

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

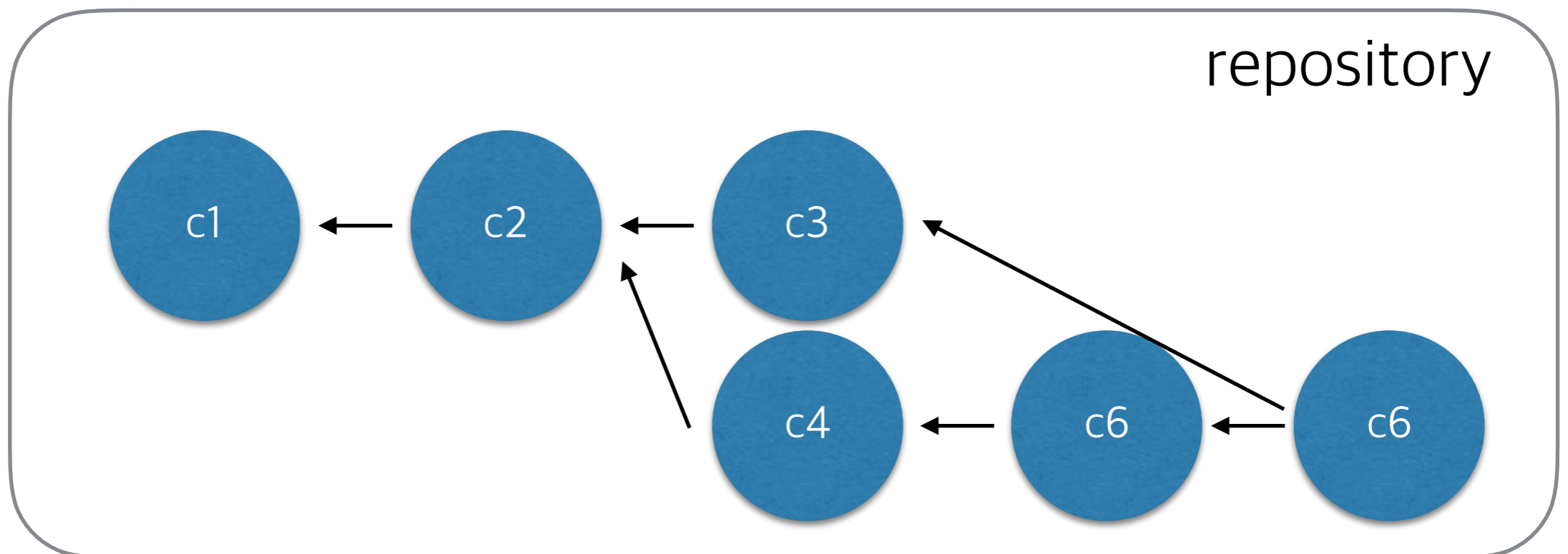
- Version Control System
 - Store versions of file and retrieve them
- Why we need it?
 - Roll back a change that caused a bug
 - Separate deployed version from development version
 - Document history of who did what when

Git

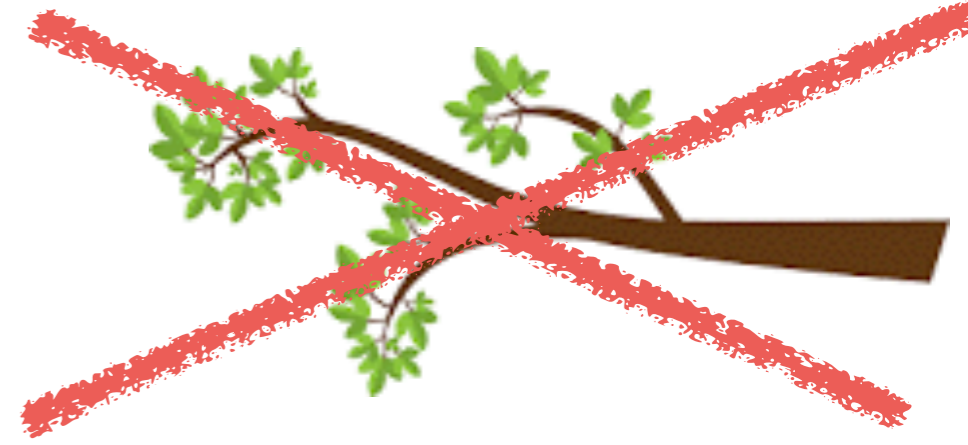
- Built by Linus Torvalds at 2005
- De facto standard in open source software
- But Git is not the only Version Control System.

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

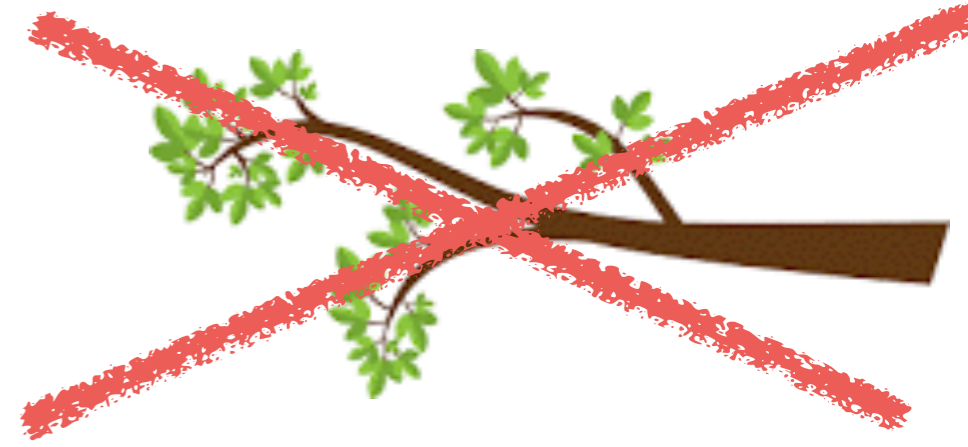


Branch

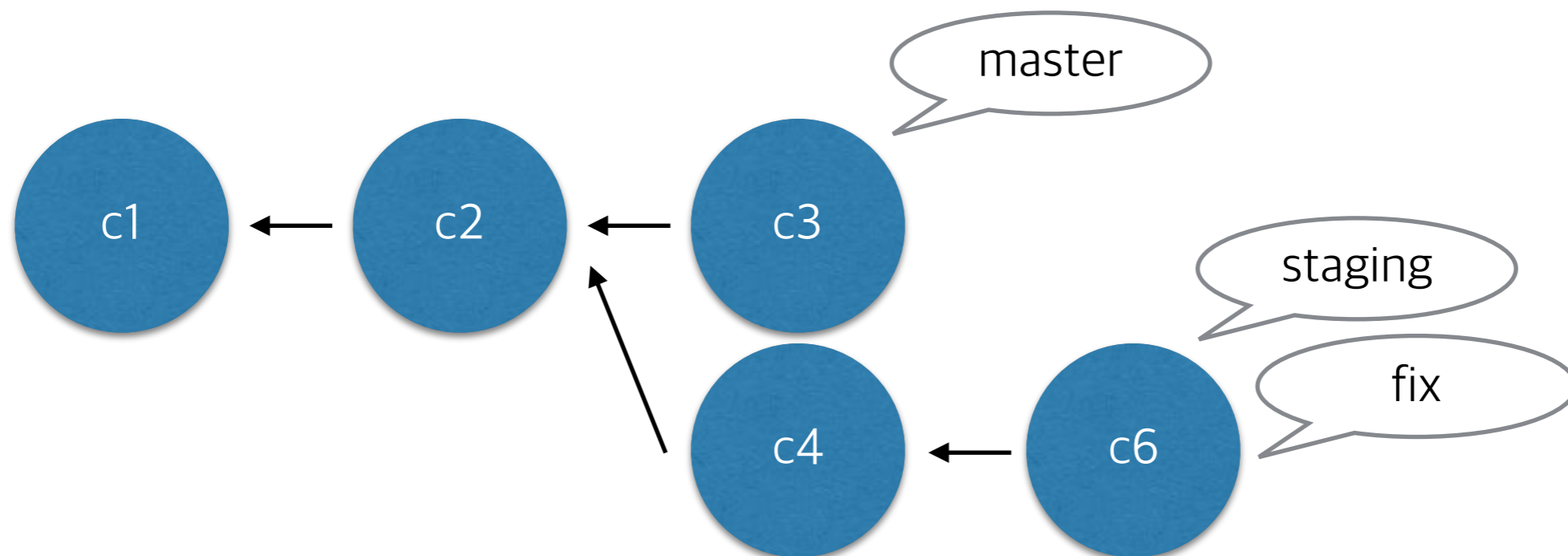


- Pointer for **A** commit (Not a series of commits)

Branch

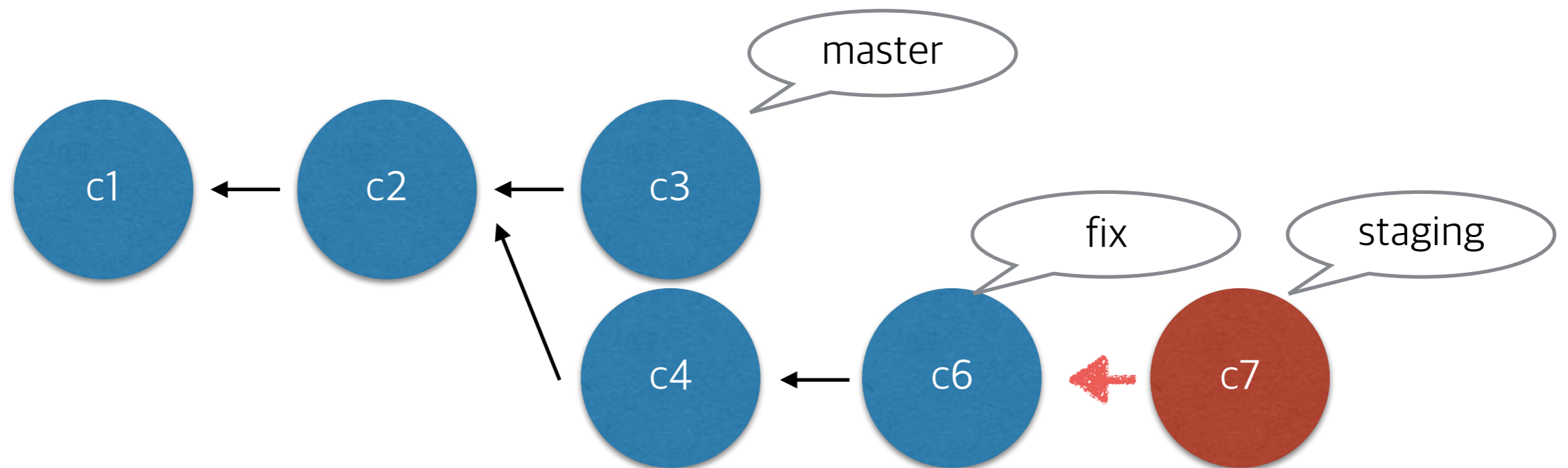


- Pointer for **A** commit (Not a series of commits)
- Creating a branch = Making a new pointer



Commit

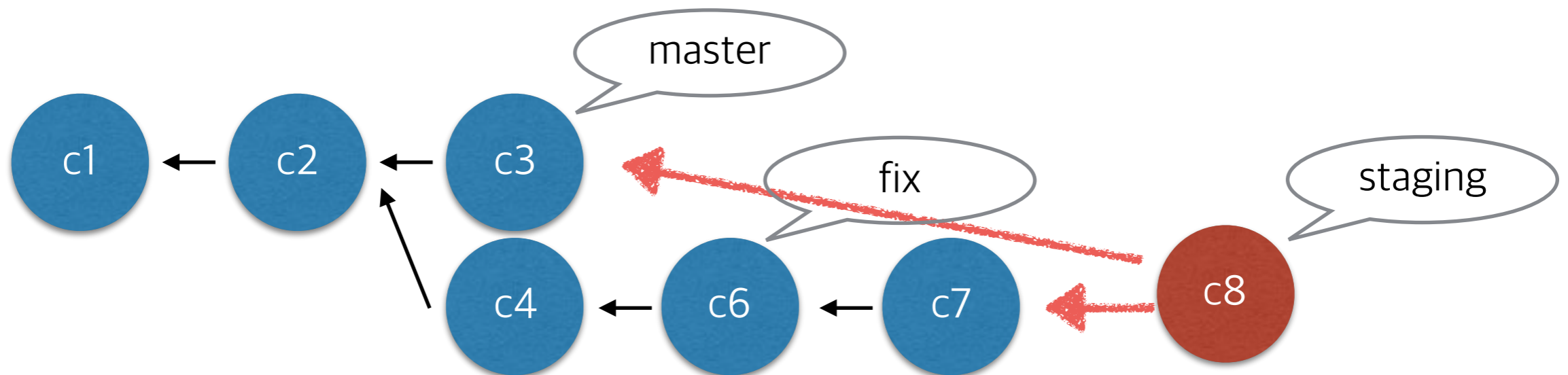
- Add a “fresh” commit with some change



Merge

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

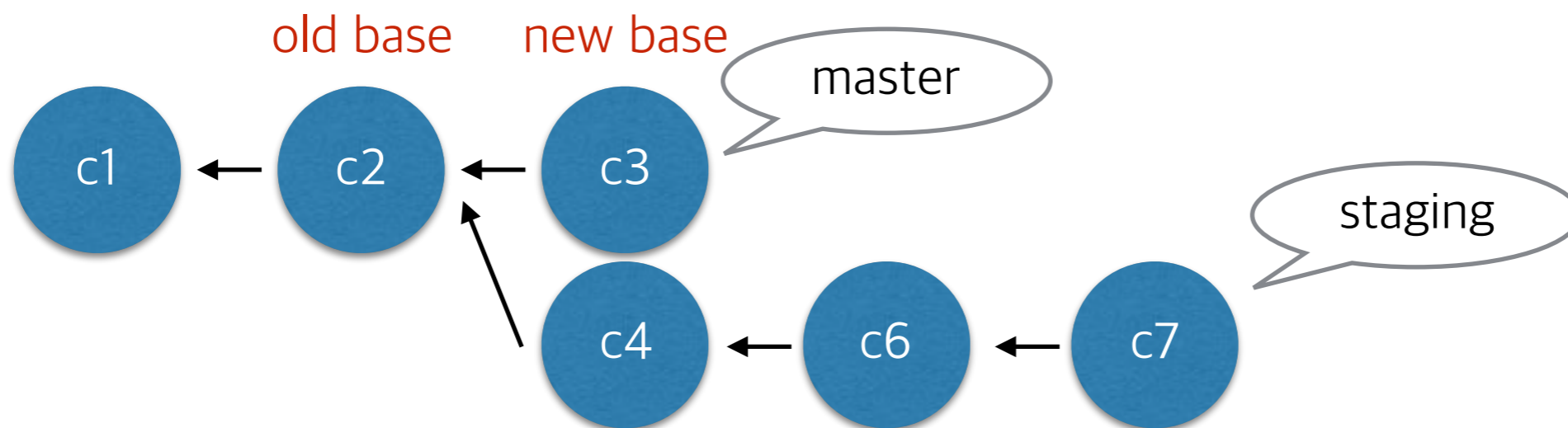
`(staging) git merge master`



Rebase

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

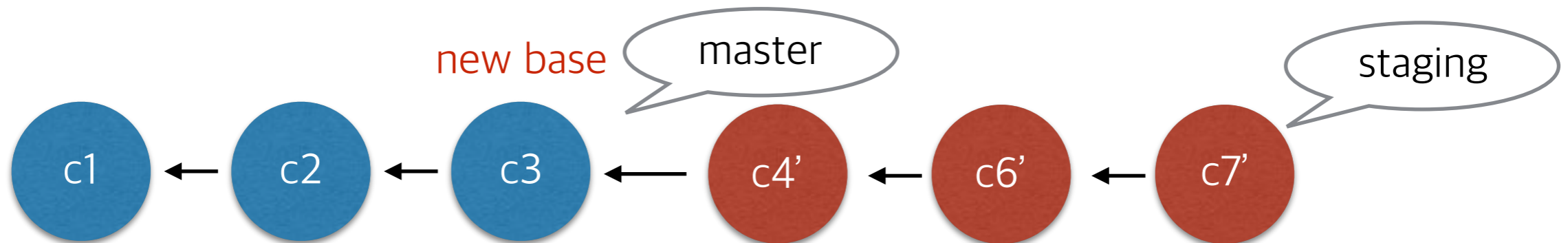
`(staging) git rebase master`



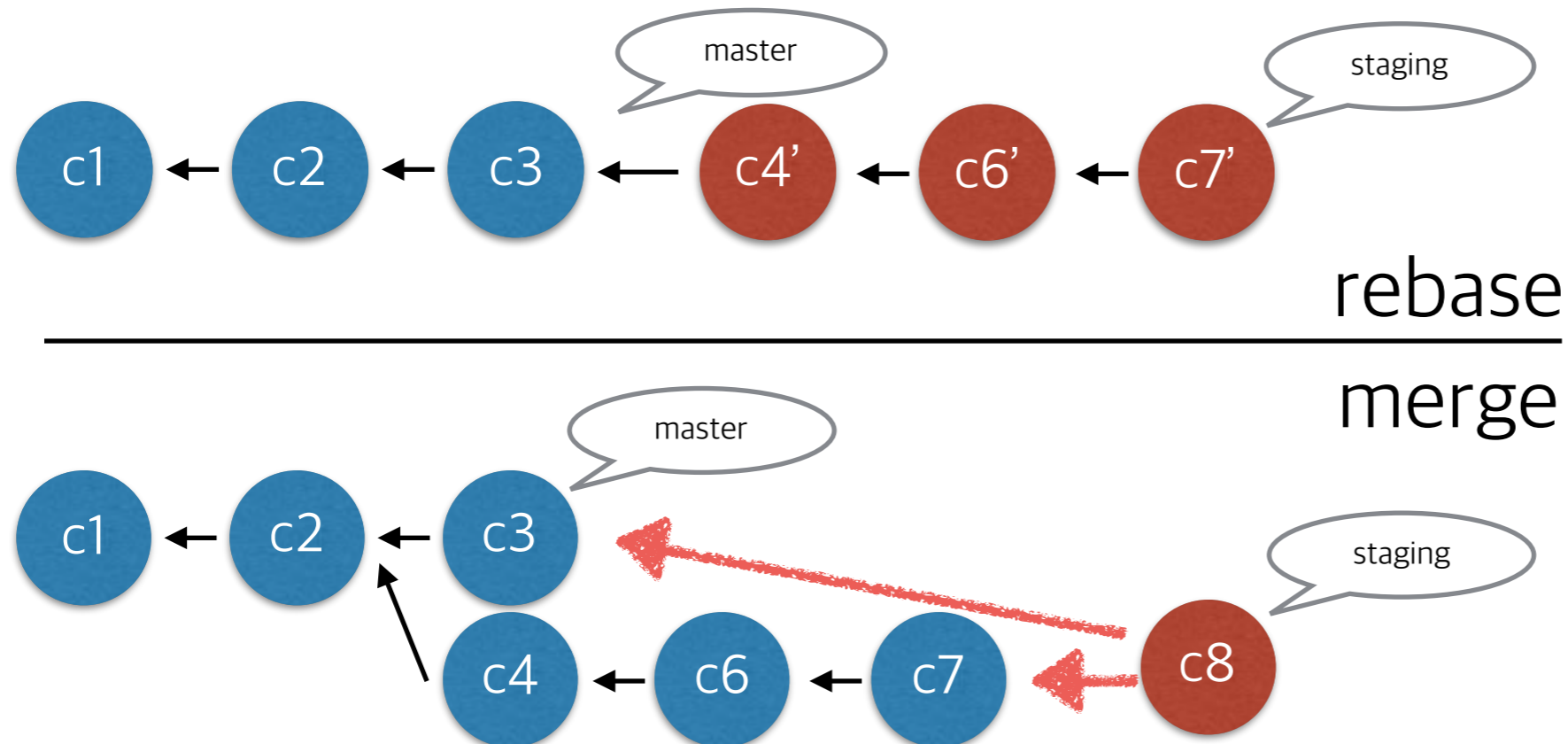
Rebase

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

`(staging) git rebase master`



Merge and Rebase



- “Staging” branch after rebase(c7’) and merge(c8) has same content.
- But generally rebase is more recommended, since it’s history is much neater.

gitk git

File Edit View Help

master remotes/origin/master Post 2.3 cycle (batch #8)

Merge branch 'bw/kwset-use-unsigned'

kwset: use unsigned char to store values with high-bit set

Merge branch 'ak/t5516-typofix'

t5516: correct misspelled pushInsteadOf

Merge branch 'ms/submodule-update-config-doc'

submodule: improve documentation of update subcommand

Merge branch 'ja/clean-confirm-i18n'

Add hint interactive cleaning

Merge branch 'mk/diff-shortstat-dirstat-fix'

diff --shortstat --dirstat: remove duplicate output

Merge branch 'mg/doc-remote-tags-or-not'

git-remote.txt: describe behavior without --tags and --no-

Merge branch 'nd/grep-exclude-standard-help-fix'

grep: correct help string for --exclude-standard

Merge branch 'mr/doc-clean-f-f'

Documentation/git-clean.txt: document that -f m-

Merge branch 'ye/http-accept-language'

gettext.c: move get_preferred_languages()

Sync with 2.3.2

v2.3.2 remotes/origin/maint Git

Merge branch 'rj/no-xopen-source-fo'

Merge branch 'rs/simple-cleanup'

Merge branch 'mm/am-c-doc' into

Merge branch 'ew/svn-maint-fixes'

Merge branch 'km/send-email-get'

Sync with maint

Prepare for 2.3.2

Merge branch 'sb/plug-leak-in-ma'

Merge branch 'jk/fast-import-die'

Merge branch 'es/blame-comm'

Merge branch 'ab/merge-file-pr'

Merge branch 'ps/submodule-s'

Merge branch 'jk/prune-mtime'

Merge branch 'tc/curl-vernum-o'

Merge branch 'es/squelch-oper'

Merge branch 'jc/conf-var-doc' i

Merge branch 'av/wincred-with-at'

Merge branch 'ch/new-gpg-drops-rfc'

Merge branch 'jc/remote-set-url-doc' int

Merge branch 'jk/pack-bitmap' into maint

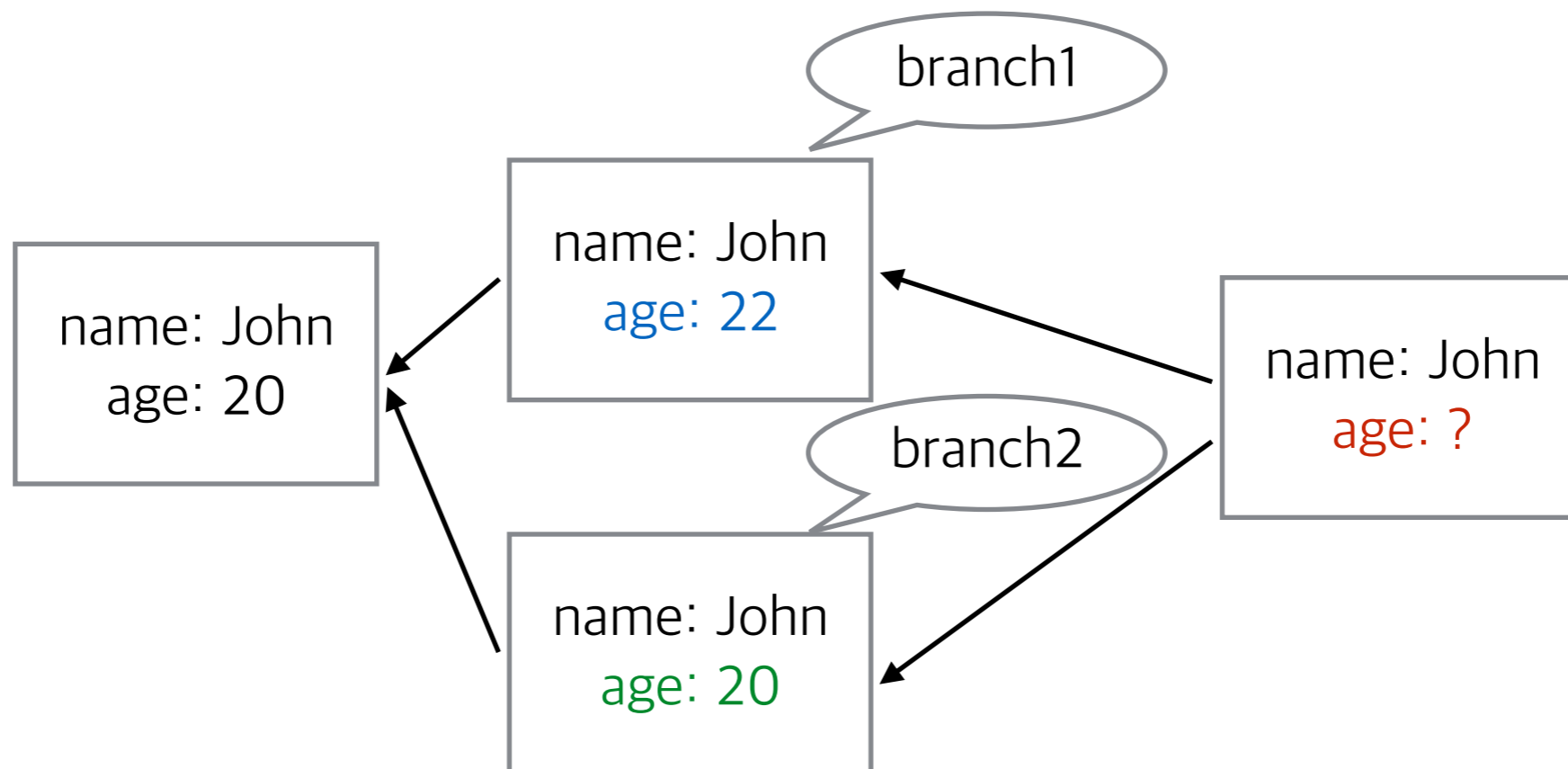
SHA1 ID: d67f9d5e8fd2c165304153a87fd96054d2b74981 Row 1 / 38898

Find commit containing: Exact All fields

Junio C Hamano <gitster@pobox.com>	2015-03-06 23:05:39
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:33
Ben Walton <bdwalton@gmail.com>	2015-03-02 19:22:31
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:32
Anders Kaseorg <andersk@mit.edu>	2015-03-01 04:18:14
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:31
Michal Sojka <sojkam1@fel.cvut.cz>	2015-03-02 22:57:58
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:29
Jean-Noel Avila <jn.avila@free.fr>	2015-03-01 11:58:25
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:28
Mårten Kongstad <marten.kongstad@gmail.com>	2015-03-02 15:05:39
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:27
Michael J Gruber <git@drmicha.warpmail.net>	2015-03-02 13:08:09
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:27
Nguyễn Thái Ngọc Duy <pclouds@gmail.com>	2015-02-27 14:01:58
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:26
Mikko Rapeli <mikko.rapeli@iki.fi>	2015-02-26 13:16:49
Junio C Hamano <gitster@pobox.com>	2015-03-06 23:02:24
Jeff King <peff@peff.net>	2015-02-26 03:04:16
Junio C Hamano <gitster@pobox.com>	2015-03-06 22:59:12
Junio C Hamano <gitster@pobox.com>	2015-03-06 22:58:14
Junio C Hamano <gitster@pobox.com>	2015-03-06 22:57:58
Junio C Hamano <gitster@pobox.com>	2015-03-06 22:57:57
Junio C Hamano <gitster@pobox.com>	2015-03-06 22:57:56
Junio C Hamano <gitster@pobox.com>	2015-03-06 22:57:55
Junio C Hamano <gitster@pobox.com>	2015-03-06 22:57:54
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:16:27
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:15:53
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:13
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:12
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:12
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:10
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:09
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:08
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:07
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:06
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:05
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:04
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:03
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:03
Junio C Hamano <gitster@pobox.com>	2015-03-05 21:13:02

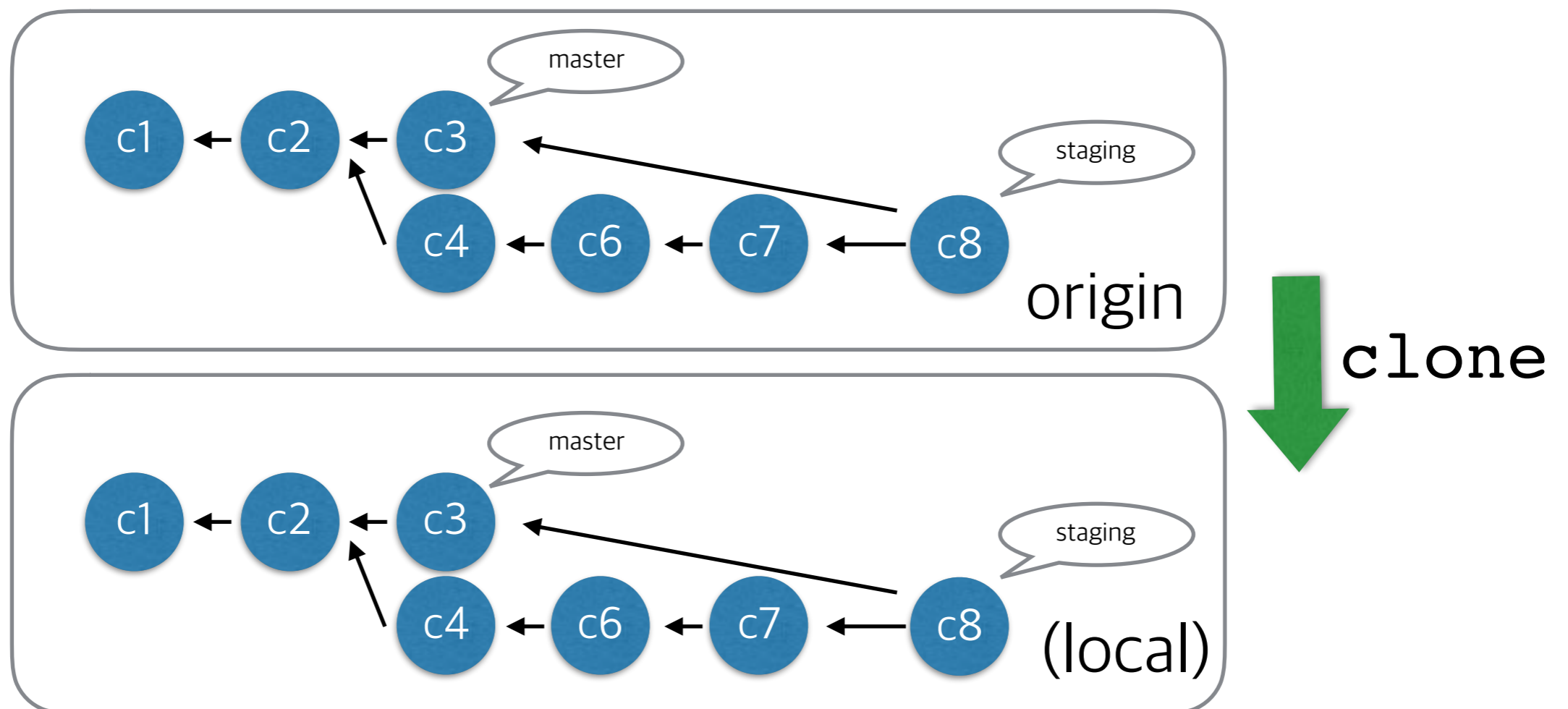
Conflict

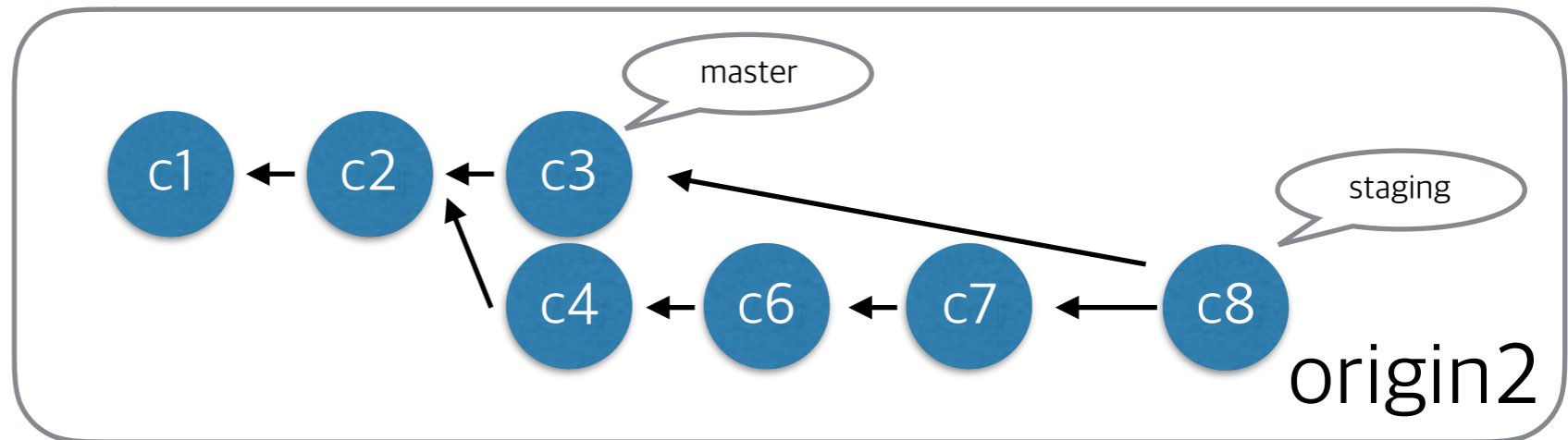
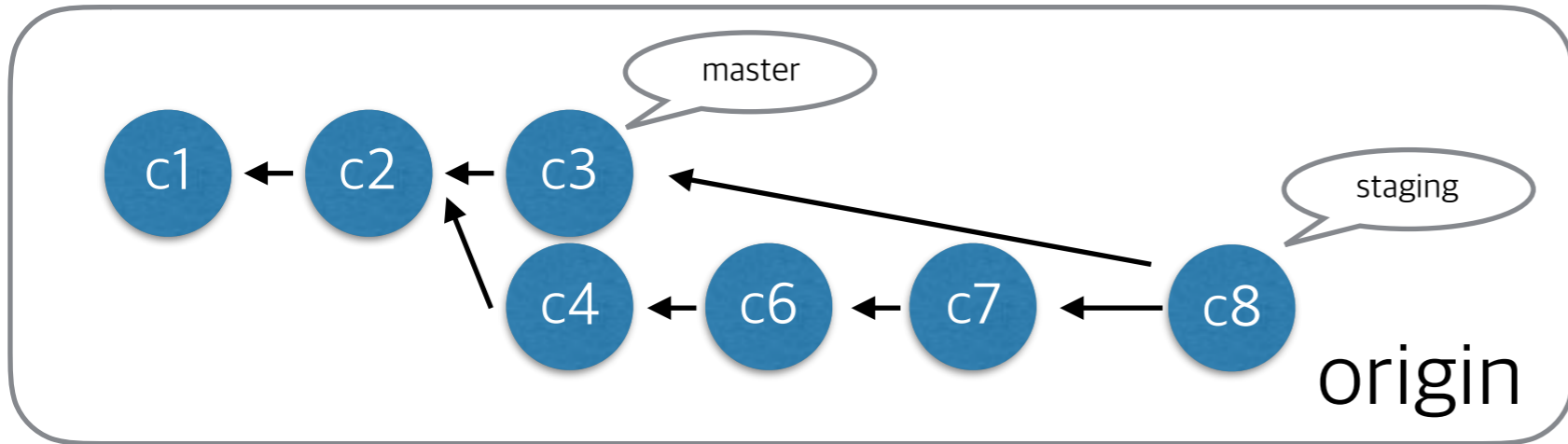
- Don't get scared
- Conflict happens when Git failed to merge
- Usually when two commits changed same part of a file
- Developer should choose which one to follow



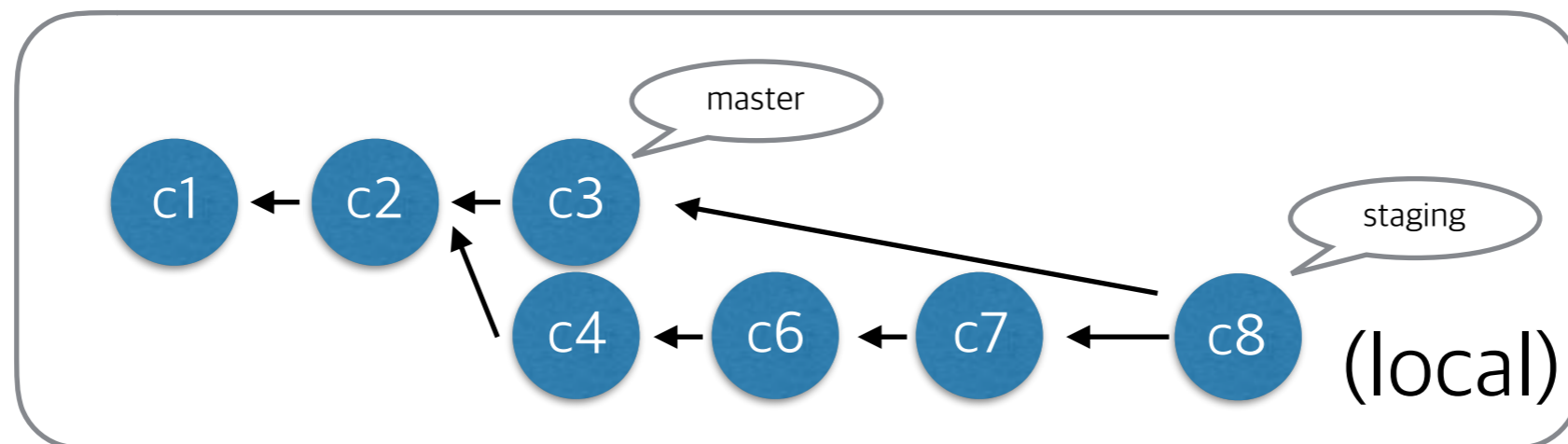
Remote Repository

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



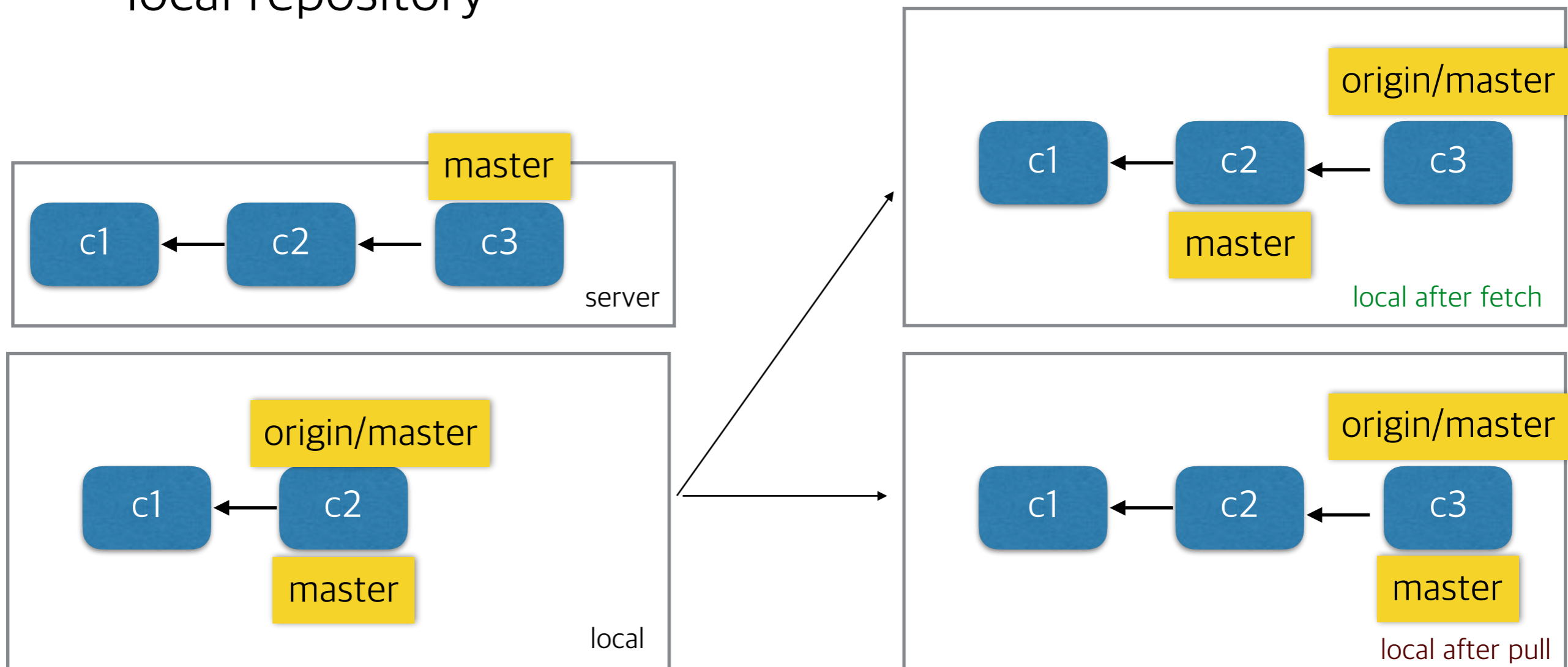


We can have multiple remote repositories.
`origin` is just one of them.



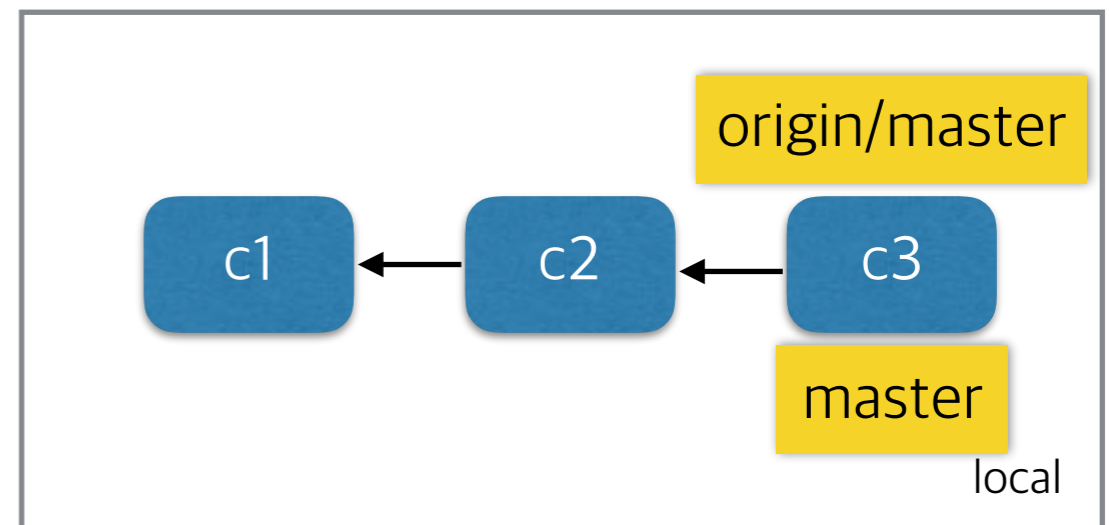
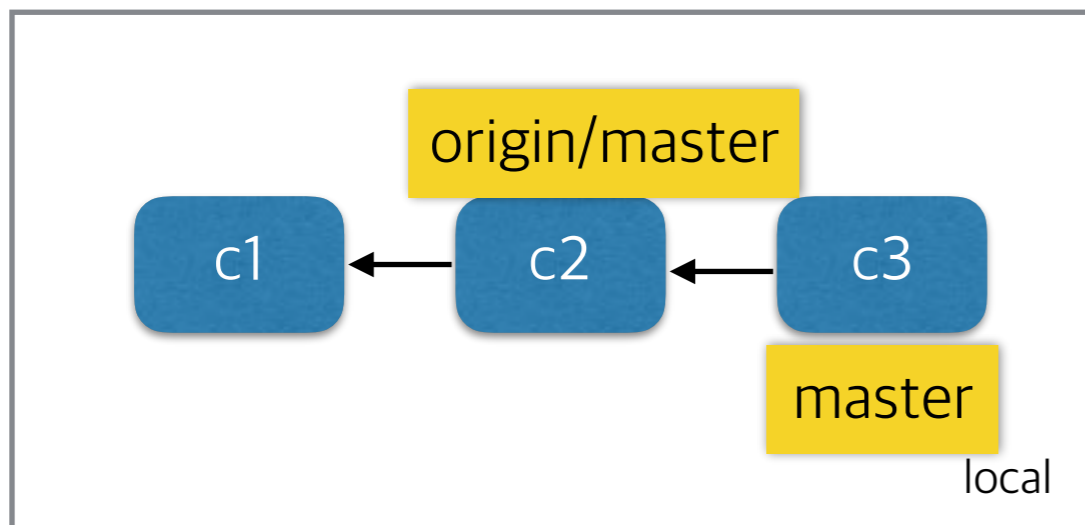
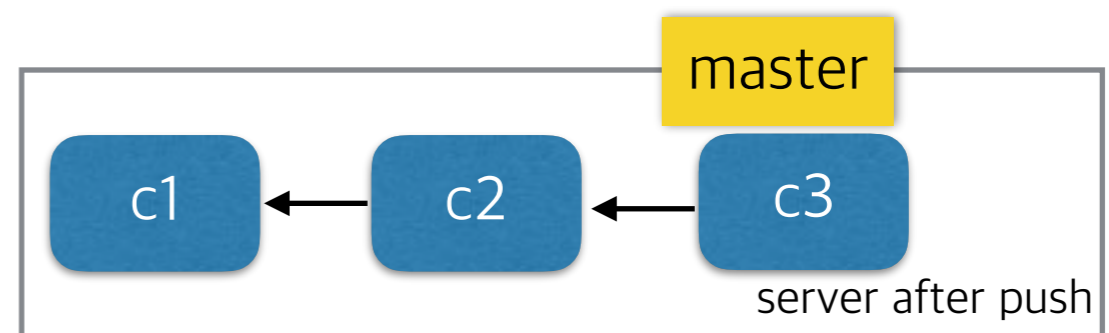
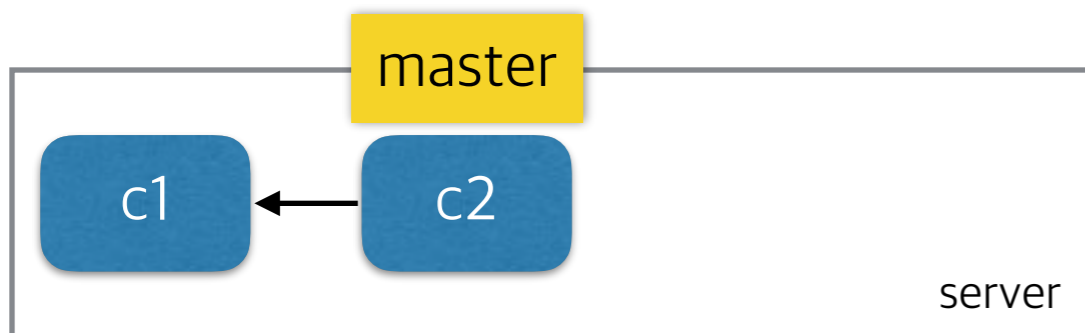
Fetch, Pull

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



Push

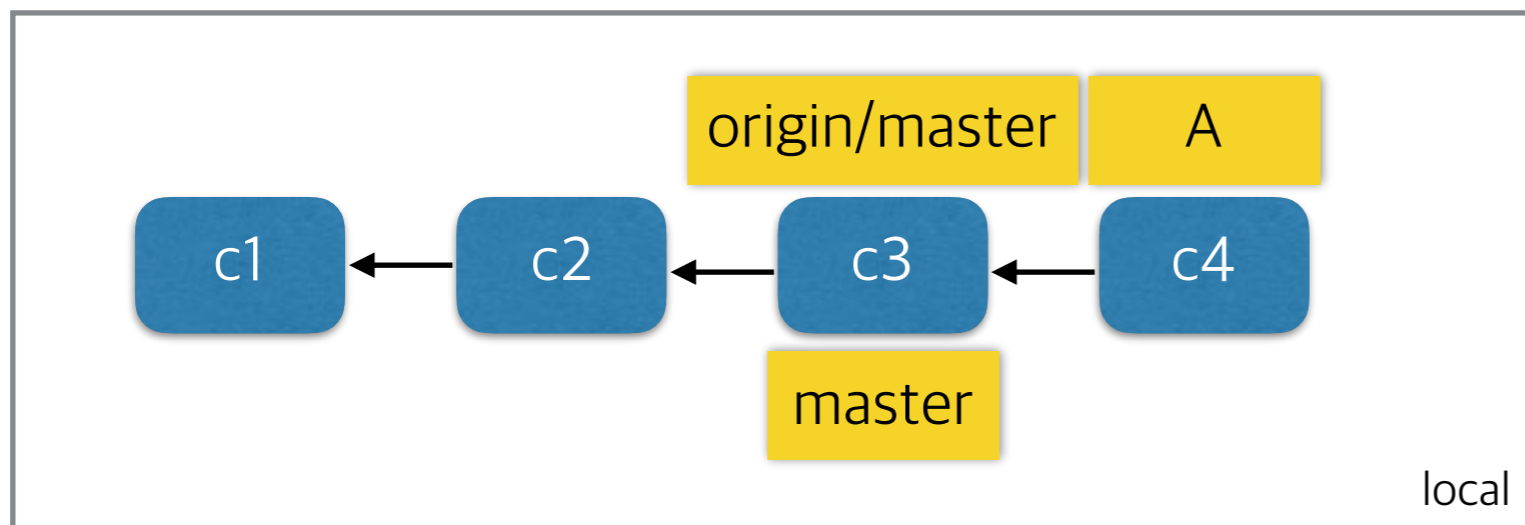
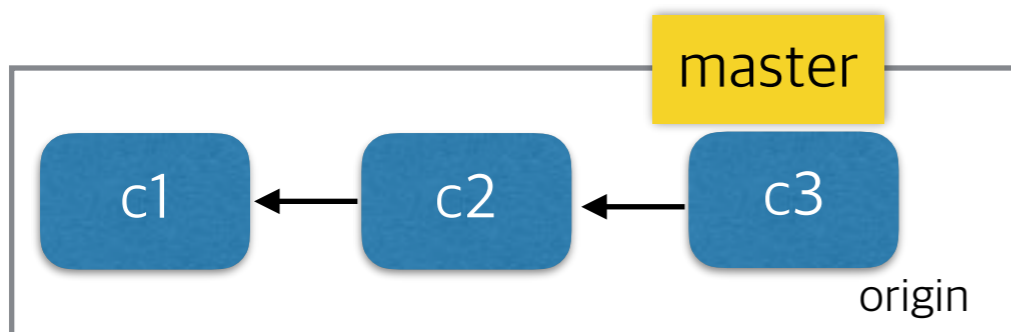
- Apply local commits to remote repo



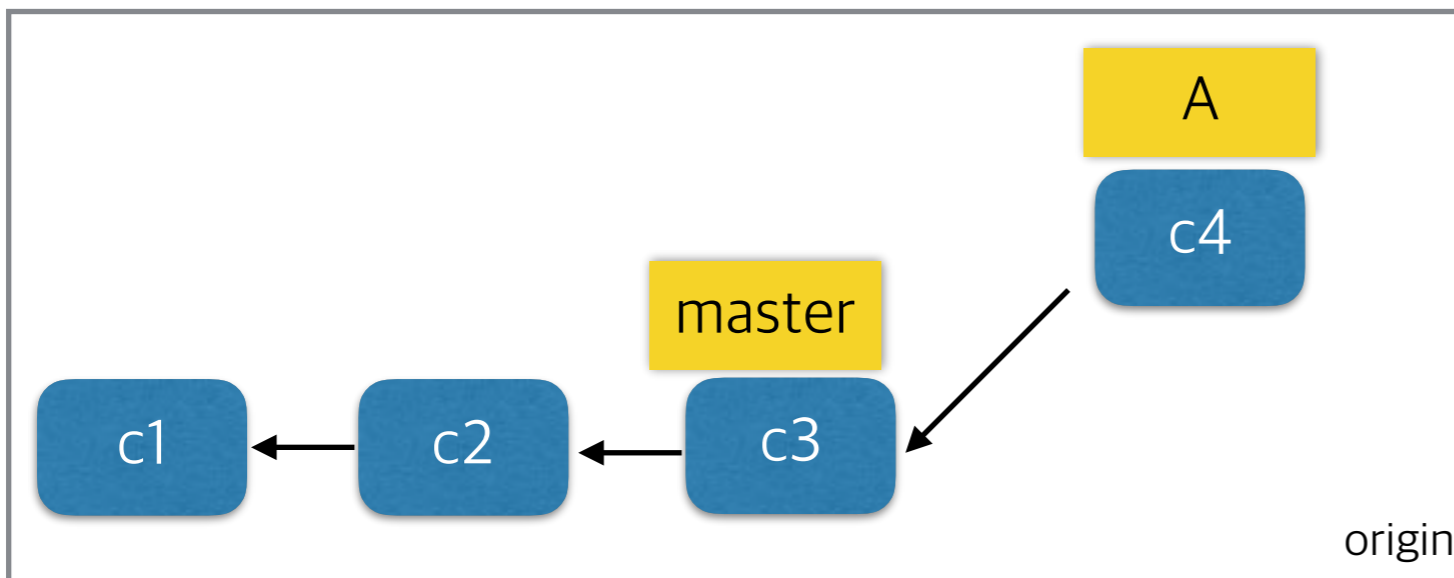
Pull Request

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

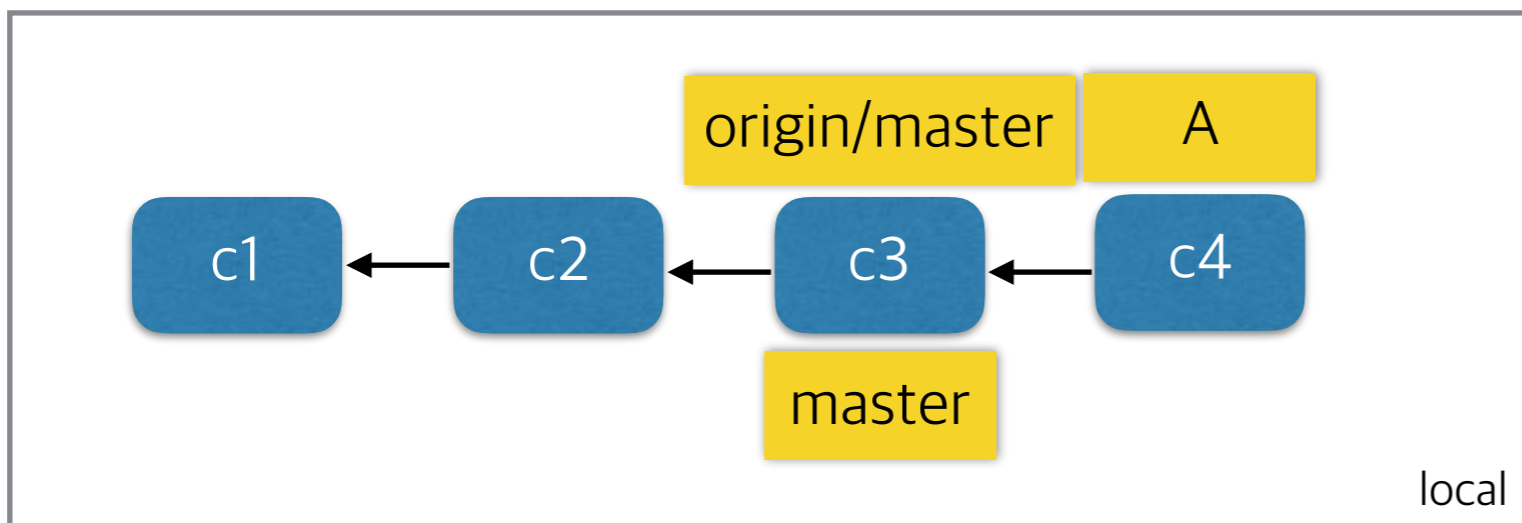
The screenshot shows the GitHub interface for creating a pull request. At the top, the repository is identified as 'octa-org / octa-repo' (Private). It shows 2 watchers, 0 stars, and 1 fork. Below this is a navigation bar with links for Code, Issues (1), Pull requests (4), Projects (0), Pulse, Graphs, and Settings. The main heading is 'Open a pull request', with a subtext: 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).' Below this is a comparison bar showing 'base: update-readme' and 'compare: modifications-to-143v', with a green checkmark and the text 'Able to merge. These branches can be automatically merged.' A yellow banner below that says 'Please review the [guidelines for contributing](#) to this repository.' At the bottom, there is a text input field with the title 'Modifications to 143v' and a 'Reviewers' section on the right that says 'No reviews— request one'.

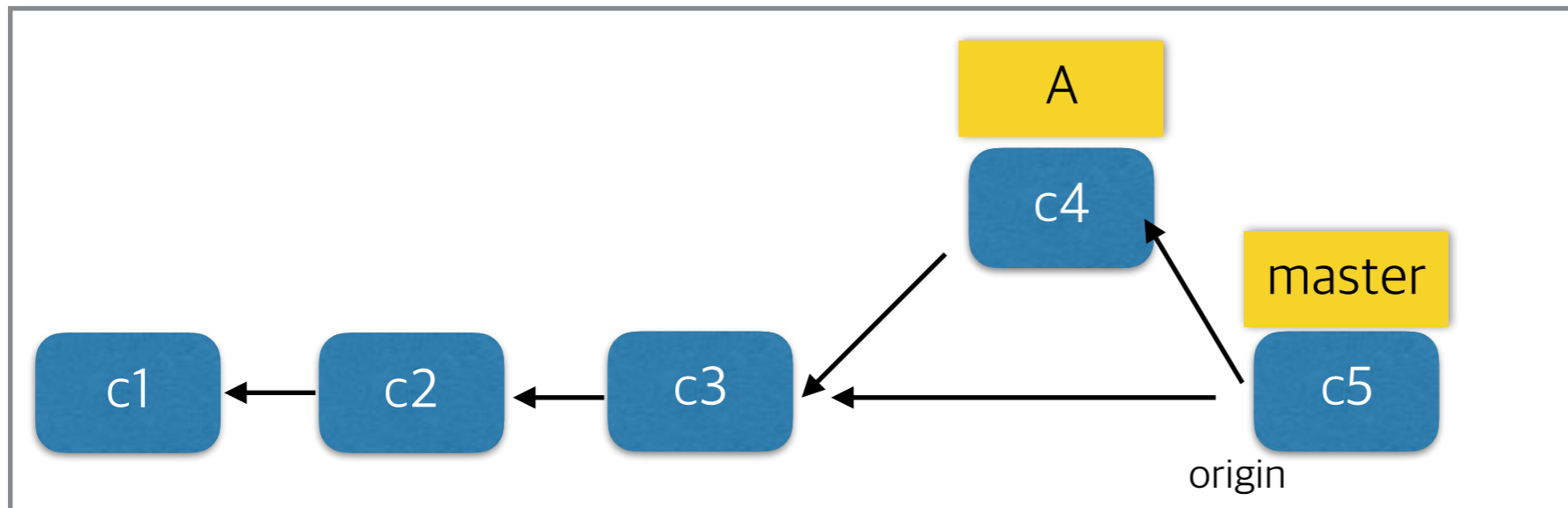


Has a new local branch A,
but cannot push it to master

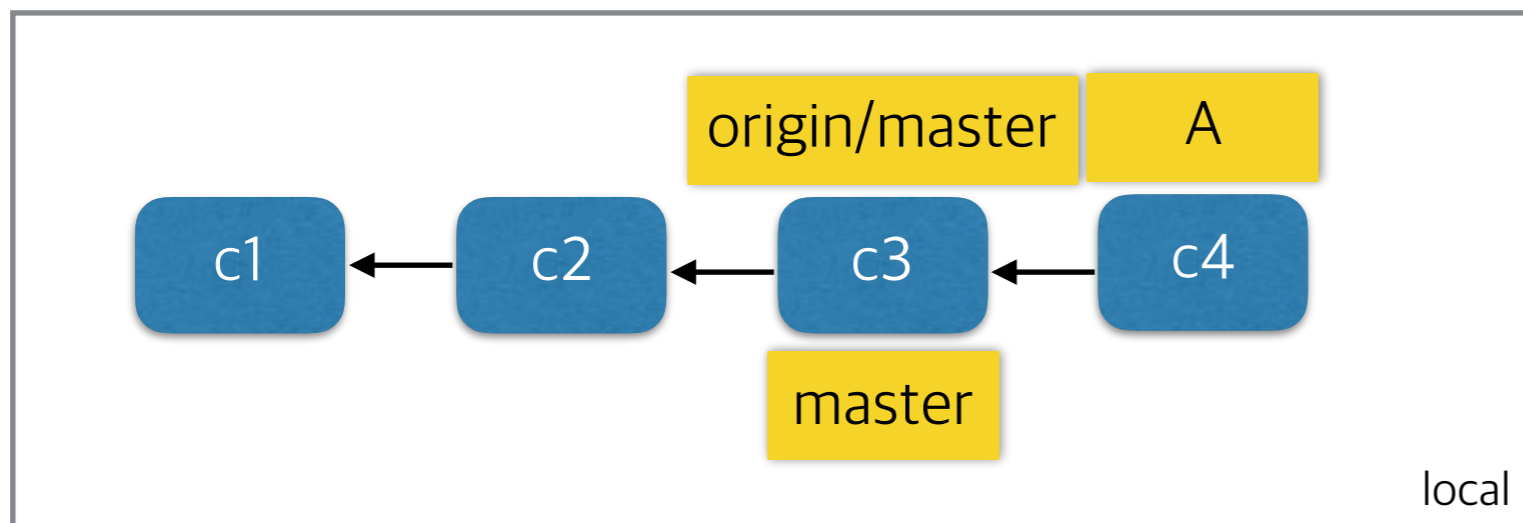


Upload new branch A,
and make **PULL REQUEST**



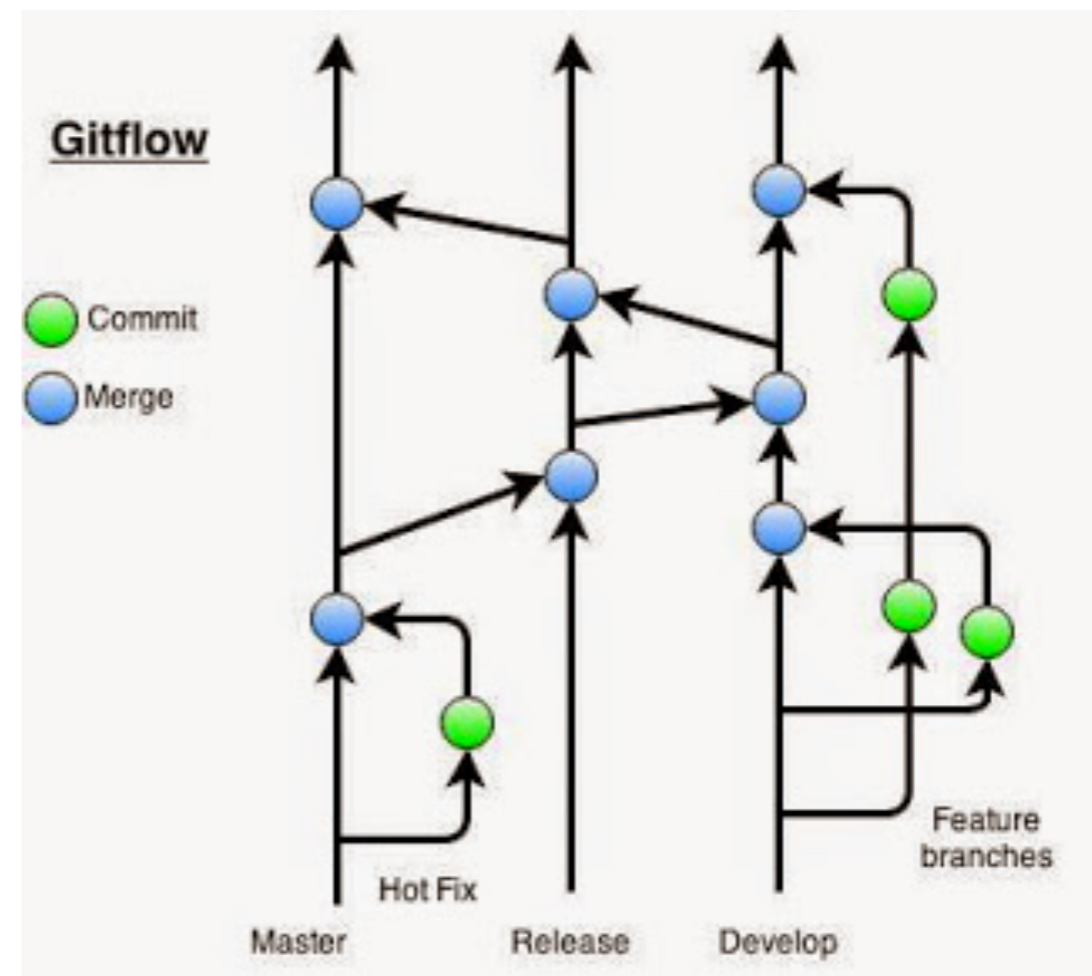
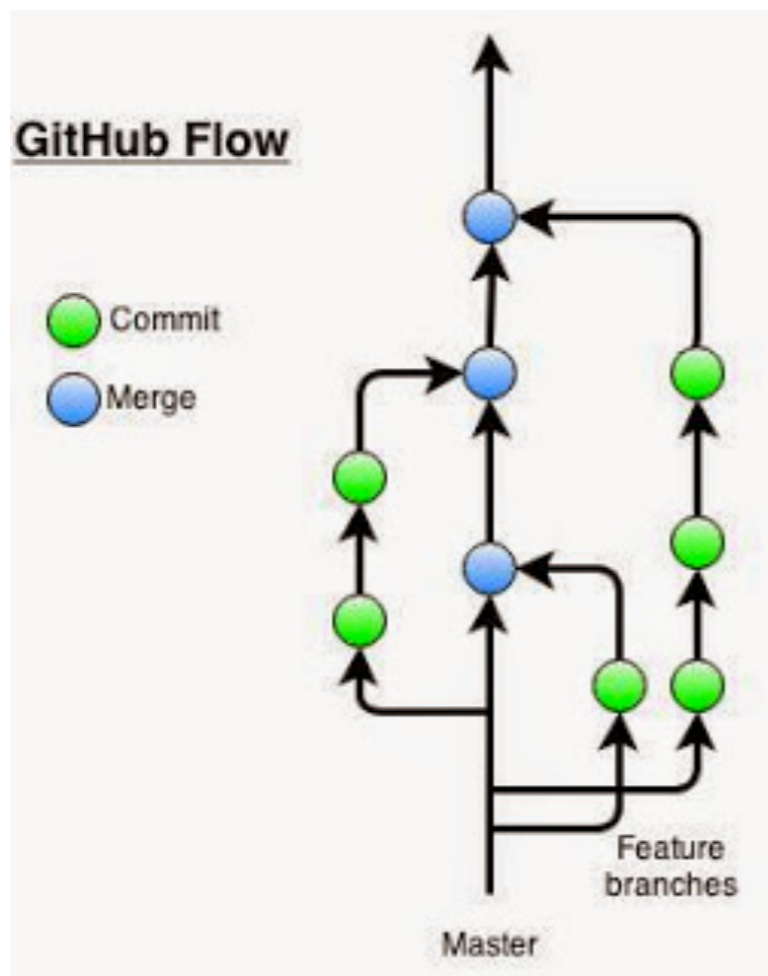


After PR is accepted,
new branch is
merged to master



Working with others

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



Since you are a student



The screenshot shows the GitHub Education website's landing page for the Student Developer Pack. The page has a dark orange header with the GitHub Education logo on the left and navigation links for Stories, Events, Student pack (underlined), Classroom, Community, and Contact us on the right. A 'Request a discount' button is also present. The main heading is 'Student Developer Pack' with the subtitle 'The best developer tools, free for students'. Below this is a white content area featuring a yellow backpack with the GitHub logo on the left. To the right of the backpack, the text reads 'Learn to ship software like a pro' with 'Like 41K' and 'Tweet' buttons. A paragraph explains that the pack provides free access to developer tools for students. A blue 'Get your pack' button is centered below the text. At the bottom of the white area, the text 'THE TOOLS' is partially visible.

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)