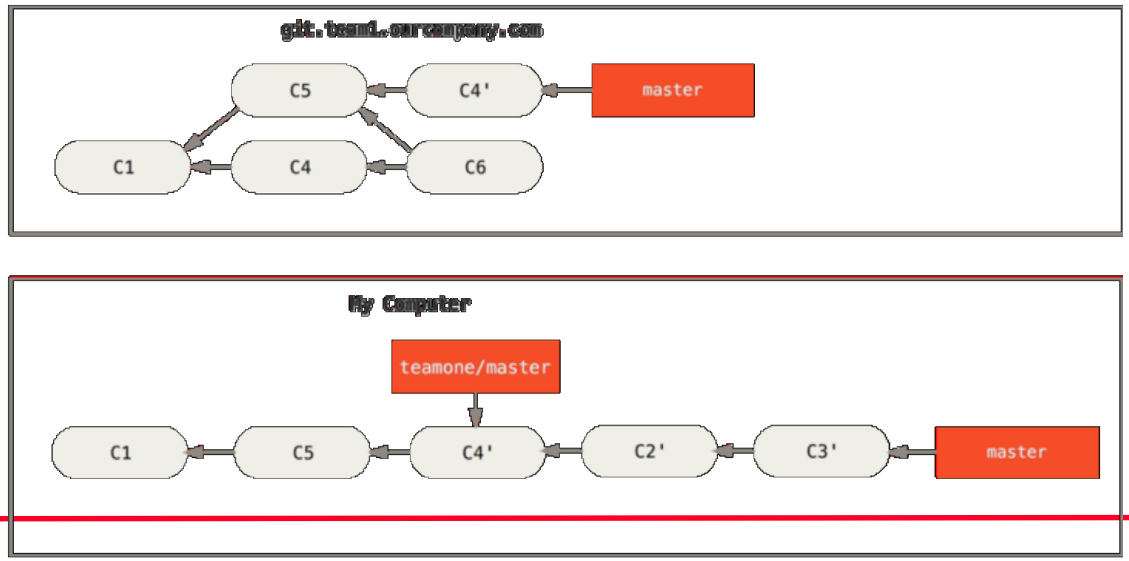
- If you do a git pull,
  - you'll create a merge commit which includes both lines of history,
  - your repository will look like this: → two commits that have the same author, date, and message → will be confusing





The Perils of Rebasing

- Rebase When You Rebase
  - git rebase teamone/master
    - Determine what work is unique to our branch (C2, C3, C4, C6, C7)
    - Determine which are not merge commits (C2, C3, C4)
    - Determine which have not been rewritten into the target branch (just C2 and C3, since C4 is the same patch as C4')
    - Apply those commits to the top of teamone/master



The Perils of Rebasing

- Rebase When You Rebase
  - If you or a partner does find it necessary at some point,
    - make sure everyone knows
    - to run git pull --rebase to try to make the pain after it happens a little bit simpler.
- Rebase vs. Merge
  - to rebase local changes you've made but haven't shared yet before you push them in order to clean up your story,
  - but never rebase anything you've pushed somewhere.

HW #2

- A, B 가 협업을 한다.
- A 가 지난 번 숙제를 GitHub 에 올린다.
- B 가 A 가 올린 숙제를 GitHub 에서 pull 한다.
- A는 지난번 숙제에서 print(“ CapstoneDesign 1”) 을 마지막 라인에 추가한다.
- B도 A에서 받은 소스에서 마지막 라인에 print(“CapstoneDesign2”) 을 추가한다
- B 가 만든 SW 를 GitHub 에 올린다.
- A는 B가 만든 S/W 를 fetch 한다.
- 그리고 자신이 변경한 것을 merge 한 후 서버에 올린다.
- B는 그것을 가지고 pull 해서 사용한다.