Tutorial #4:
Source Control

CS374: Introduction to HCI

2017. 5. 1

Hyeungshik Jung

Before start

- This tutorial will not cover how to create repo, make branch, merge, push…
    - You can learn it by yourself with the materials in the last slide
- Instead, I'll try to focus on basic concepts.

Outline

- Why do we need Version Control System
- Git
- Repository / Commit / Branch
- Merge / Rebase / Conflict
- Remote repository
- Working with others
- Recommended Materials

Why do we need Version Control System

- Version Control System
  - Store versions of file and retrieve them
- Why we need it?
  - Roll back a change that caused a bug
  - Separate deployed version from development version
  - Document history of who did what when

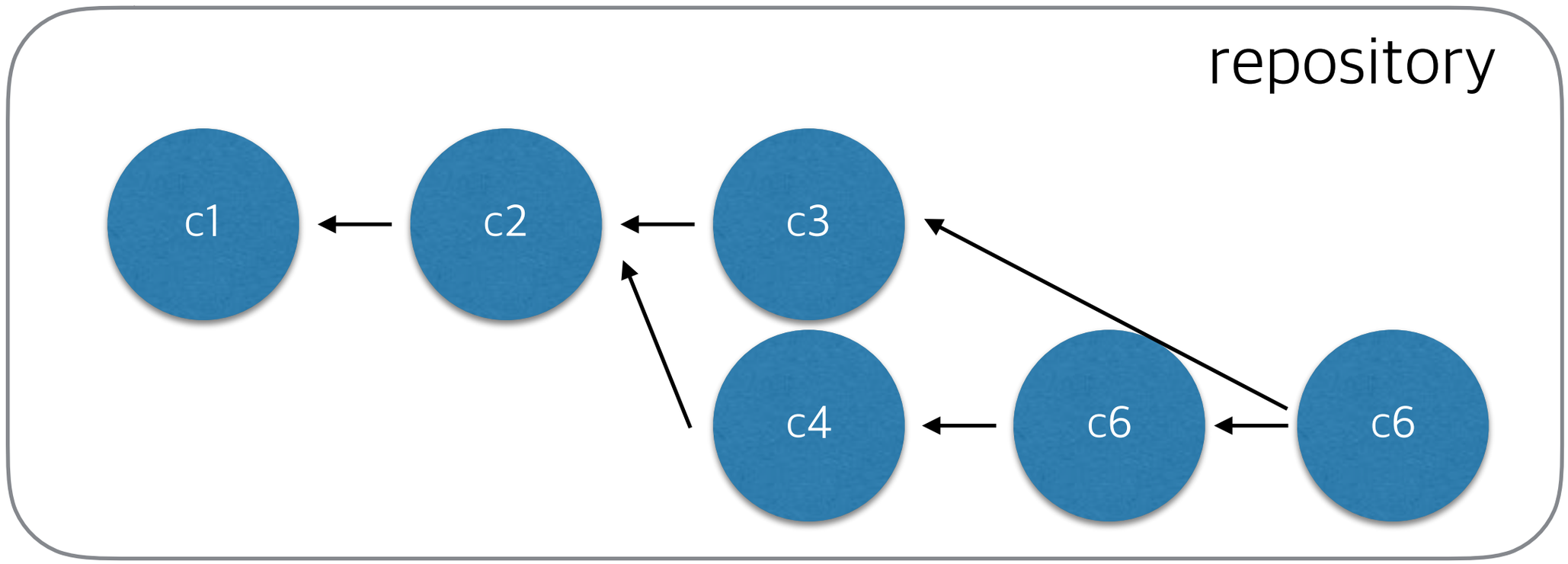
Git

- Built by Linus Torvalds at 2005
- De facto standard in open source software
- But Git is not the only Version Control System.

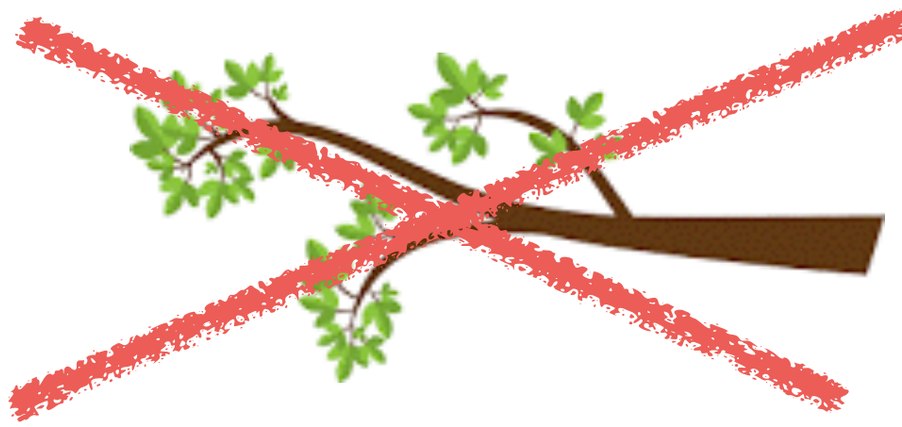
https://www.linux.com/blog/10-years-git-interview-git-creator-linus-torvalds

Repository

- **Repository** stores history of development
  - “Directed Acyclic Graph with Root”
- Node (“**Commit**”): A snapshot of codes from some point
- Edge: Point to the parent node

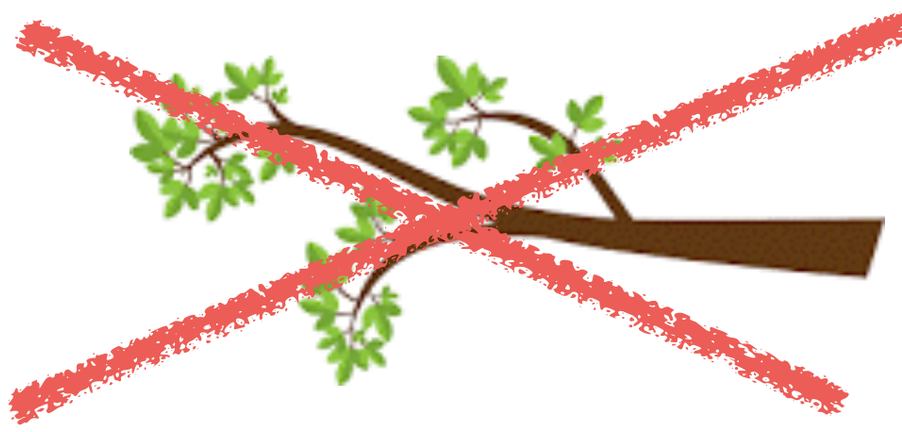


Branch

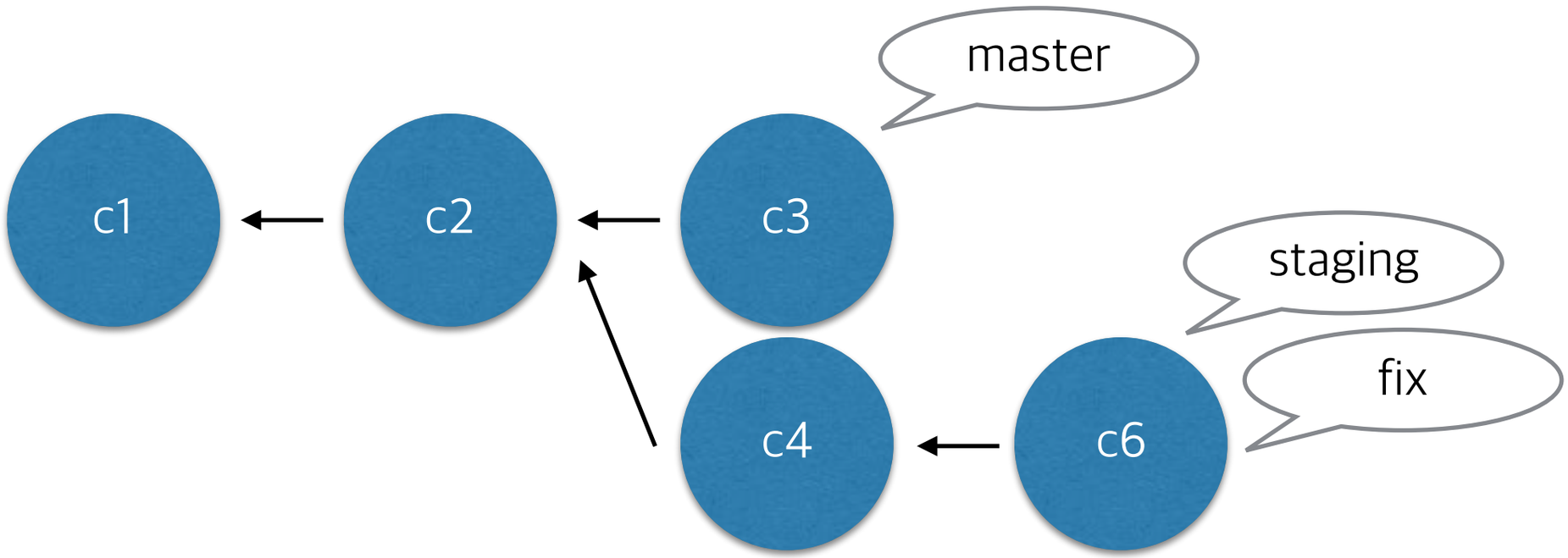


- Pointer for **A** commit (Not a series of commits)

Branch

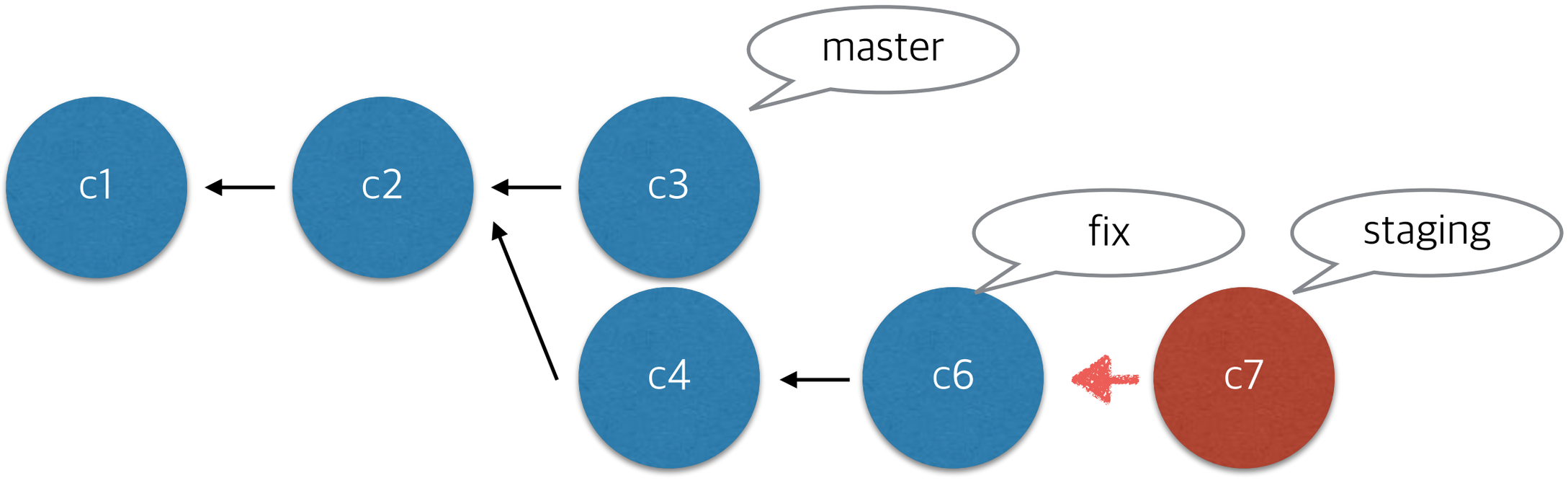


- Pointer for **A** commit (Not a series of commits)
- Creating a branch = Making a new pointer



Commit

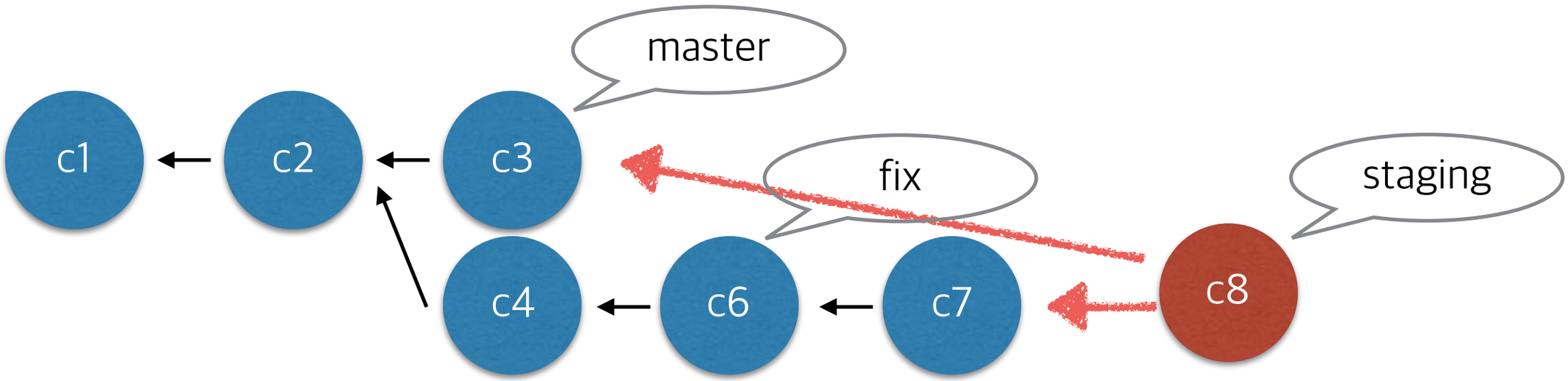
- Add a “fresh” commit with some change



Merge

- Add a commit by merging two branches (3-way-merge)

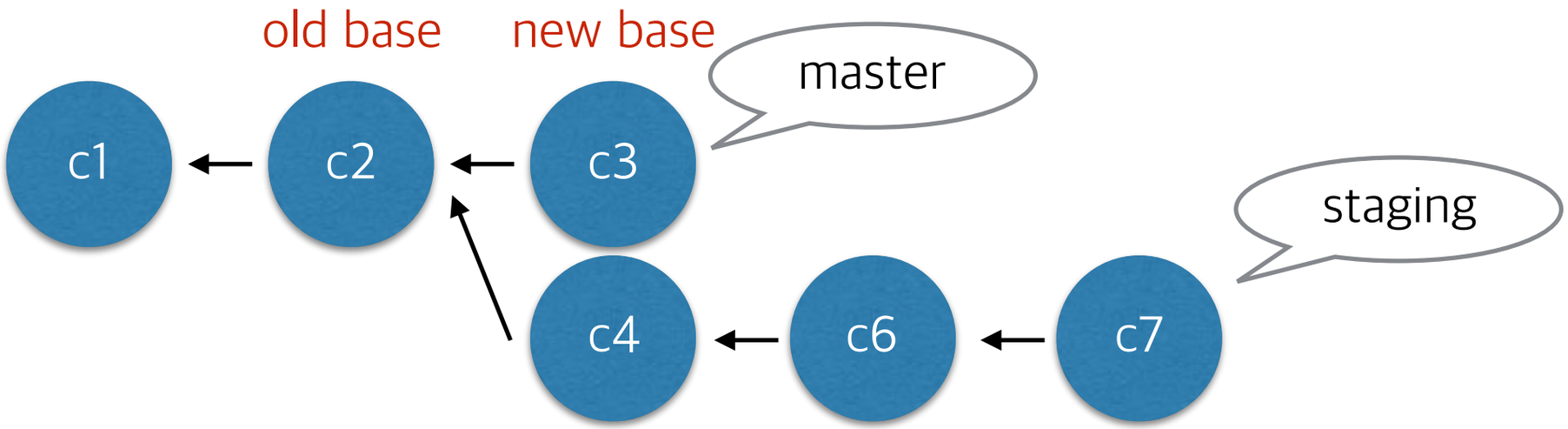
```
(staging) git merge master
```



Rebase

- “Re” + “Base”: Set new base
- Base: common ancestor

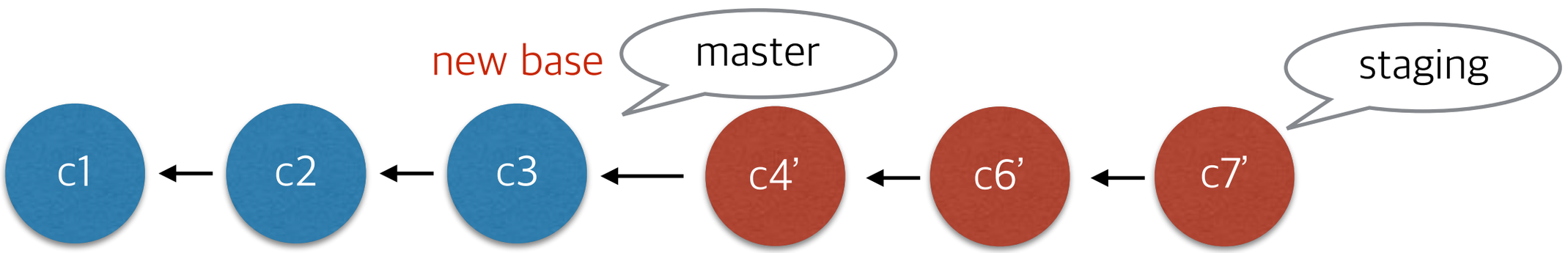
```
(staging) git rebase master
```



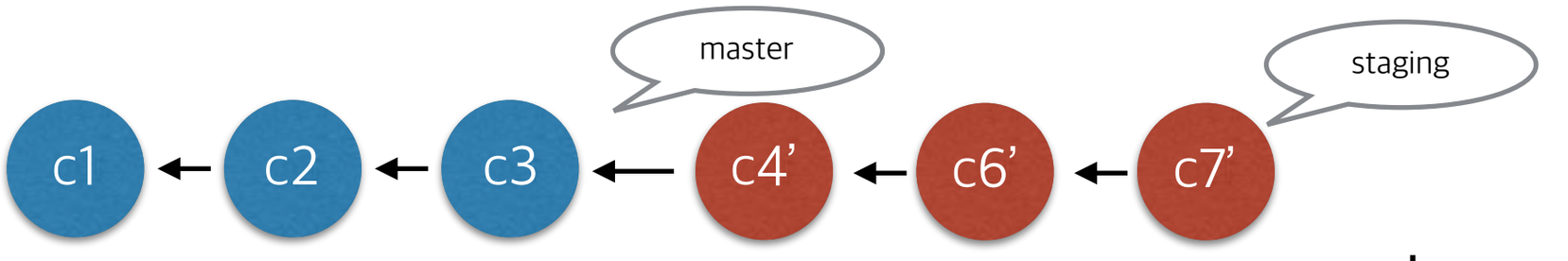
Rebase

- “Re” + “Base”: Set new base
- Base: common ancestor

```
(staging) git rebase master
```

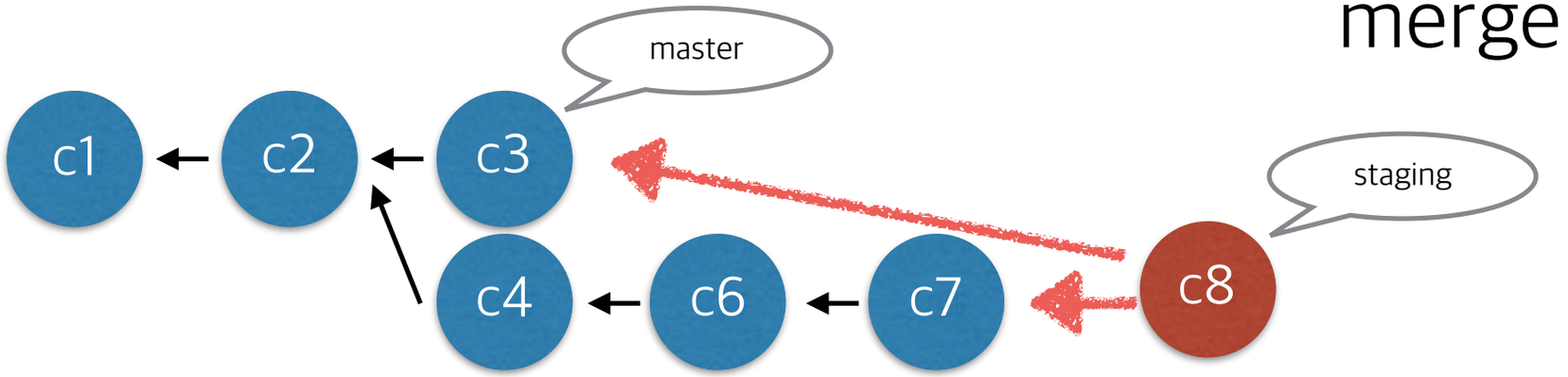


Merge and Rebase



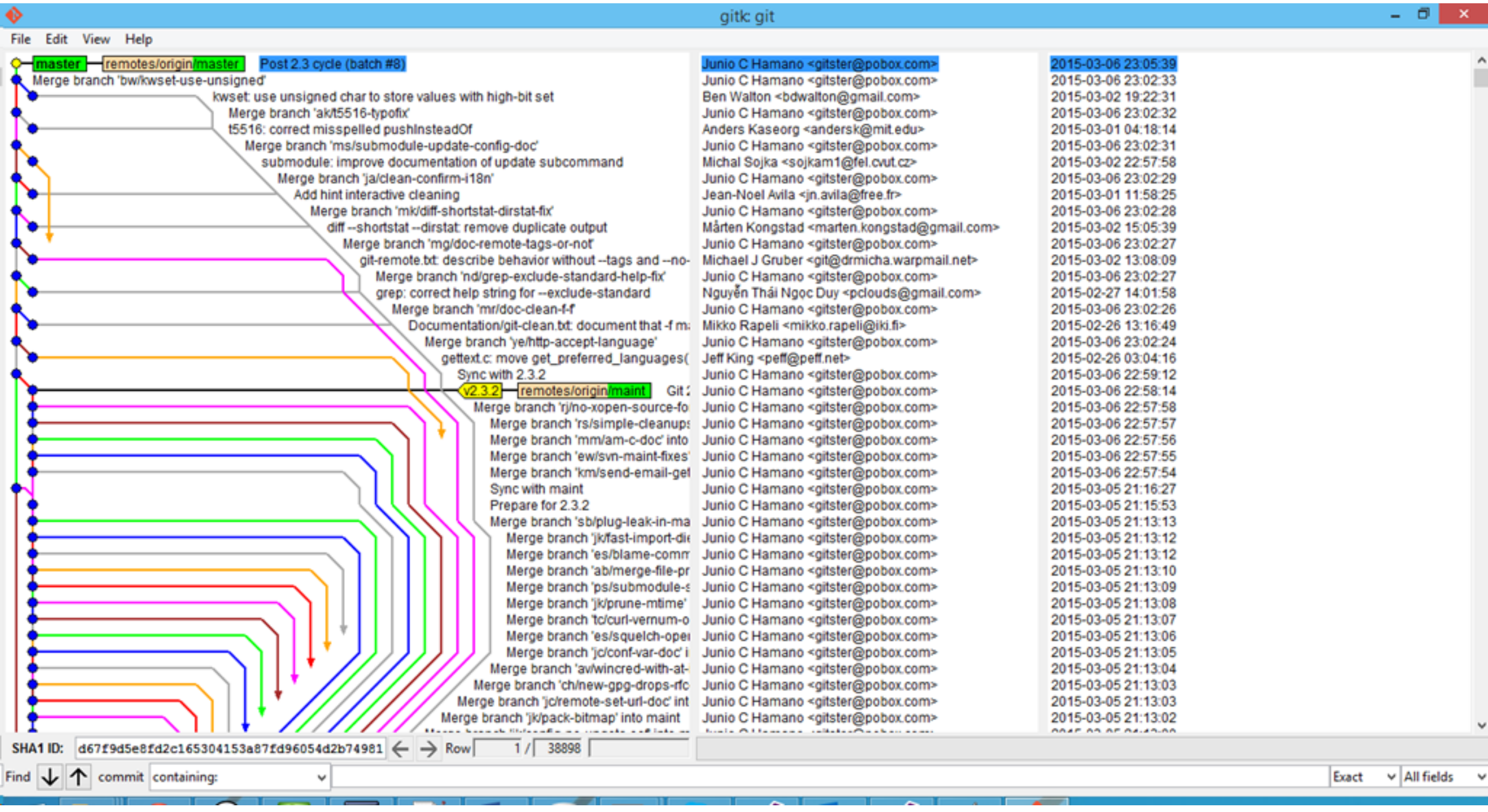
rebase

merge



- “Staging” branch after rebase(c7’) and merge(c8) has same content.

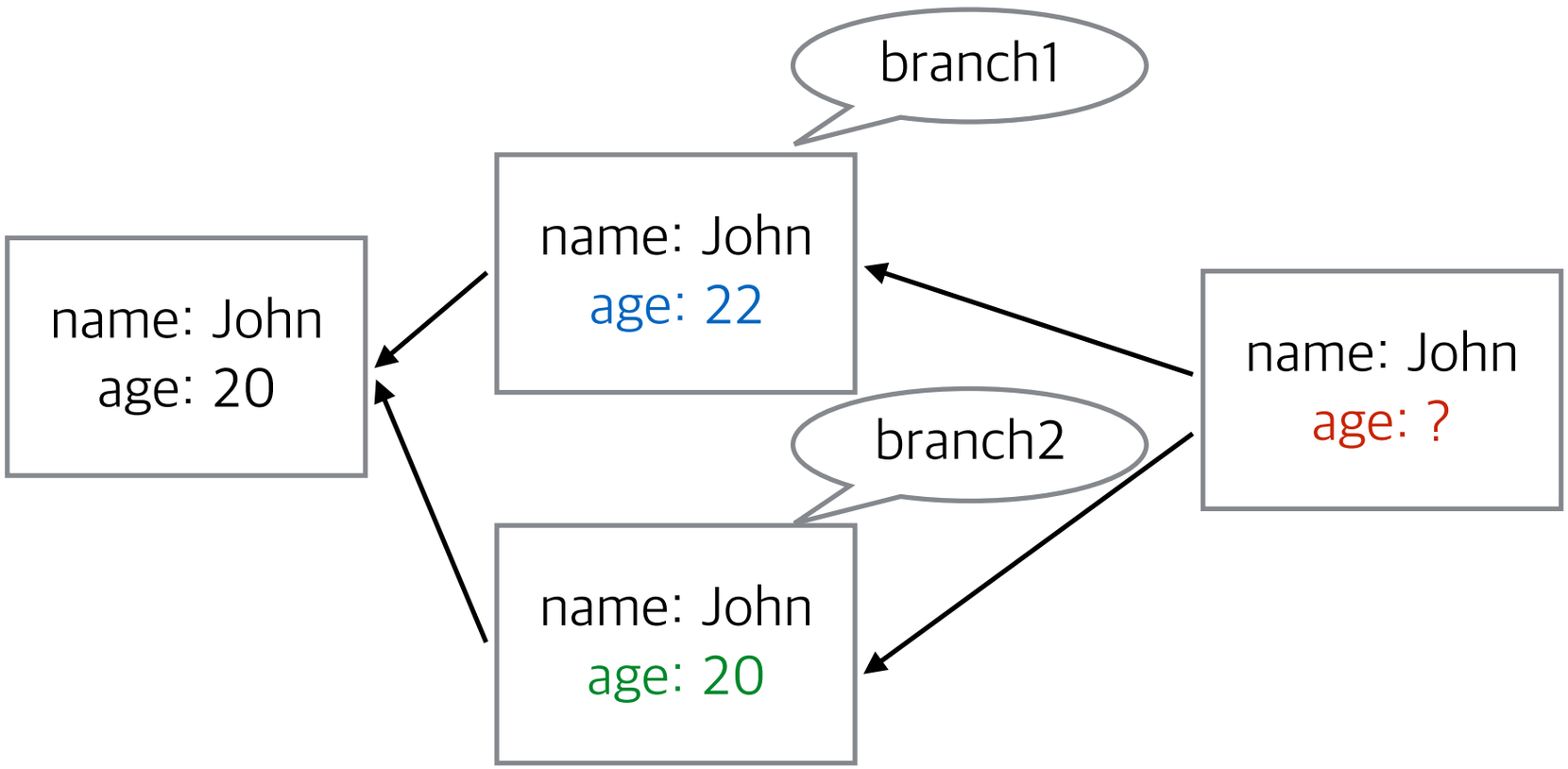
- But generally rebase is more recommended, since it’s history is much neater.



http://www.tugberkugurlu.com/archive/resistance-against-london-tube-map-commit-history-a-k-a--git-merge-hell

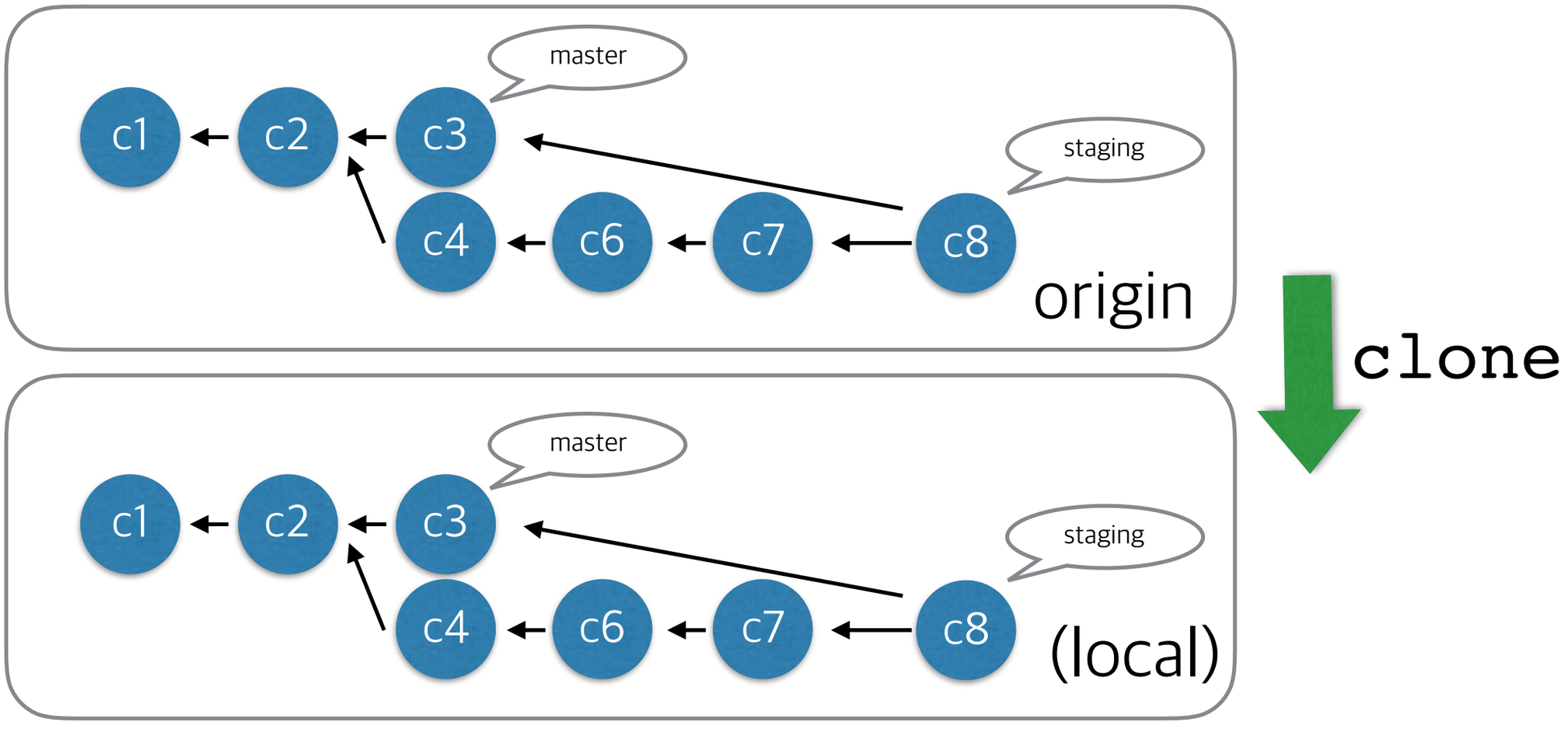
Conflict

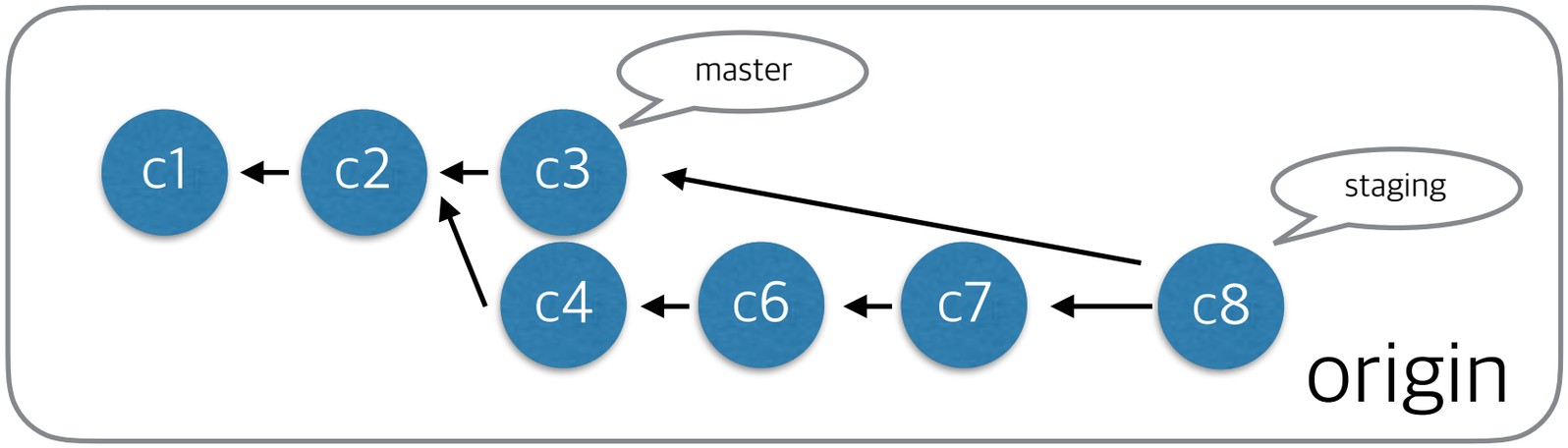
- Don’t get scared
- Conflict happens when Git failed to merge
- Usually when two commits changed same part of a file
- Developer should choose which one to follow

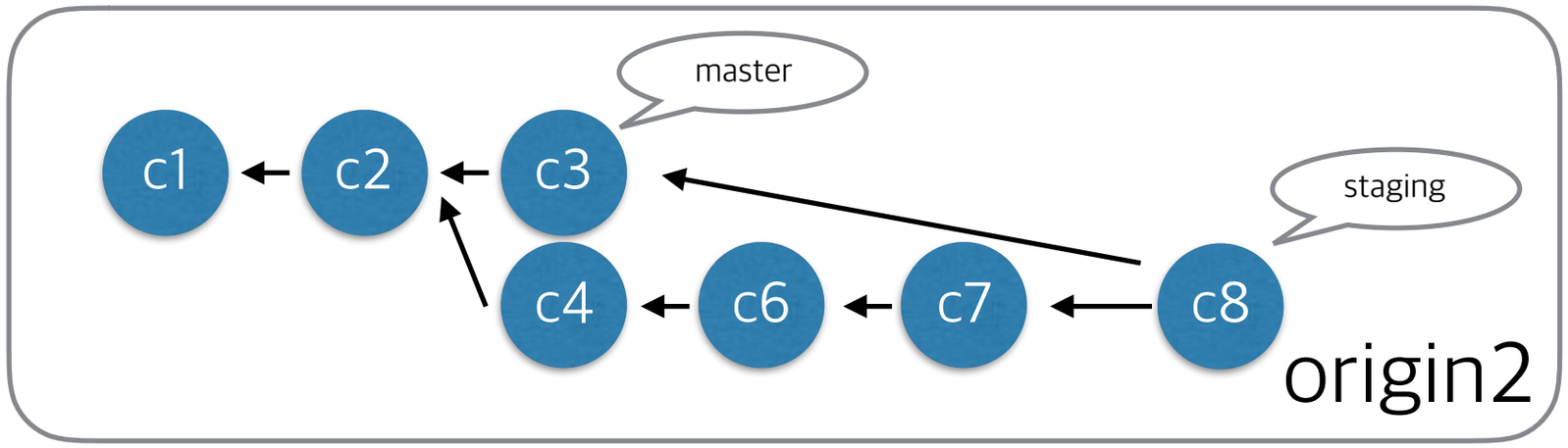


Remote Repository

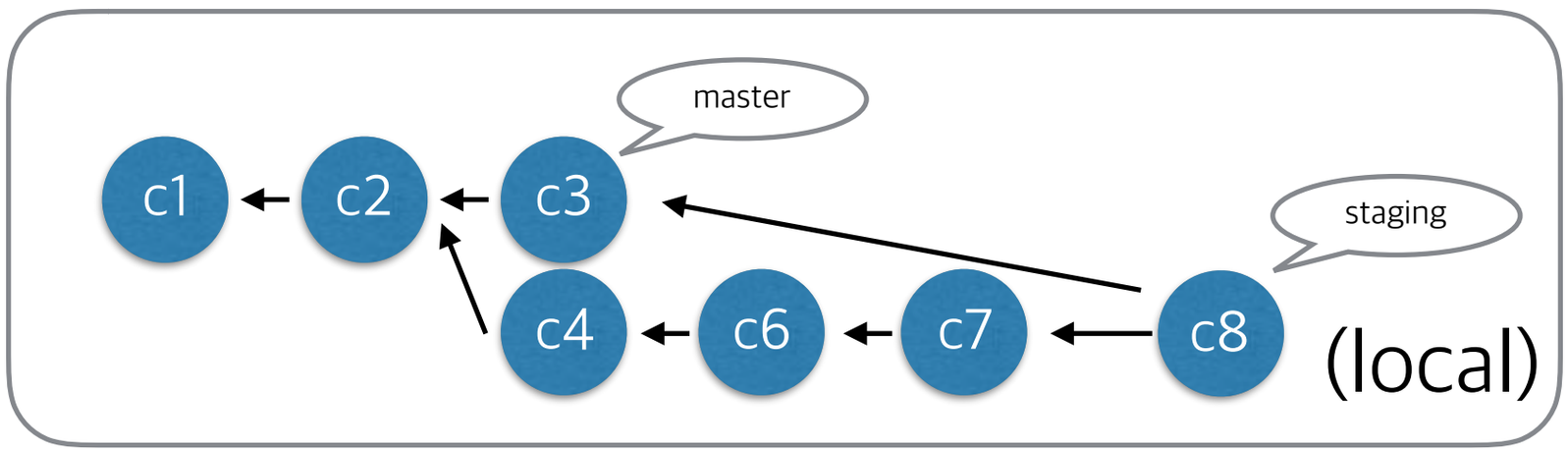
- Git is a distributed version control system
  - Repository can be at server or developer's system
  - But they are equal repositories





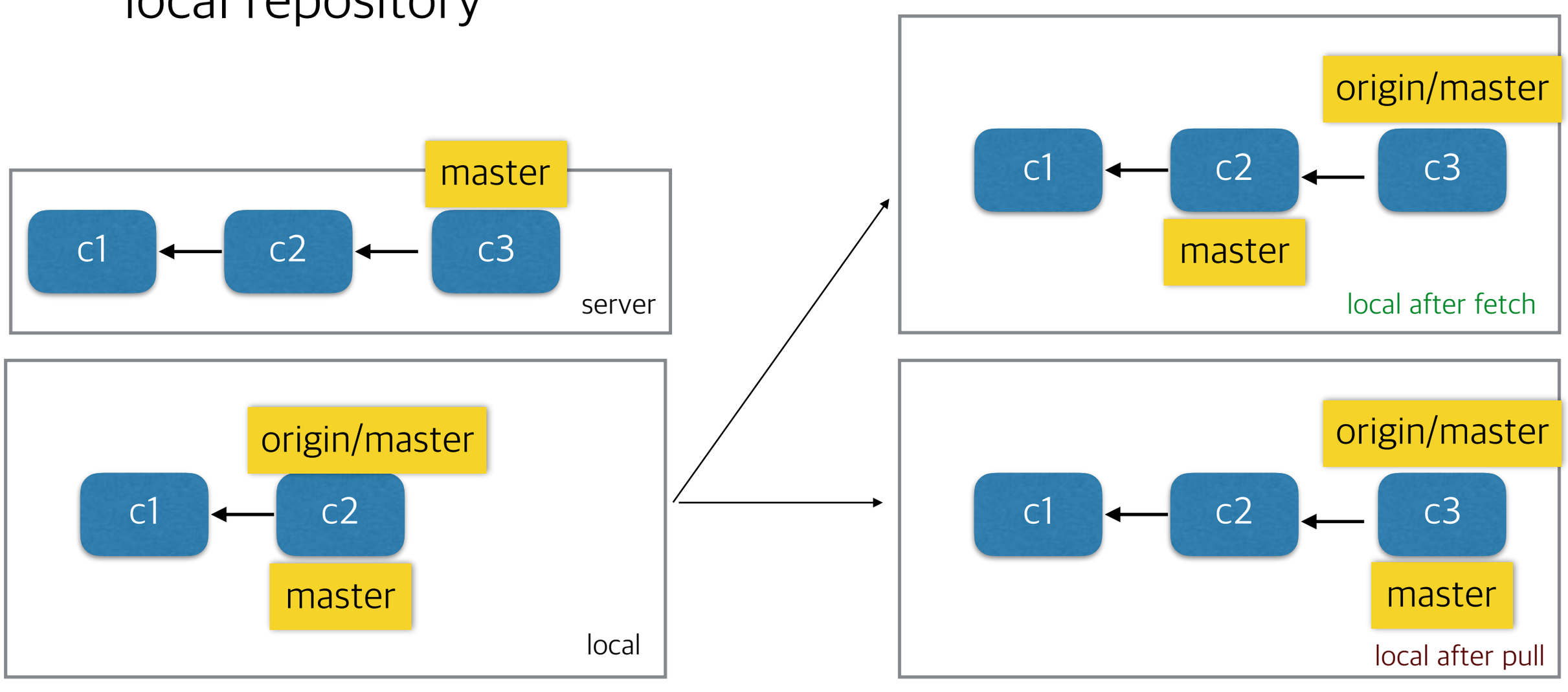


We can have multiple remote repositories.
`origin` is just one of them.



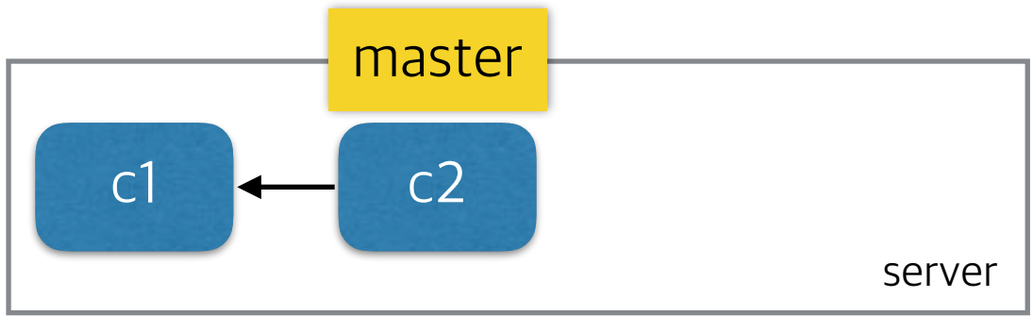
Fetch, Pull

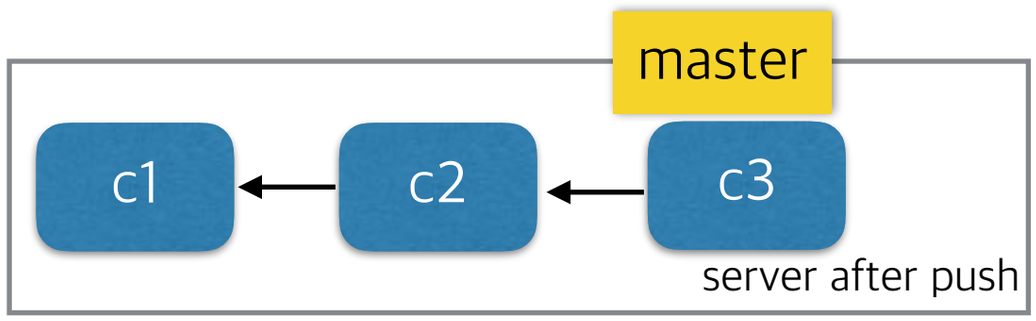
- Fetch: Just download all the content from remote repository
- Pull: Fetch and apply commits from remote repository to local repository

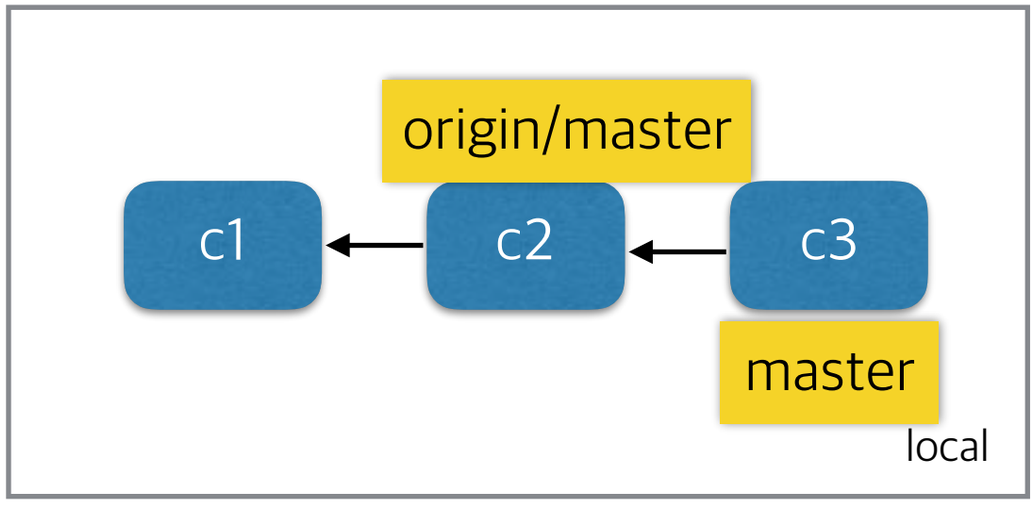


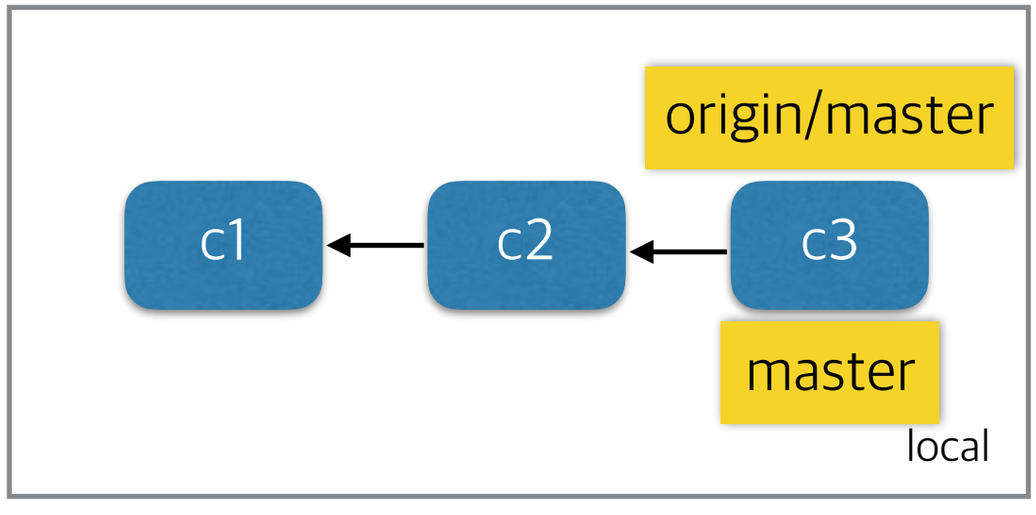
Push

- Apply local commits to remote repo





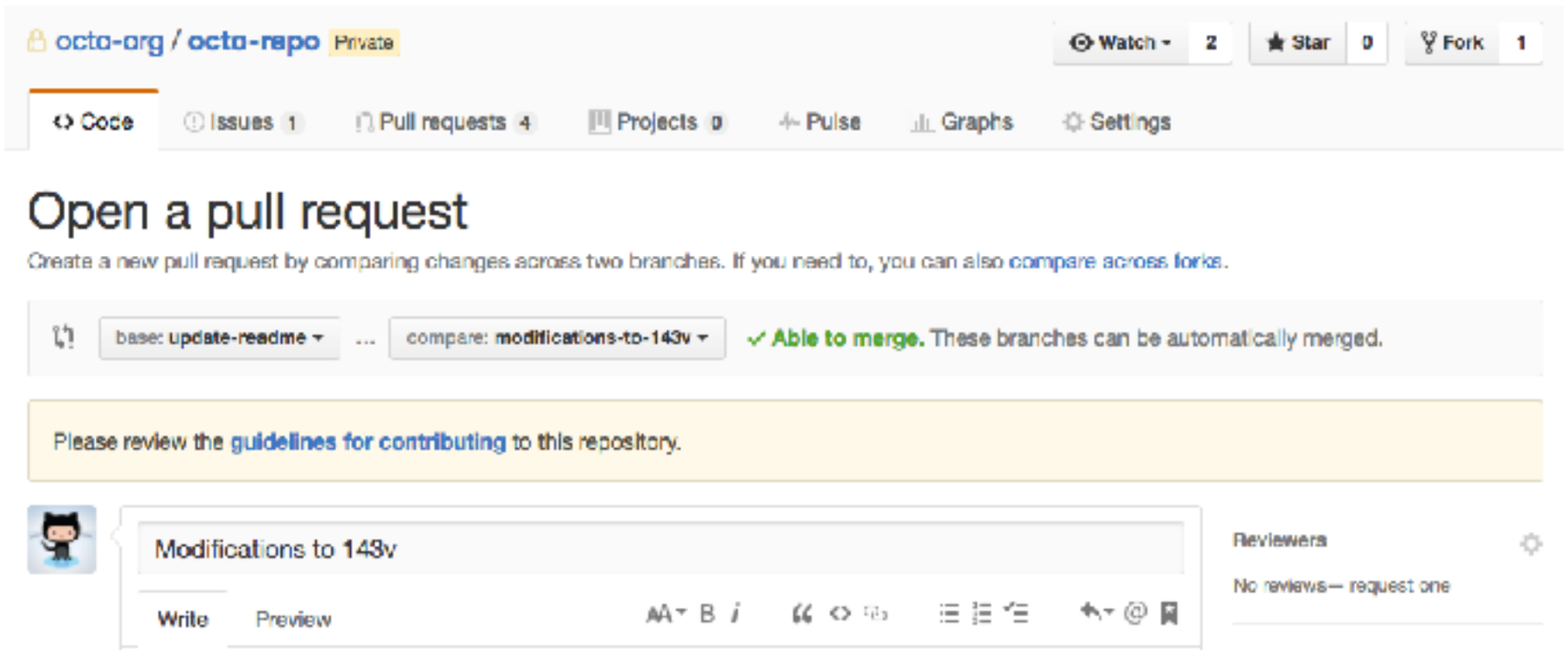


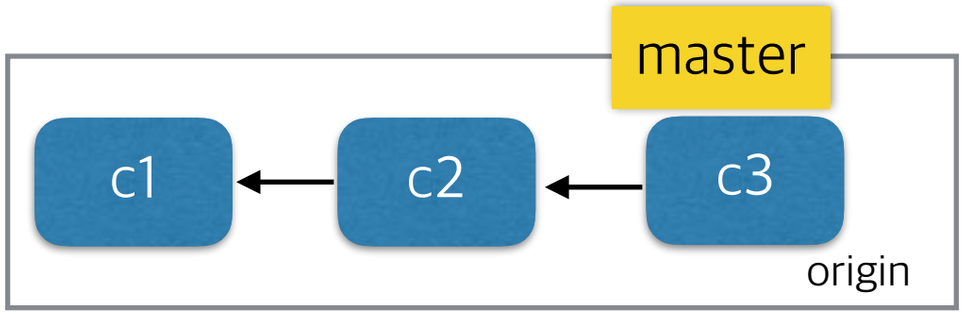


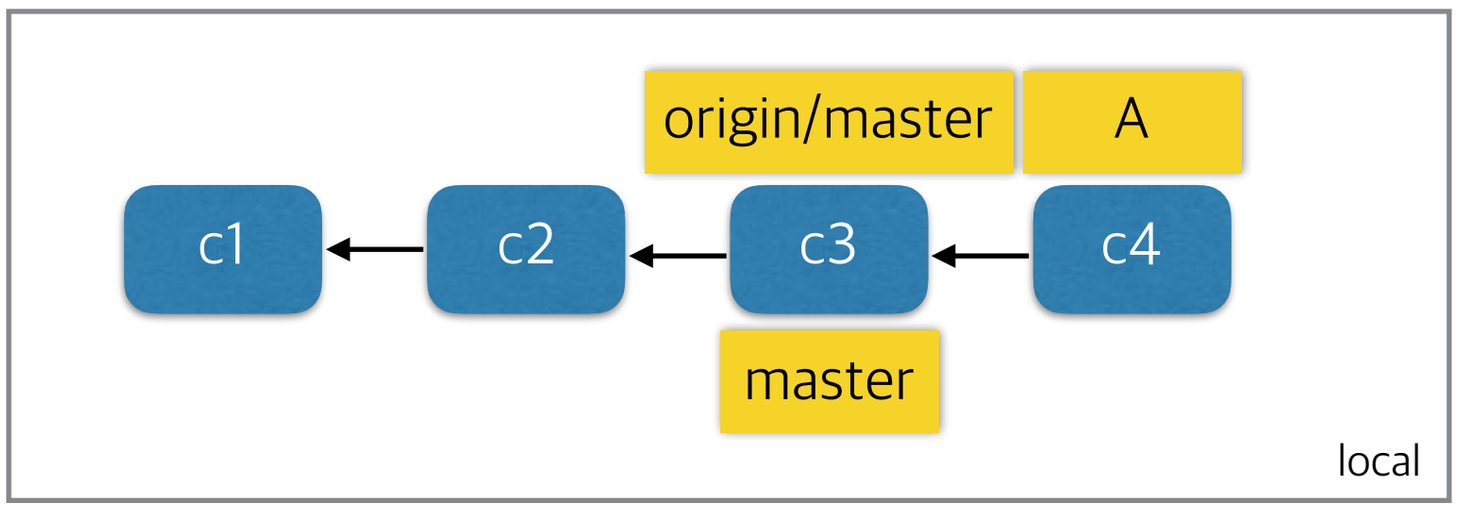
Pull Request

- What if a developer doesn't have permission to push? (Or we cannot give permission to anyone?)

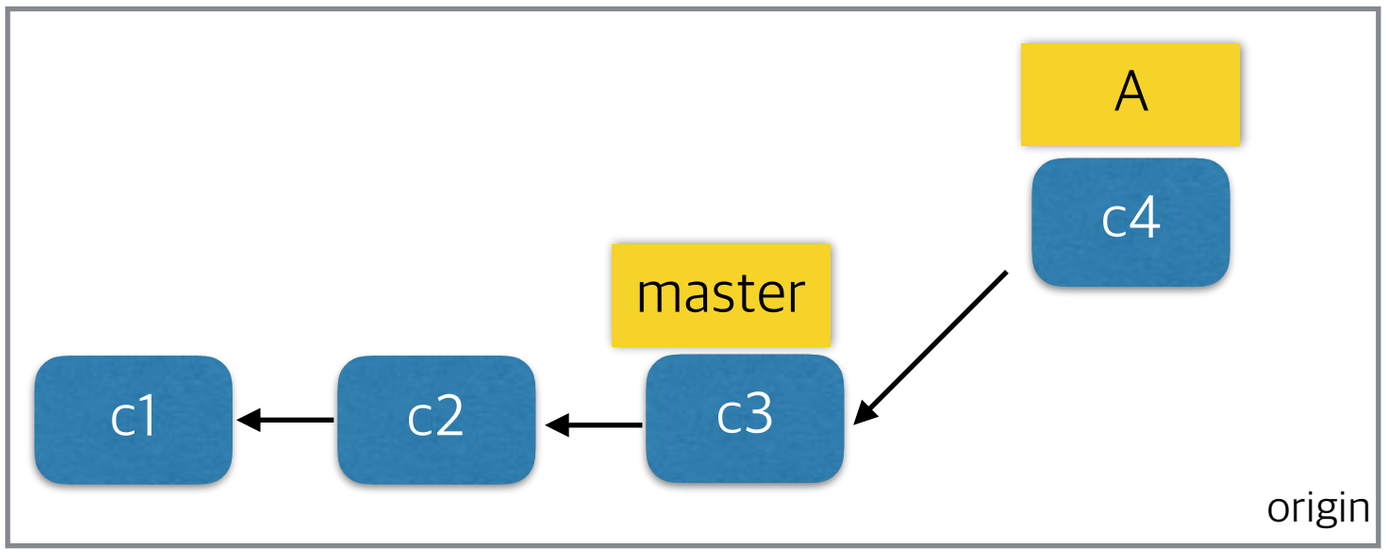
- Asking to apply changes to a branch.



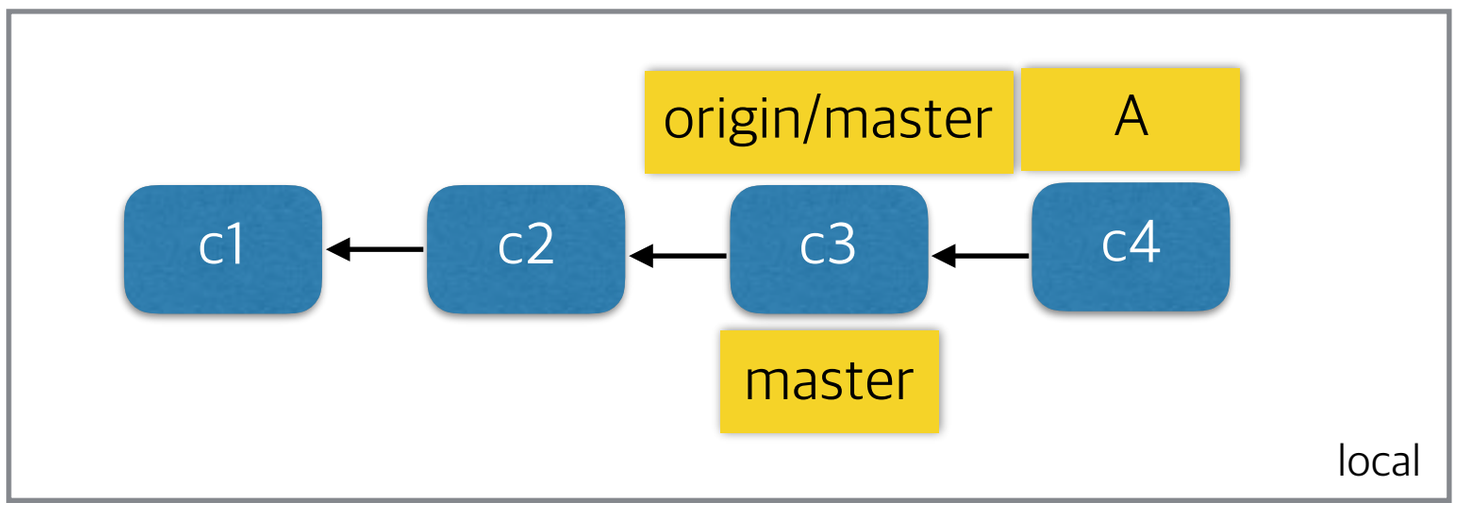


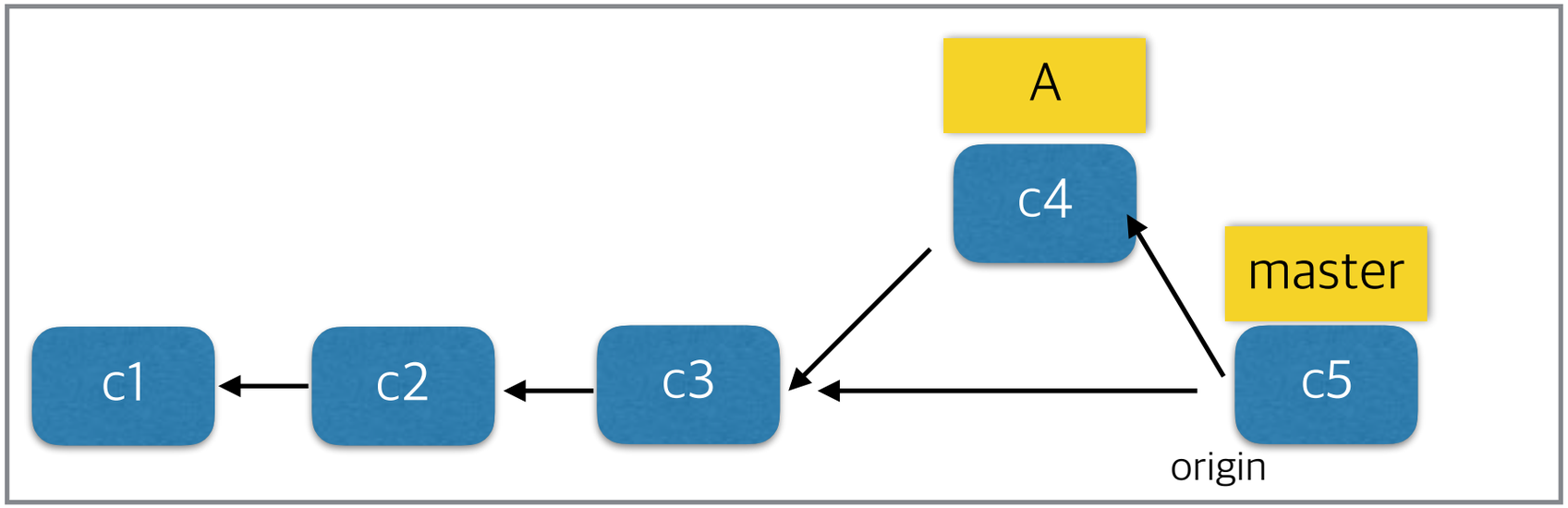


Has a new local branch A,
but cannot push it to master

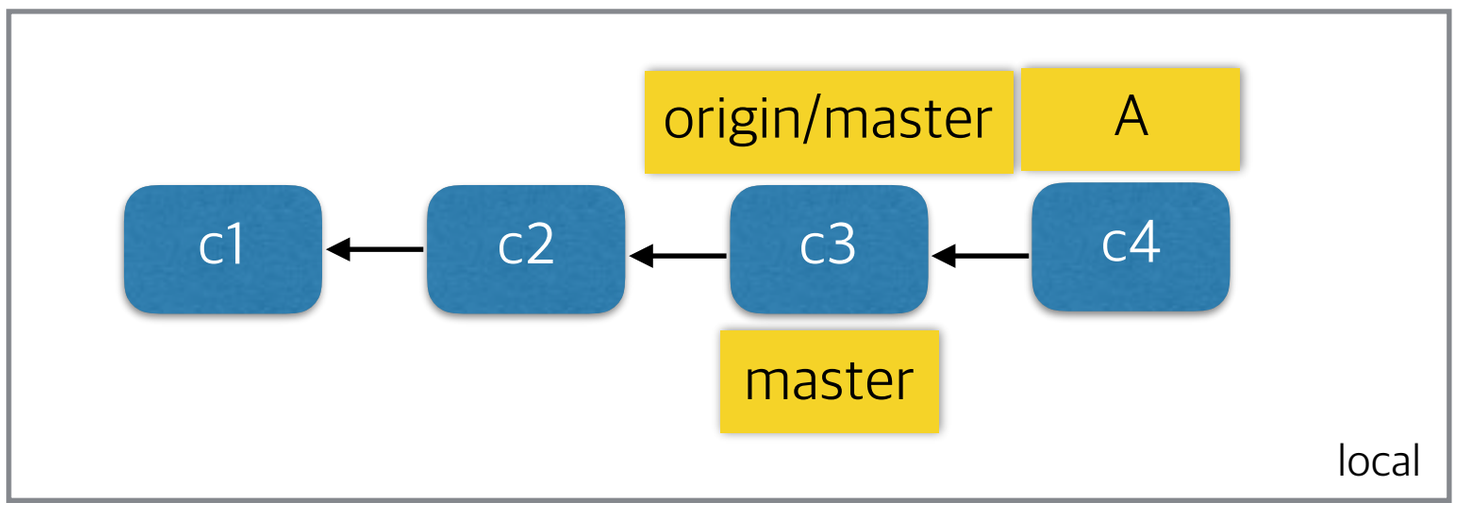


Upload new branch A,
and make **PULL REQUEST**



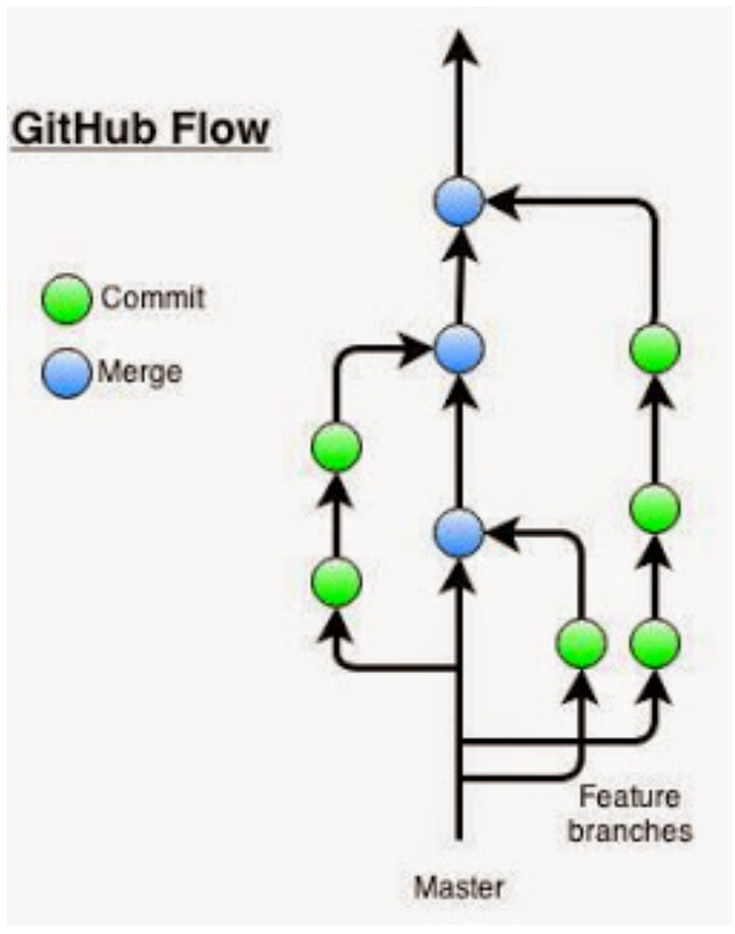


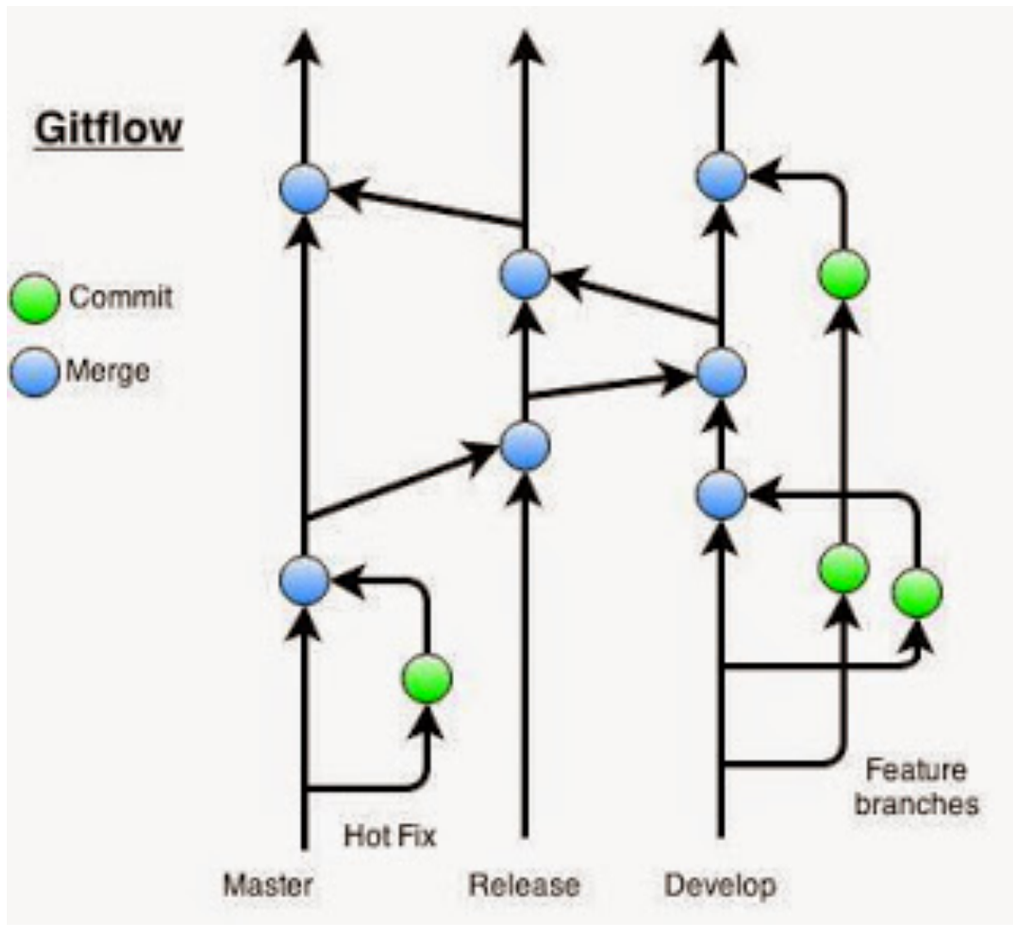
After PR is accepted, new branch is merged to master



Working with others

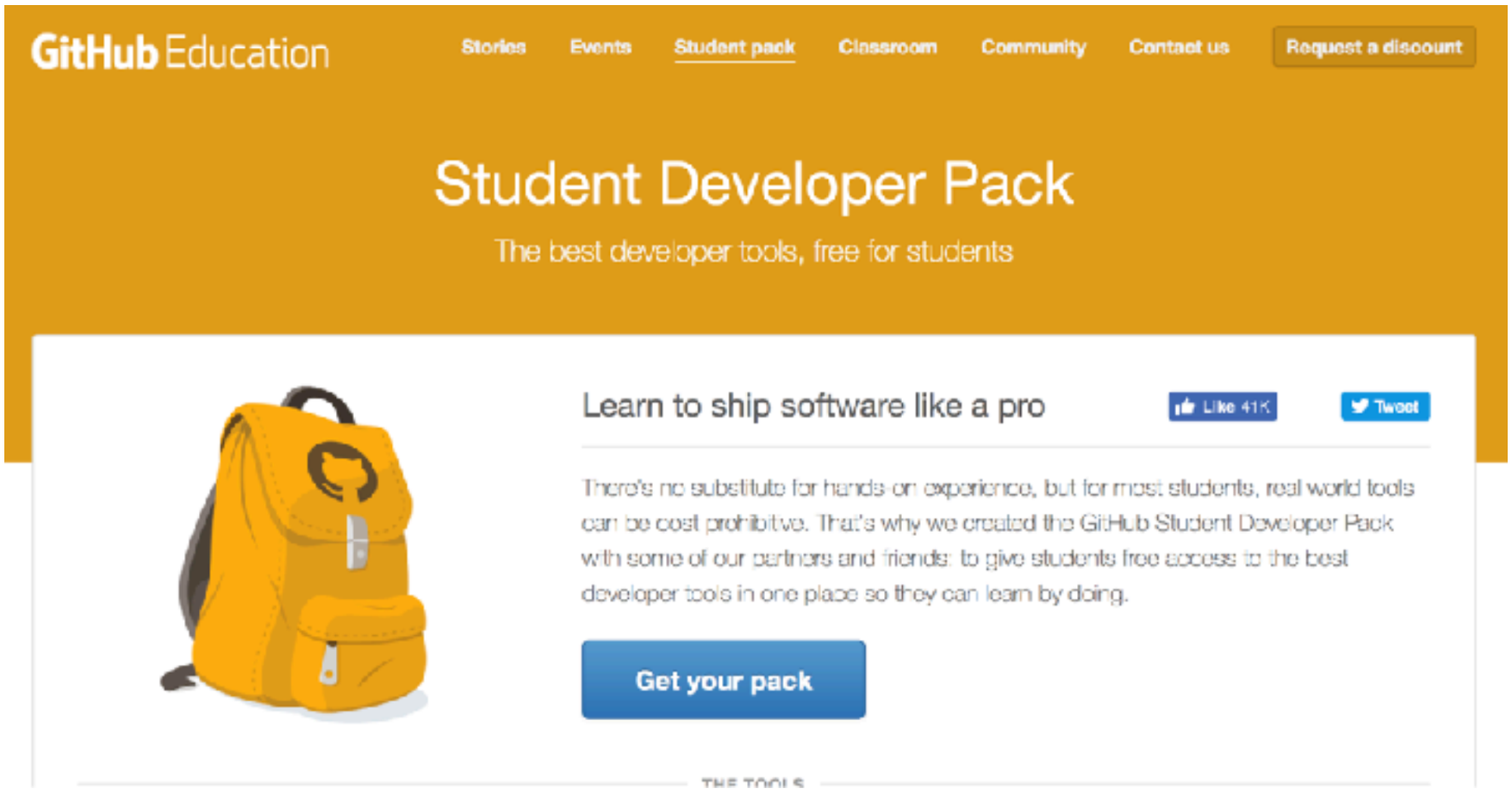
- There are many workflows on the web.
- But in every workflow, “master” branch is production-ready.





http://blog.endpoint.com/2014/05/git-workflows-that-work.html

Since you are a student



Free unlimited private repositories + alpha

Recommended Materials

- Pro Git online book (https://git-scm.com/book/en/v2)
- https://try.github.io
  (Easy interactive tutorial)
- http://learngitbranching.js.org/
  (Difficult interactive tutorial)
- (Korean) Effective Git
  (from NDC 2016. Includes so many useful tips)