

Git 101

Scott Chacon

Me

Scott Chacon



github
SOCIAL CODE HOSTING

github.com/schacon

github-gem

github-perlbal

github-services

gitlondon

gitpresent

ruby-git

ruby-github

ticgit

whygitisbetter

gittrack

gitready

gitrules

gitscm

gitx

grit

groundcontrol

hg-git

igithub

apressgit

asgit

cocoagit

dotgit

egit

gist

git-cheetah

git-lighthouse

git-phone

git-presentations

git-ruby

git-server

git-source

git-sphinx

git-wiki

gitbook

gitbrowser

githooks

github

git-scm.com

[Home](#)[About Git](#)[Documentation](#)[Download](#)[Related Tools](#)[Wiki](#)

The [Git User's Survey 2008](#) is up! Please devote a few minutes of your time to fill it out, so we can improve Git!

Git is...

Git is an **open source, distributed version control system** designed to handle everything from small to very large projects with speed and efficiency.

Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server.
Branching and merging are fast and easy to do.

Git is used for version control of files, much like tools such as [Mercurial](#), [Subversion](#), [CVS](#), [Perforce](#), [Bitkeeper](#), and [Visual SourceSafe](#).

Projects using Git

- [Git](#)
- [Linux Kernel](#)
- [Ruby on Rails](#)
- [WINE](#)
- [Fedora](#)
- [X.org](#)
- [Rubinius](#)
- [VLC](#)
- [Prototype](#)

Download Git

The latest stable Git release is

v1.6.0.2

[release notes](#) (2008-09-12)



[tar.bz2 \(sign\)](#)



[tar.gz \(sign\)](#)

[Other Download Options](#)

[Source and History](#)

Git Quick Start

Cloning and Creating a Patch

```
$ git clone git://github.com/git/hello-world.git  
$ cd hello-world
```

Creating and Committing

```
$ cd (project-directory)  
$ git init
```

Got a Question?

If you are curious about something, feel free to ask on the [IRC](#) channel or the [mailing list](#).

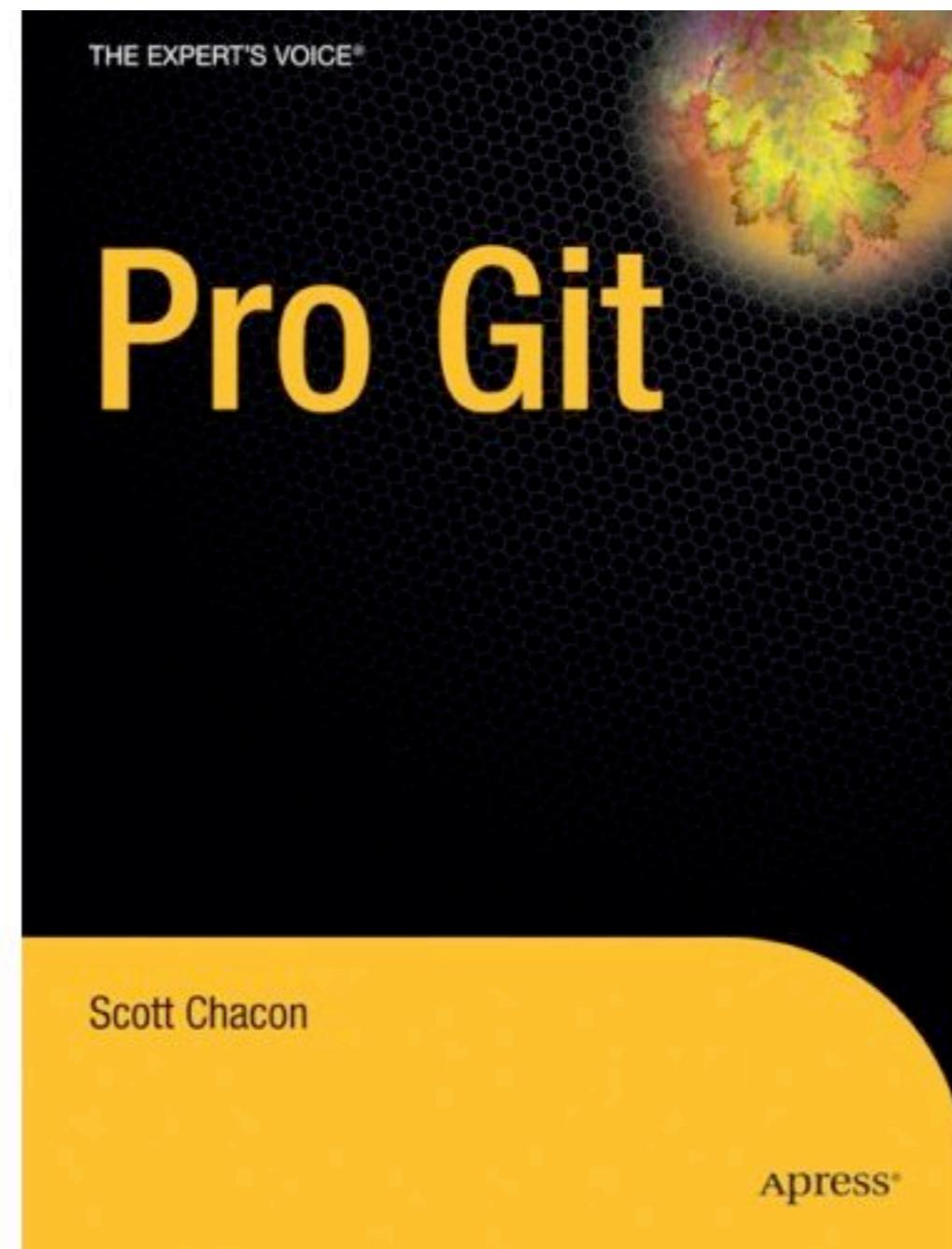
PeepCode
press

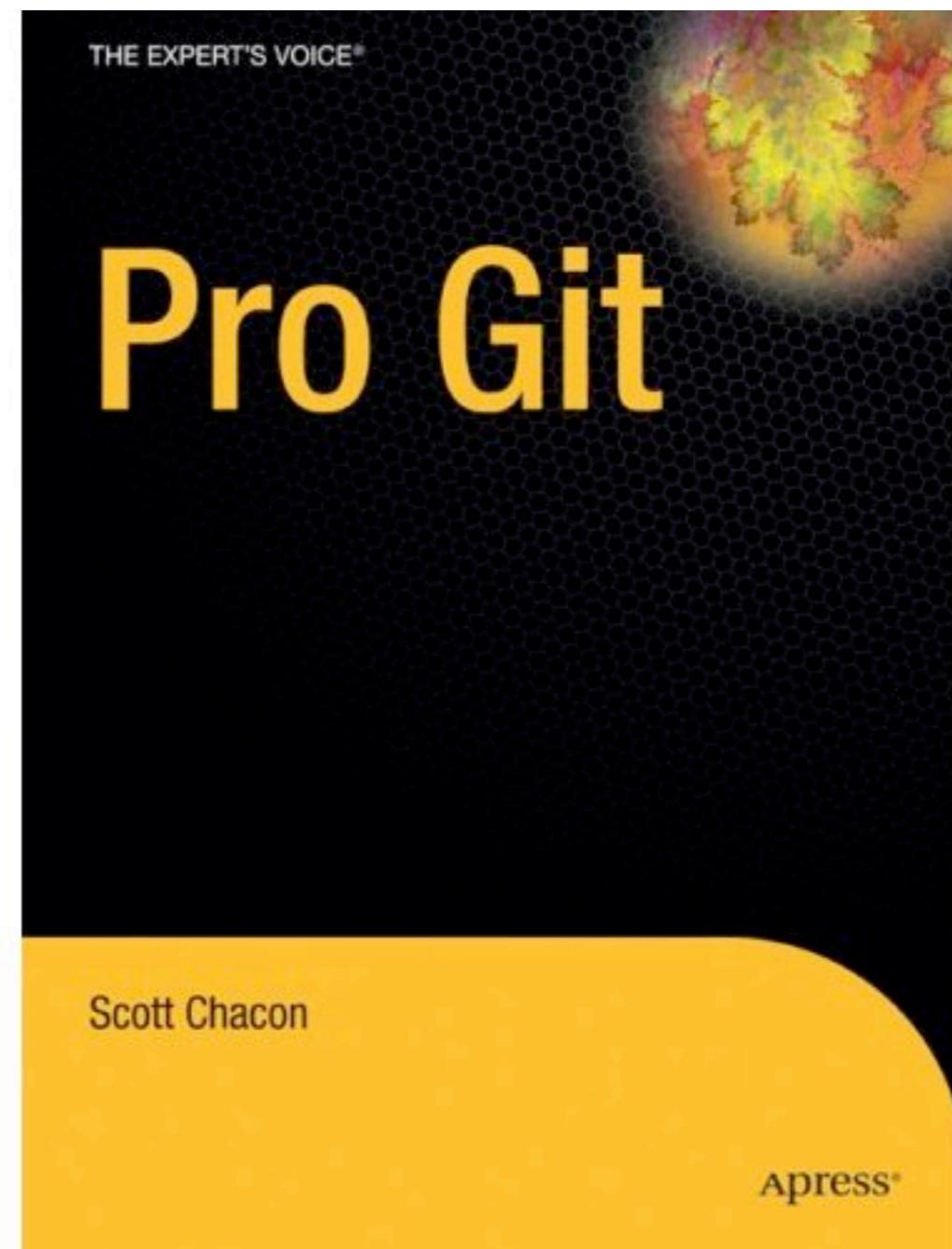
\$9

Git Internals

Source code control and beyond

by Scott Chacon





<http://progit.org>

book.git-scm.com



The Git Community Book

Home About Git Documentation Download Related Tools Wiki

The Git Community Book

Welcome to the Git Community Book. This book has been built by dozens of people in the Git community, and is meant to help you learn how to use Git as quickly and easily as possible.

If you see anything out of date, have a suggestion on how to improve it, or would like to help add to the book, please see the [How to Contribute](#) page, or just send [our maintainer](#) a note.

Download

Click [here](#) to download the current PDF version of this book.



New to Git?

If you want to really understand Git, You may want to start at [the beginning](#).

If you just want to jump right in, you can skip right to [setting it up](#).

1. Introduction

[Welcome to Git](#)

[The Git Object Model](#)

[Git Directory and Working Directory](#)

[The Git Index](#)

2. First Time

[Installing Git](#)

5. Advanced Git

[Creating New Empty Branches](#)

[Modifying your History](#)

[Advanced Branching And Merging](#)

[Finding Issues - Git Bisect](#)

[Finding Issues - Git Blame](#)

[Git and Email](#)

[Customizing Git](#)

schacon@gmail.com

@chacon

twitter

</me>

What is Git?

**Git is an open source,
distributed version control
system designed for speed
and efficiency**

**Git is an open source,
distributed version control
system designed for speed and
efficiency**

version control

mercurial (hg)

bazaar

subversion (svn)

version control

concurrent version system (cvs)

perforce

visual source safe

SoS

mercurial (hg)

bazaar

subversion (svn)

version control

concurrent version system (cvs)

perforce

visual source safe



SoS



mercurial (hg)

“not bad”

bazaar

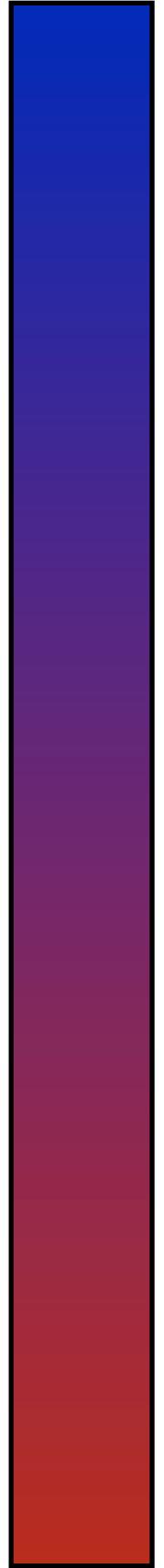
subversion (svn)

version control

concurrent version system (cvs)

perforce

visual source safe



SoS

mercurial (hg)

bazaar

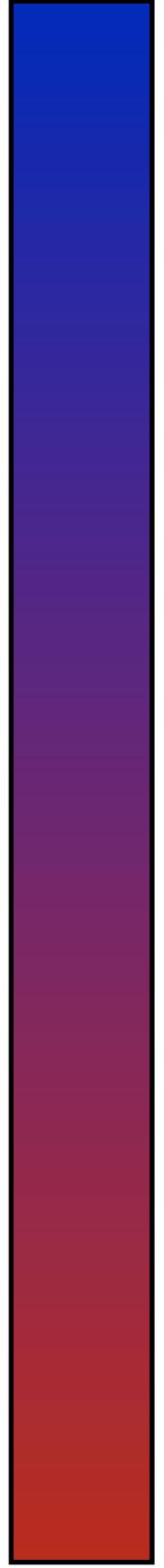
subversion (svn)

version control

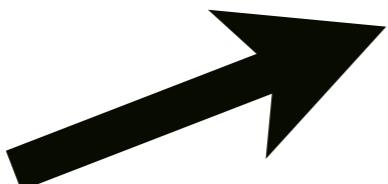
concurrent version system (cvs)

perforce

visual source safe “kill self”



git



mercurial (hg)

bazaar

subversion (svn)

version control

concurrent version system (cvs)

perforce

visual source safe

**Git is an open source,
distributed version control
system designed for speed
and efficiency**

git-scm.com

The screenshot shows the official website for Git (git-scm.com). The header features the word "git" in large letters with a small icon above it, followed by the tagline "the fast version control system". Below the header is a navigation bar with links for Home, About Git, Documentation, Download, Tools & Hosting, and Wiki. The main content area includes a section titled "Git is..." with text about Git's features and history, a list of projects using Git, and a download section for the latest stable release (v1.6.1.1). A pink circle highlights the "tar.bz2 (sign)" download link.

git the fast version control system

[Home](#) [About Git](#) [Documentation](#) [Download](#) [Tools & Hosting](#) [Wiki](#)

Git is...

Git is an **open source, distributed version control system** designed to handle everything from small to very large projects with speed and efficiency.

Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server.

Branching and merging are fast and easy to do.

Git is used for version control of files, much like tools such as [Mercurial](#), [Bazaar](#), [Subversion](#), [CVS](#), [Perforce](#), and [Visual SourceSafe](#).

Projects using Git

- [Git](#)
- [Linux Kernel](#)
- [Perl](#)
- [Ruby on Rails](#)
- [Android](#)
- [WINE](#)
- [Fedora](#)
- [X.org](#)
- [VLC](#)
- [Prototype](#)

Download Git

The latest stable Git release is **v1.6.1.1**

[Release notes \(2009-01-25\)](#)

[tar.bz2 \(sign\)](#) [tar.gz \(sign\)](#)

[Other Download Options](#) [Source and History](#)

Git Quick Start

Git is an open source,
distributed version control
system designed for speed
and efficiency

Fully Distributed

(almost) everything is local

which means

which means

everything is fast

which means

everything is fast

every clone is a backup

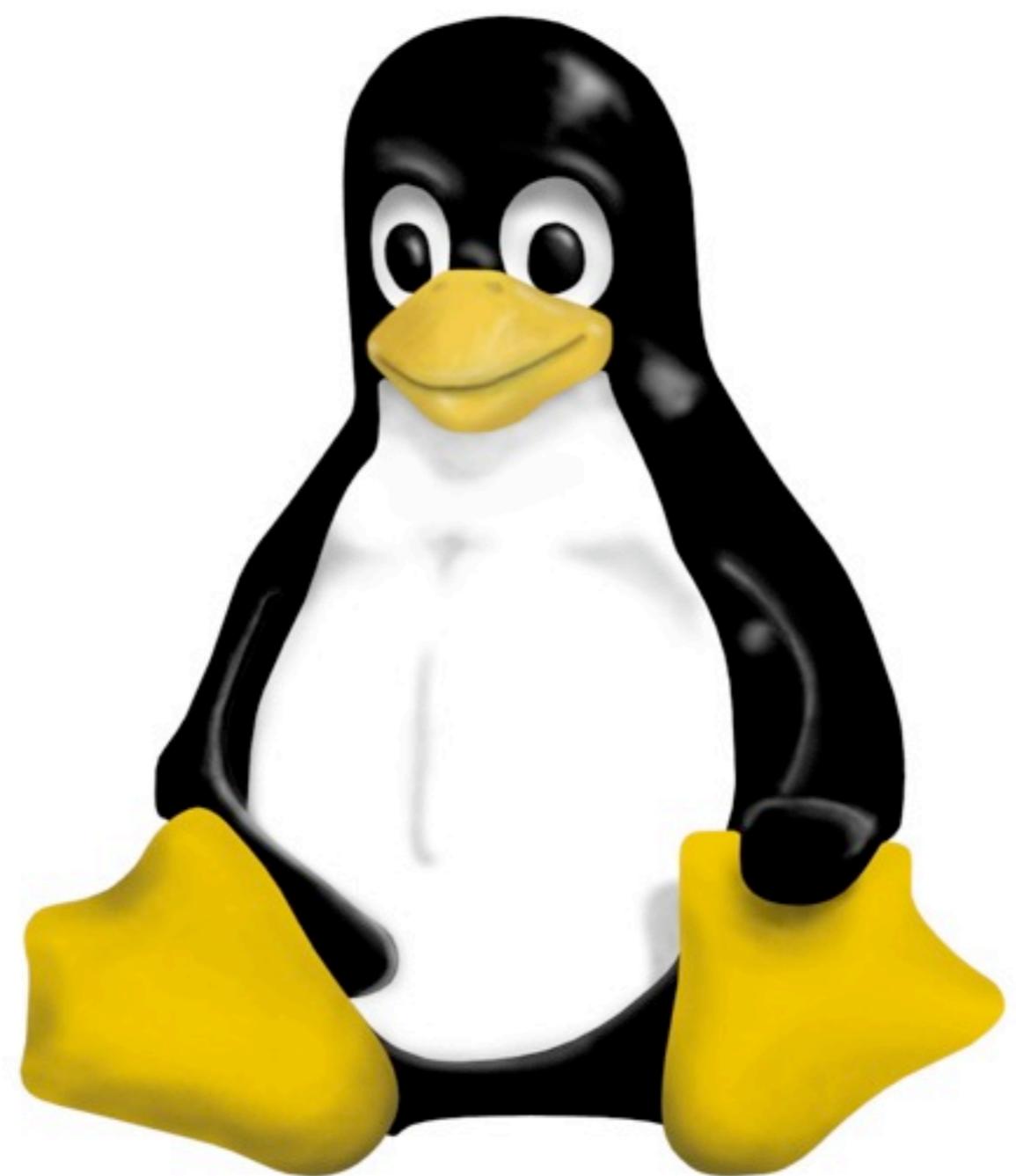
which means

everything is fast

every clone is a backup

work offline

**Git is an open source,
distributed version control
system designed for
speed and efficiency**

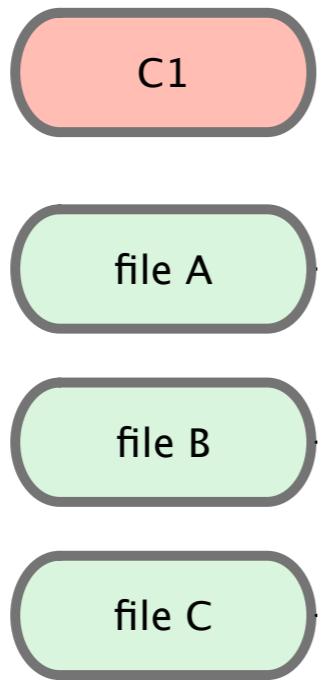
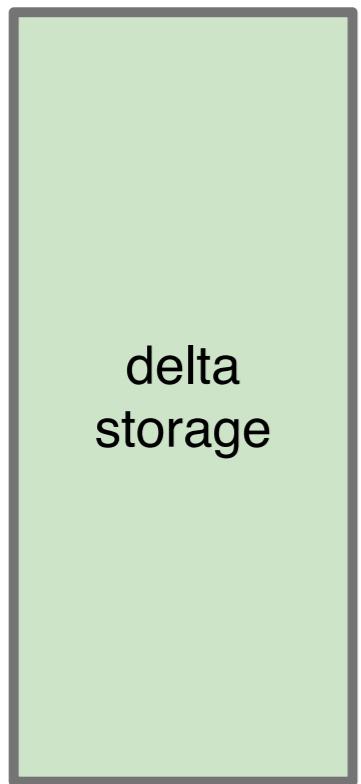


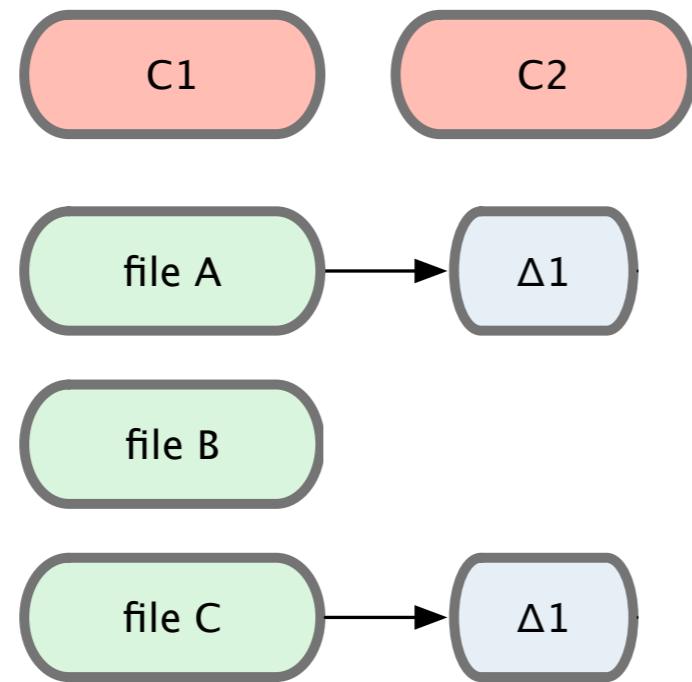
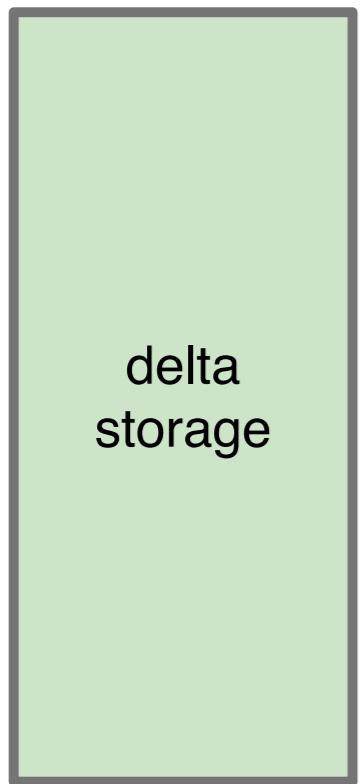
Immutable

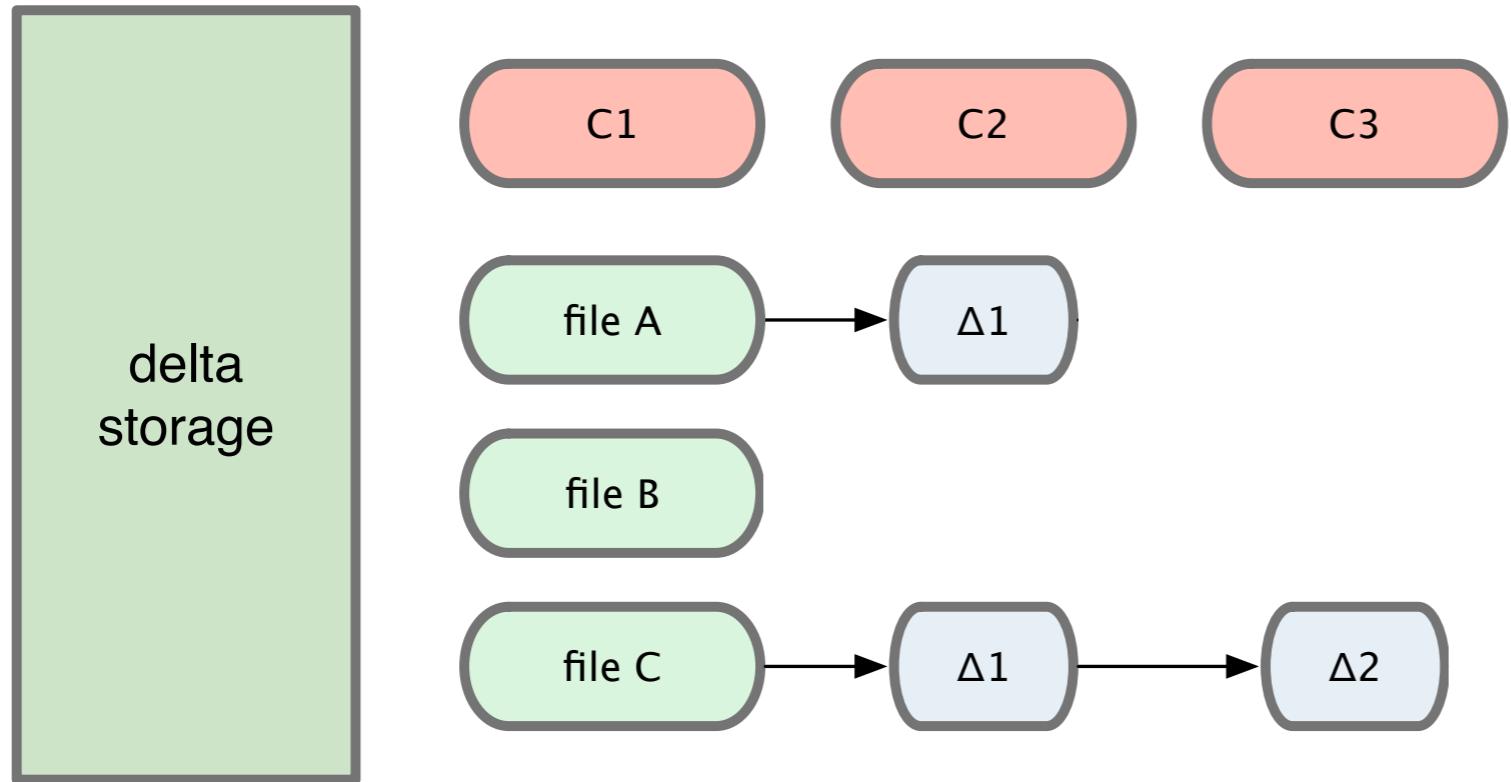
(almost) never removes data

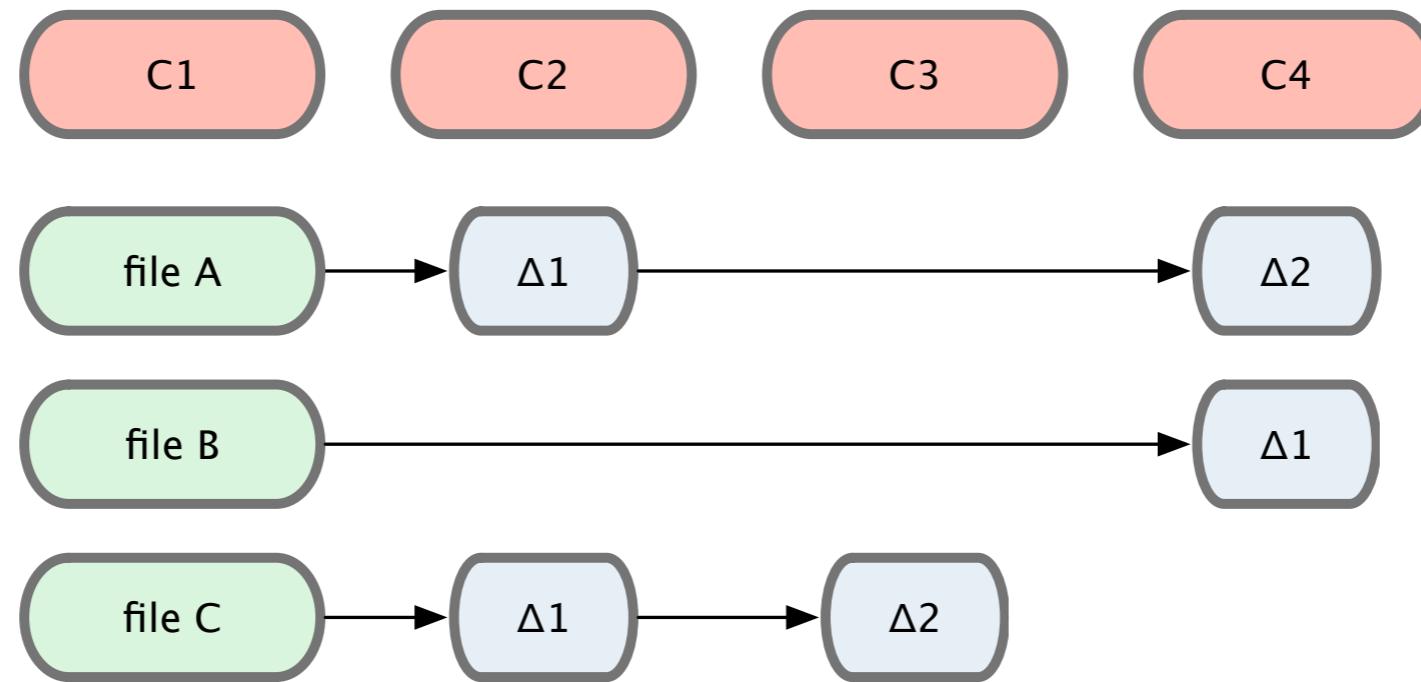
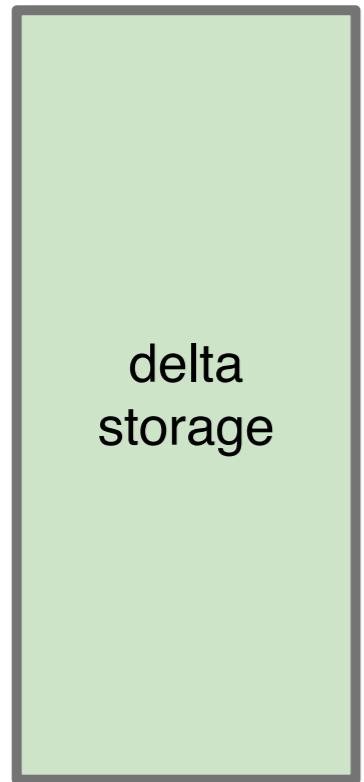
Snapshots, not Patches

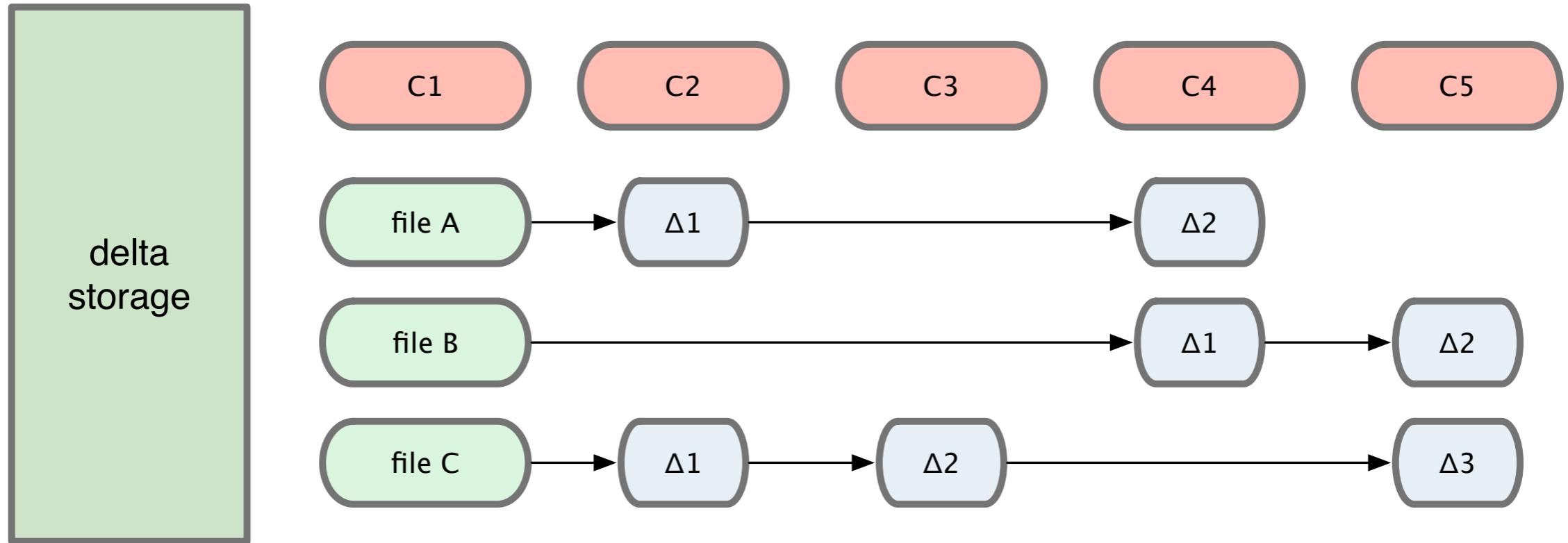
delta
storage

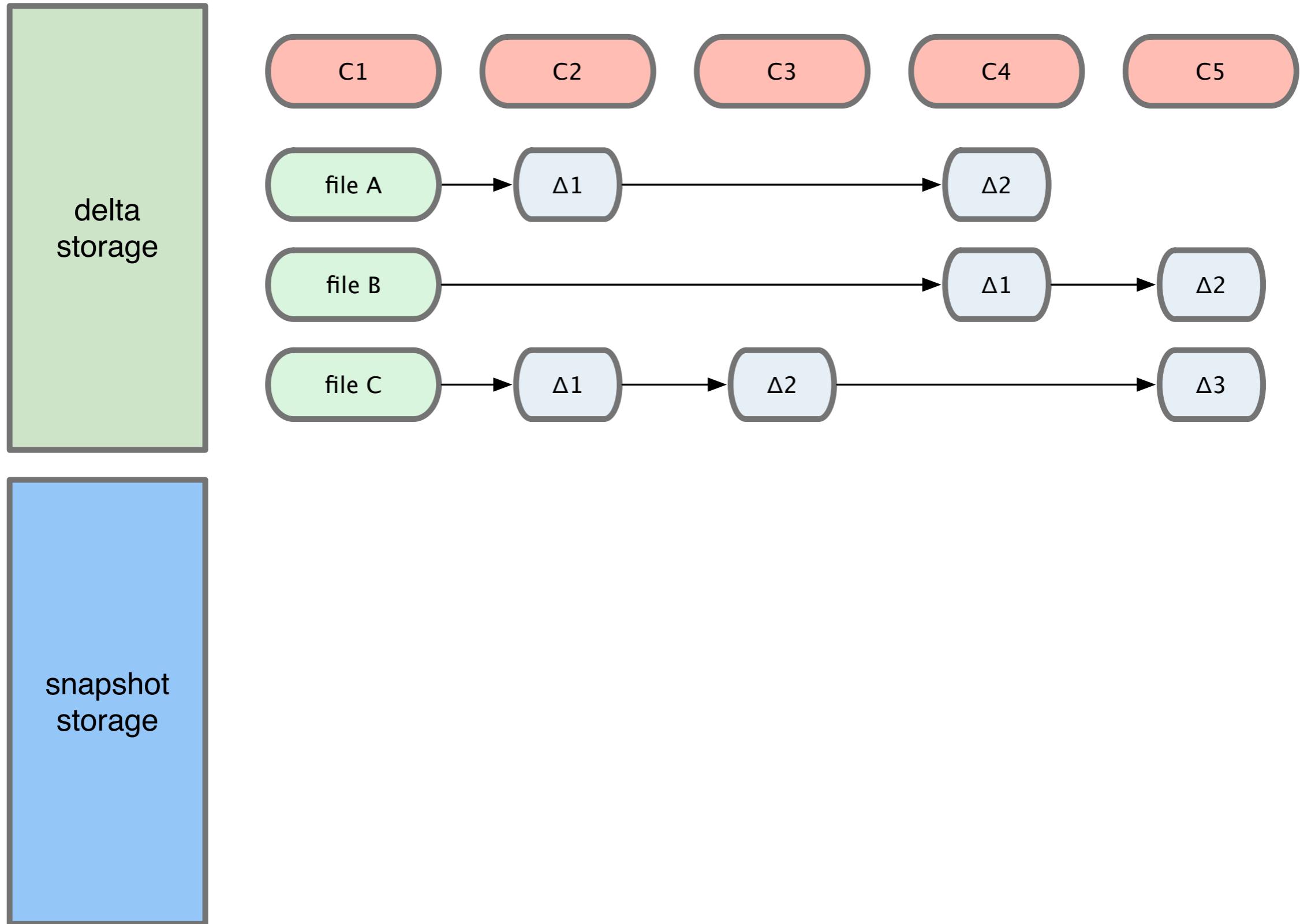


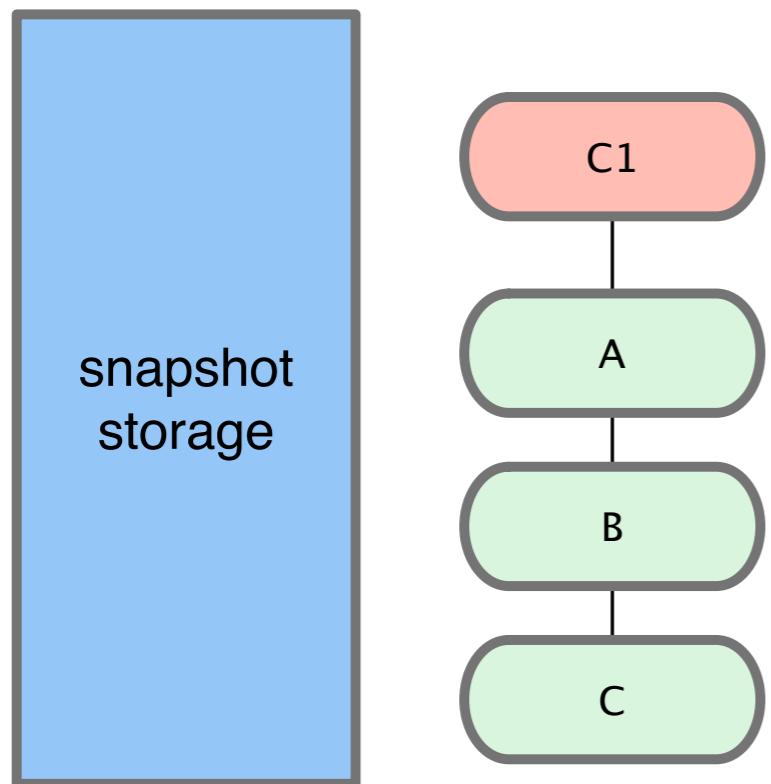
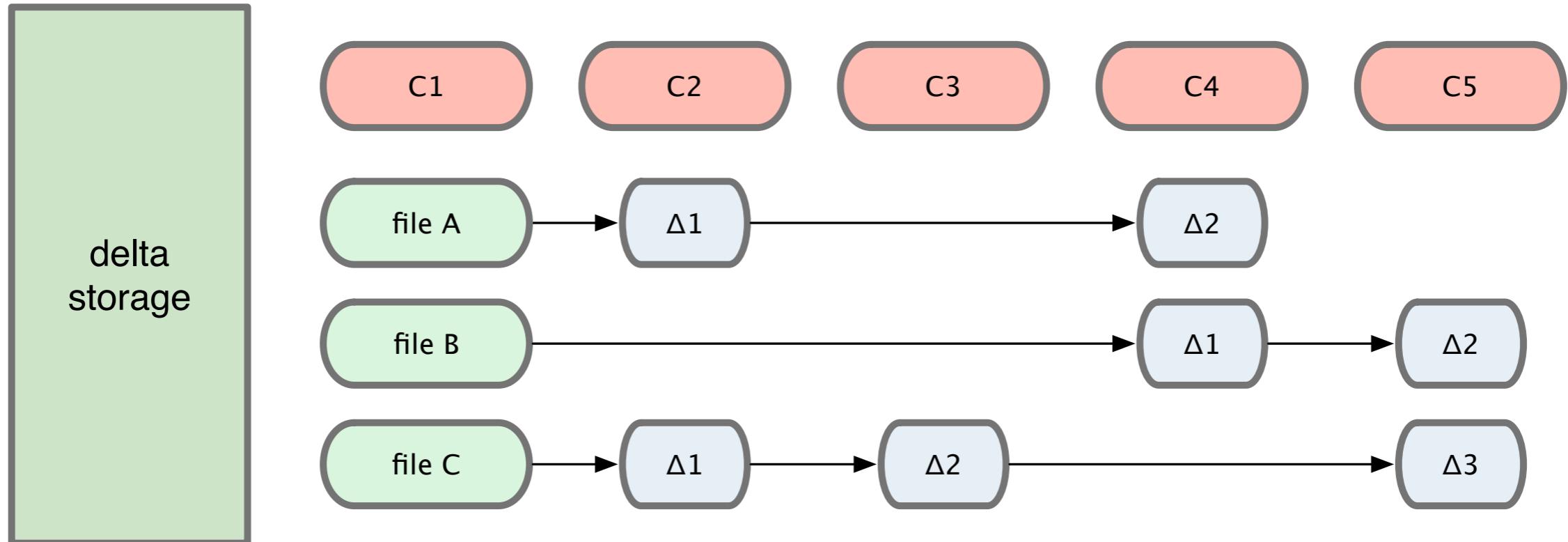


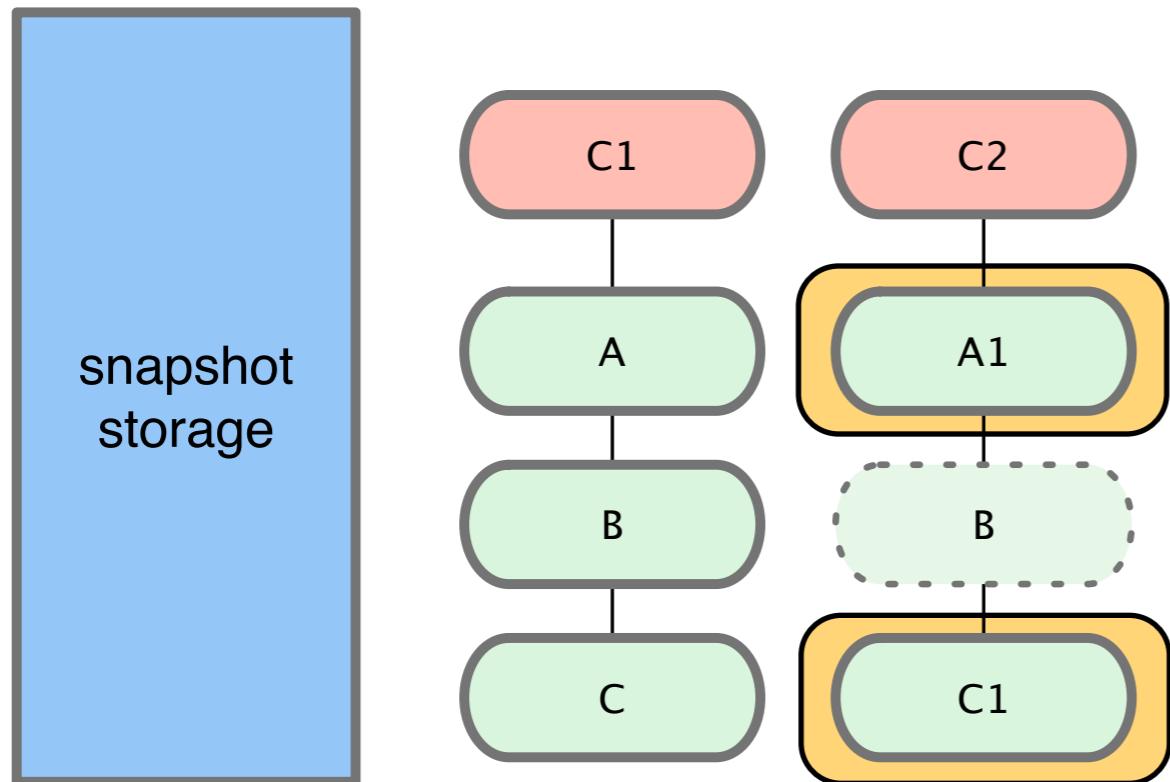
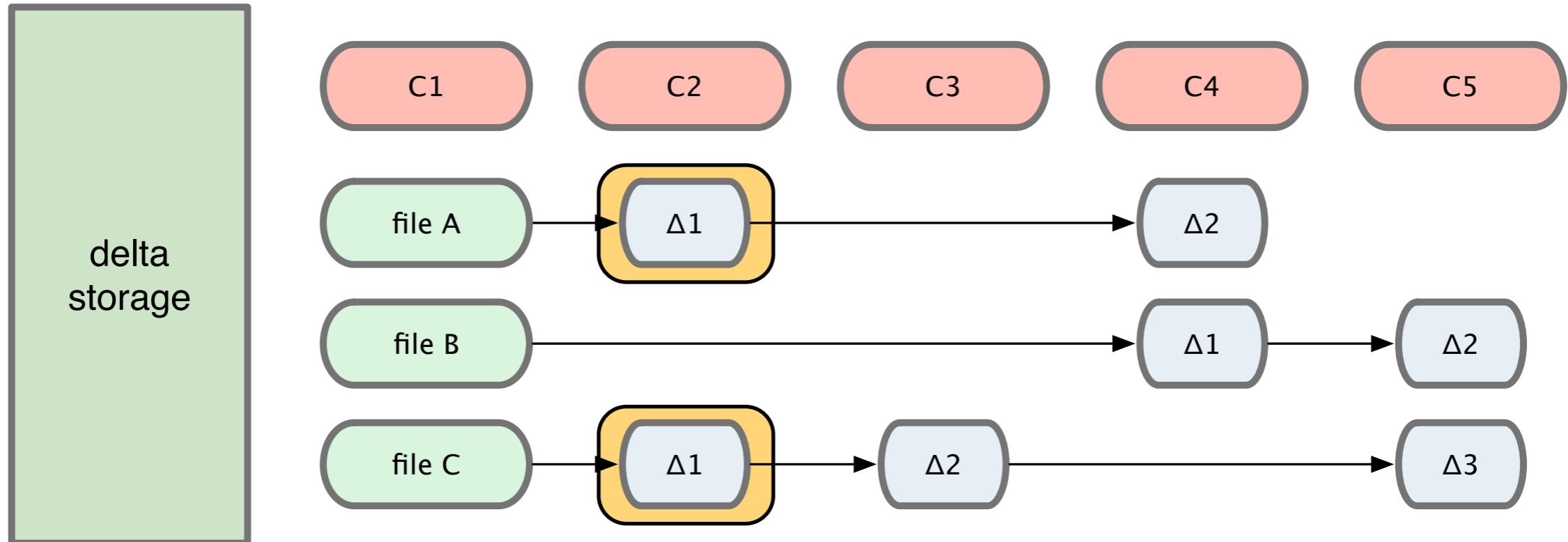


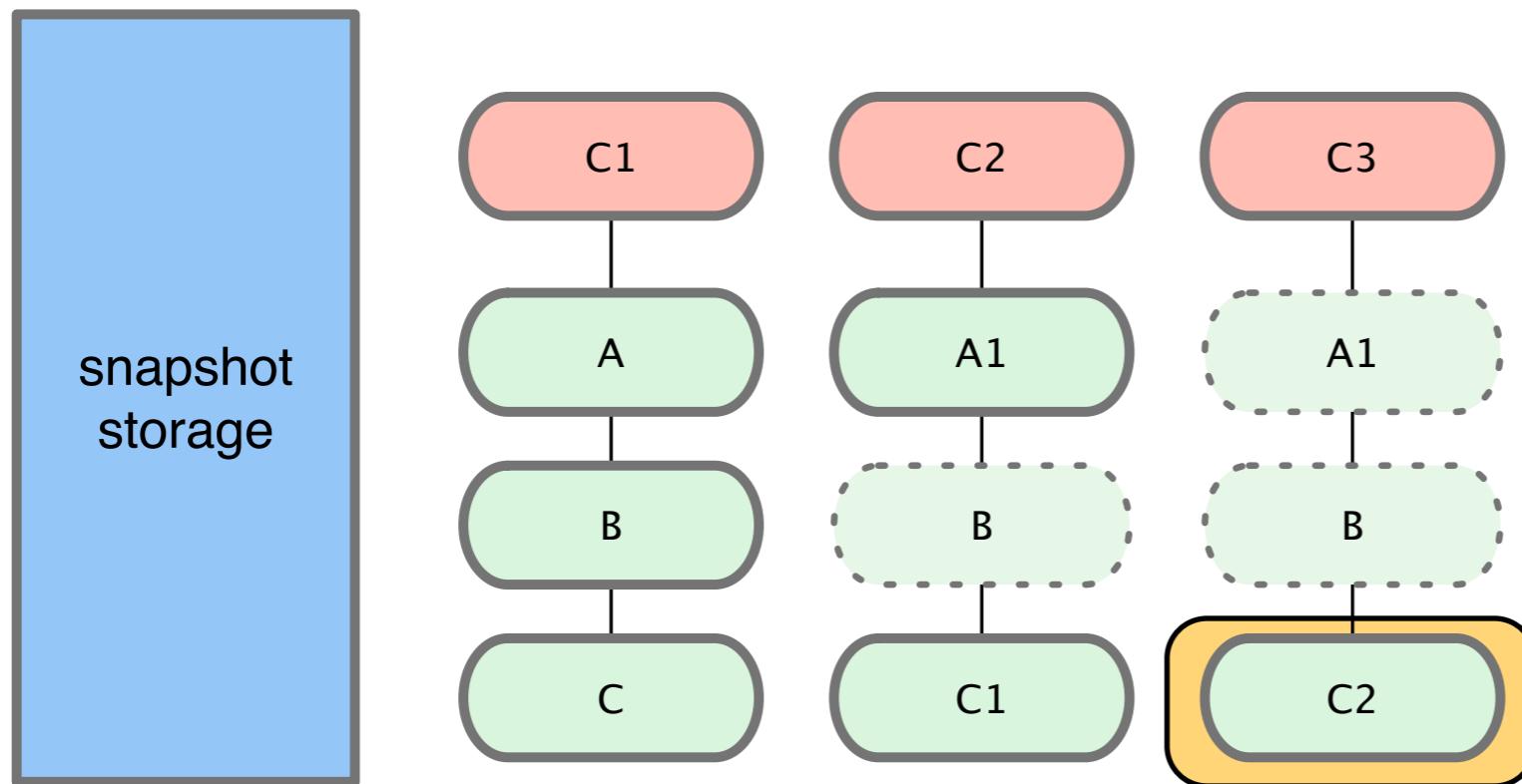
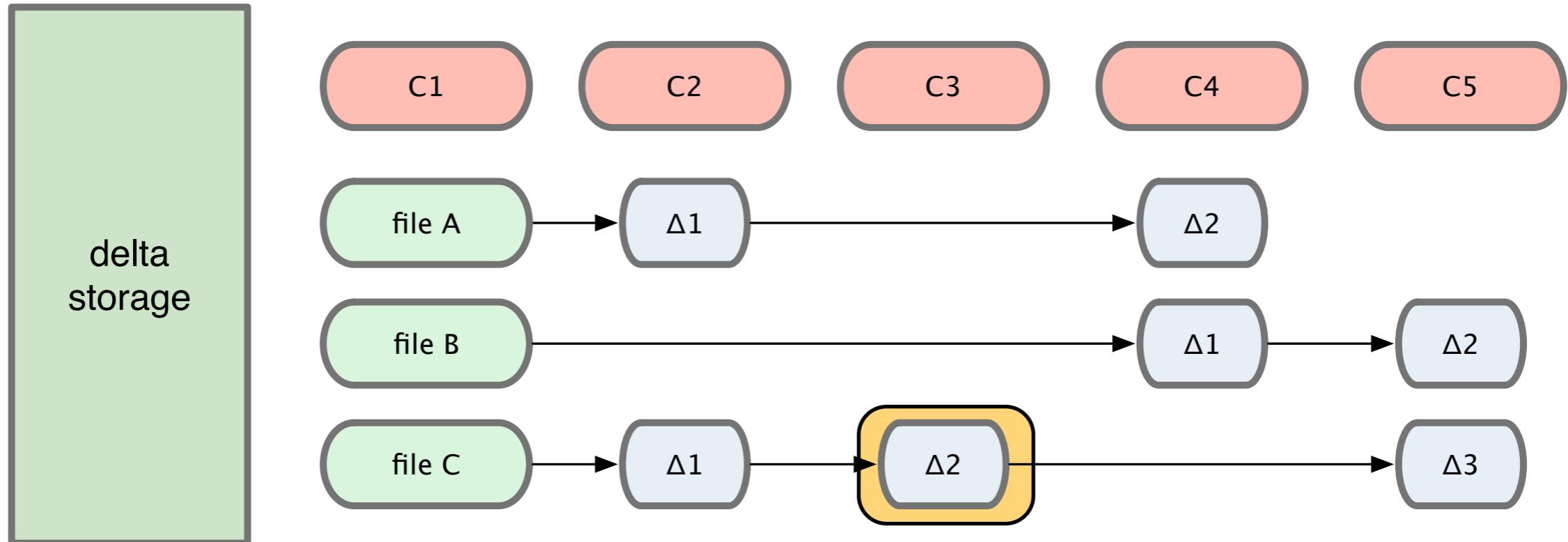


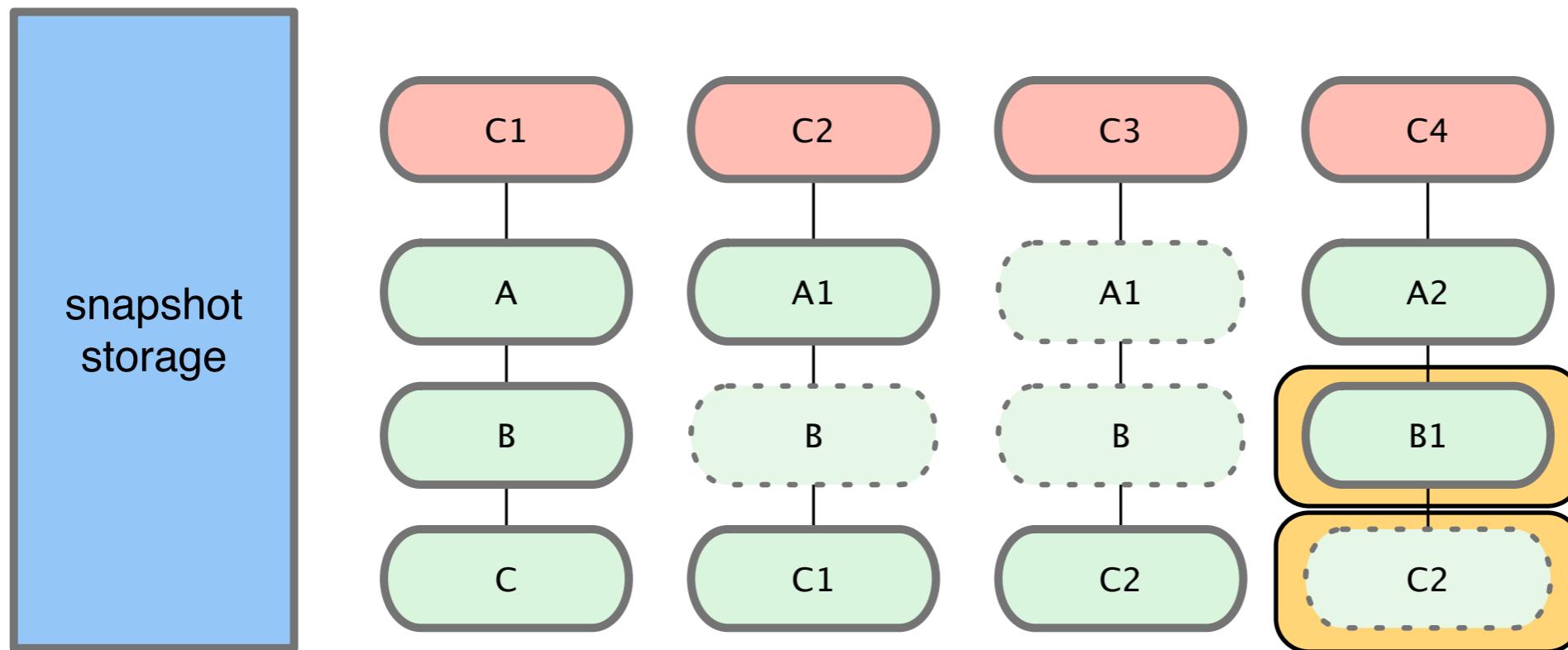
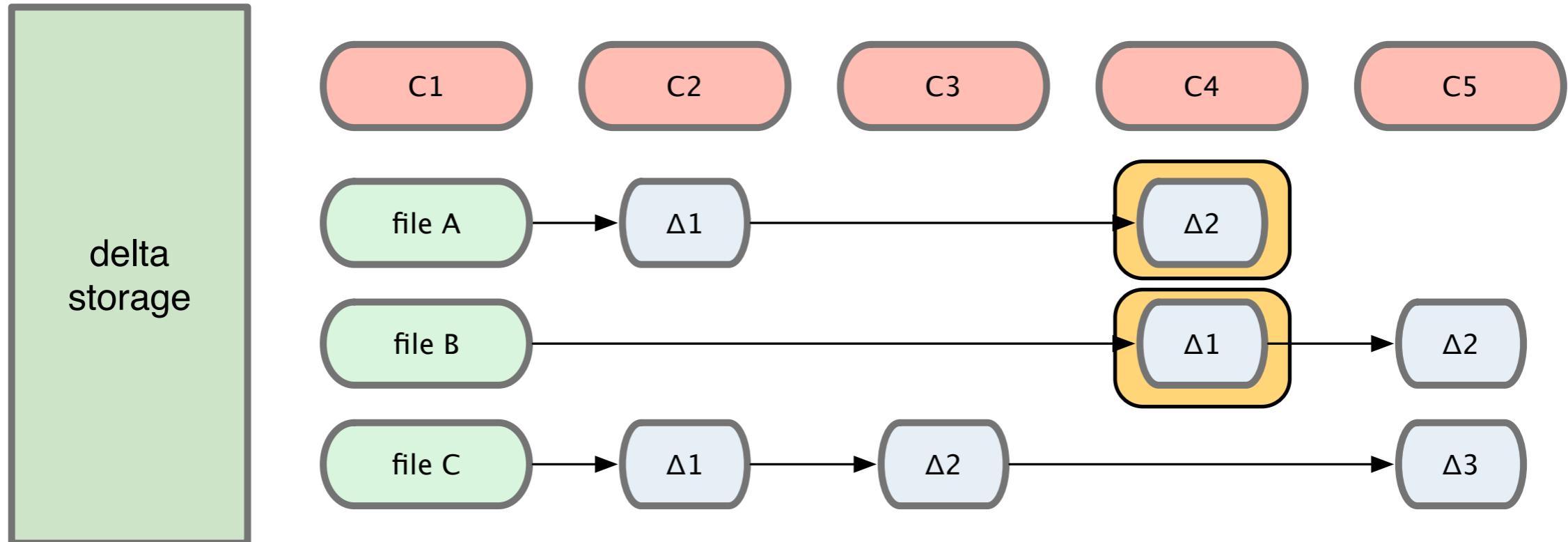


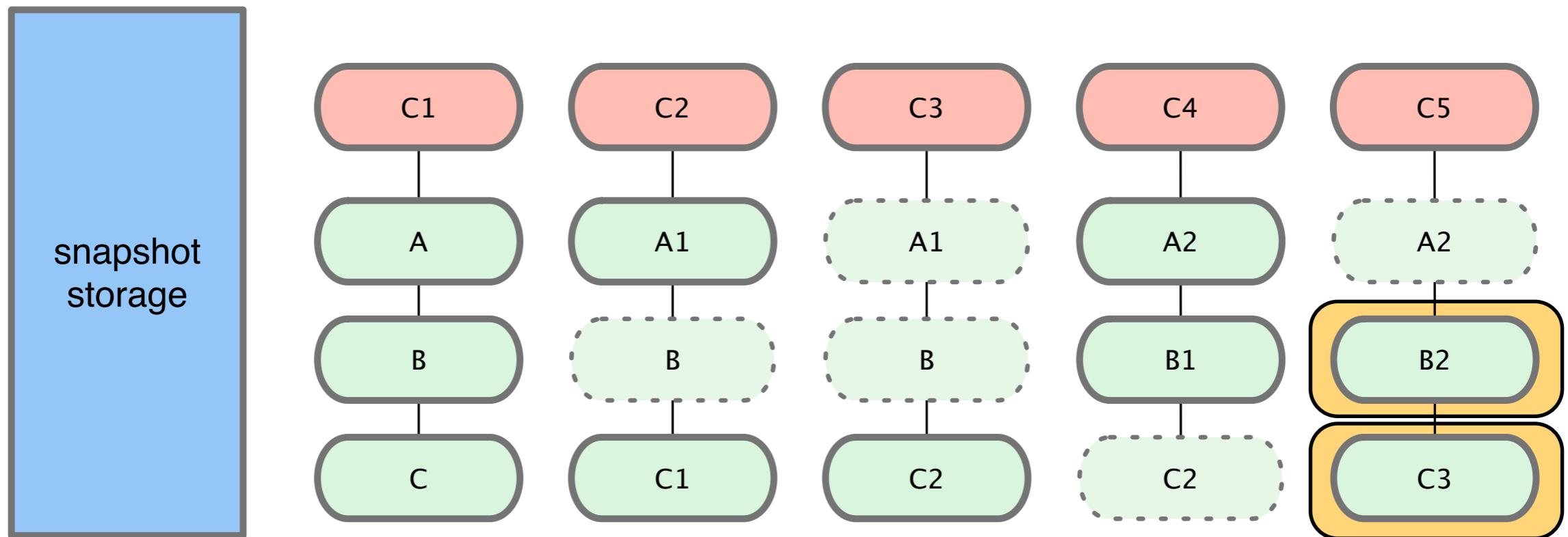
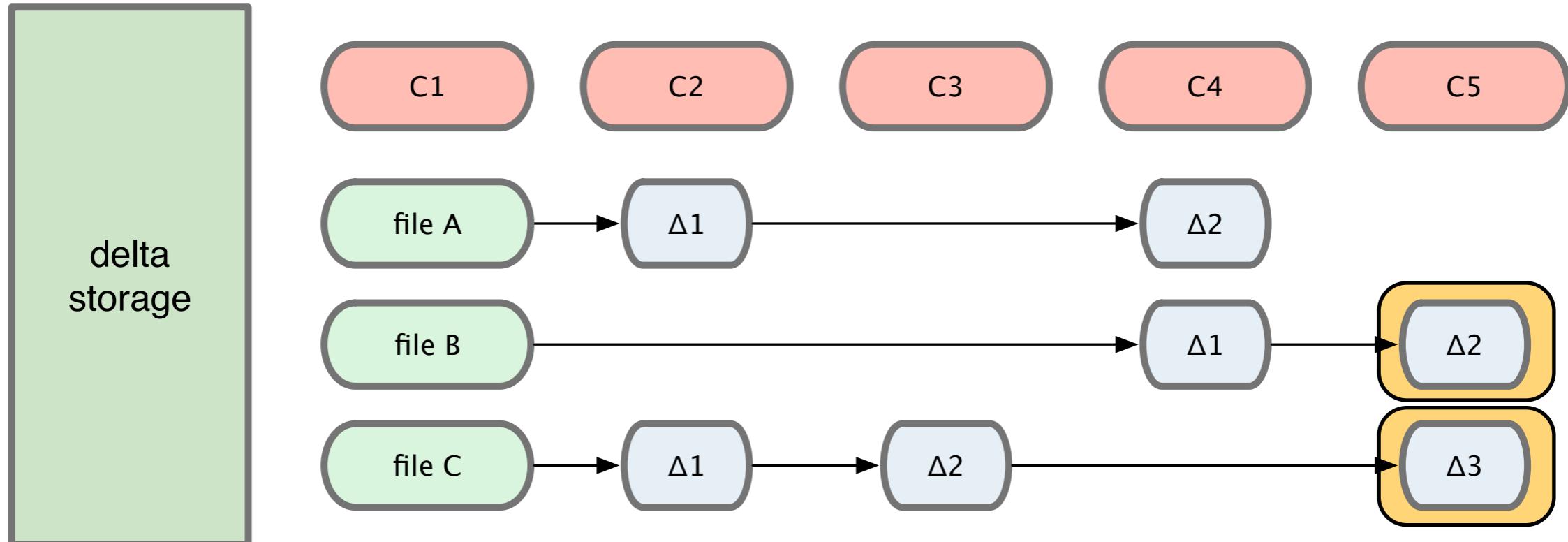


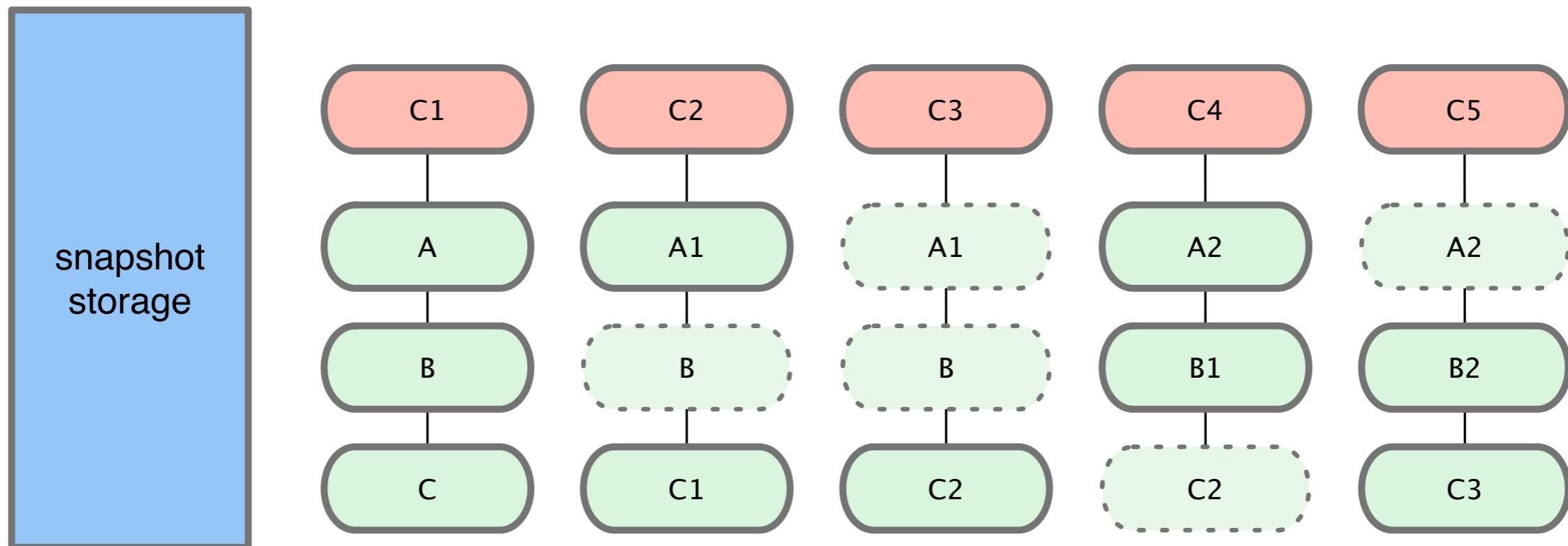
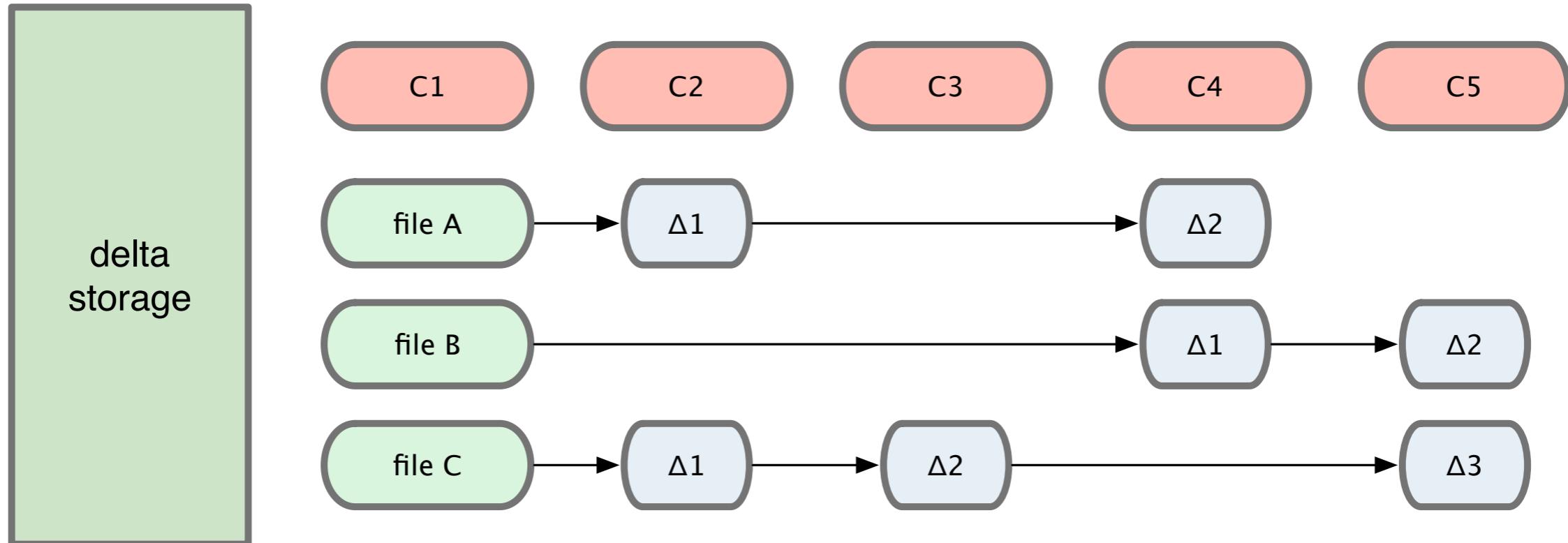












Why Git?

git ist schnell!

No Network Needed

No Network Needed

Performing a diff

No Network Needed

Performing a diff

Viewing file history

No Network Needed

Performing a diff

Viewing file history

Committing changes

No Network Needed

Performing a diff

Viewing file history

Committing changes

Merging branches

No Network Needed

Performing a diff

Viewing file history

Committing changes

Merging branches

Obtaining any other revision of a file

No Network Needed

Performing a diff

Viewing file history

Committing changes

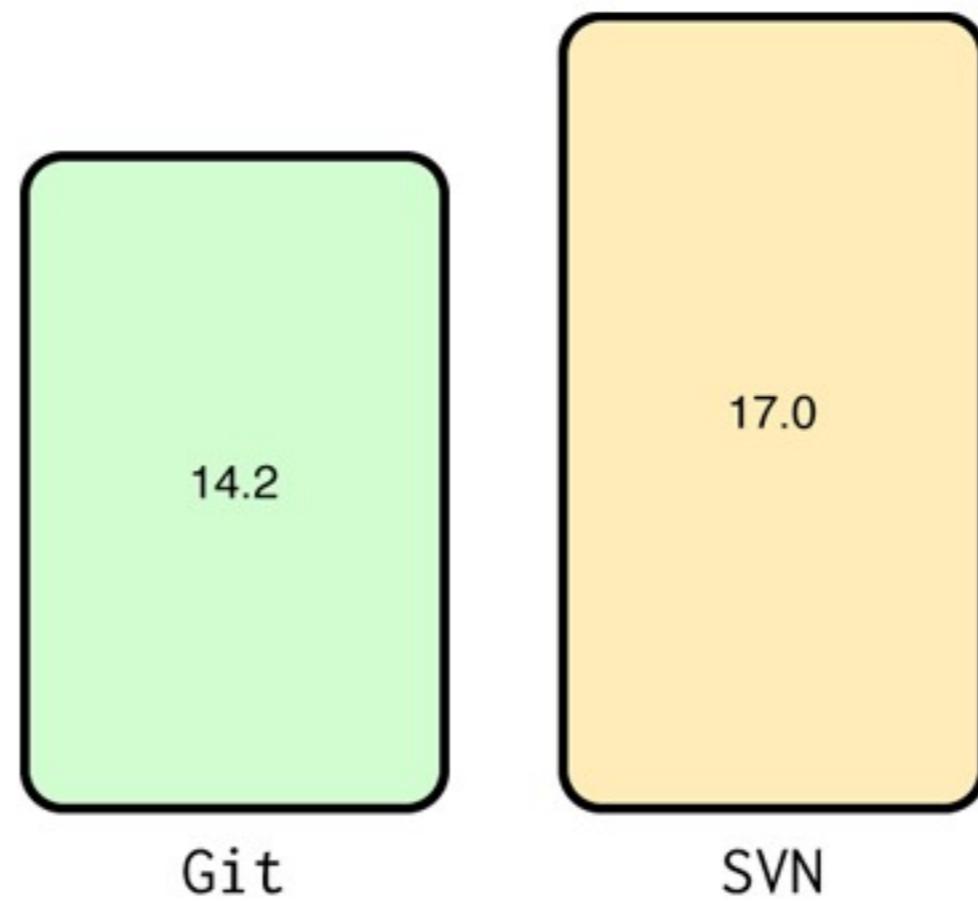
Merging branches

Obtaining any other revision of a file

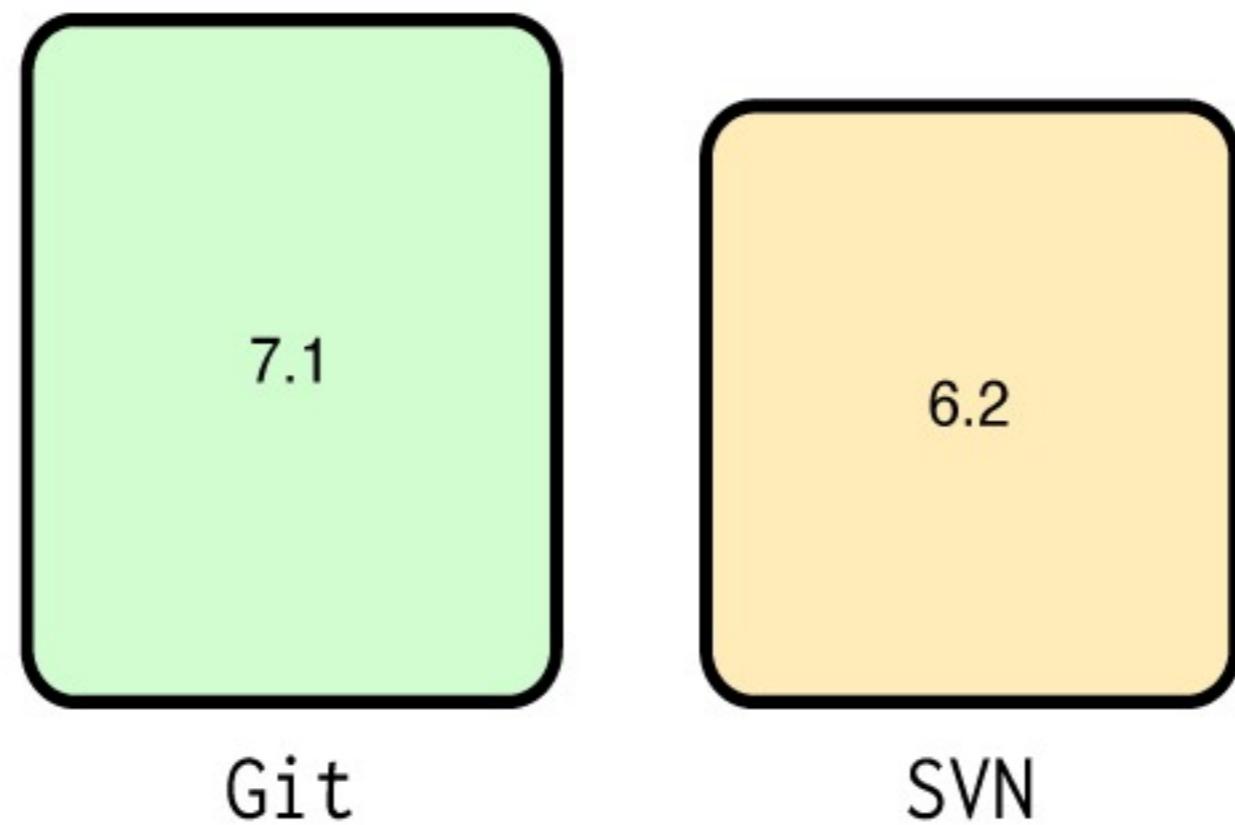
Switching branches

jQuery tests

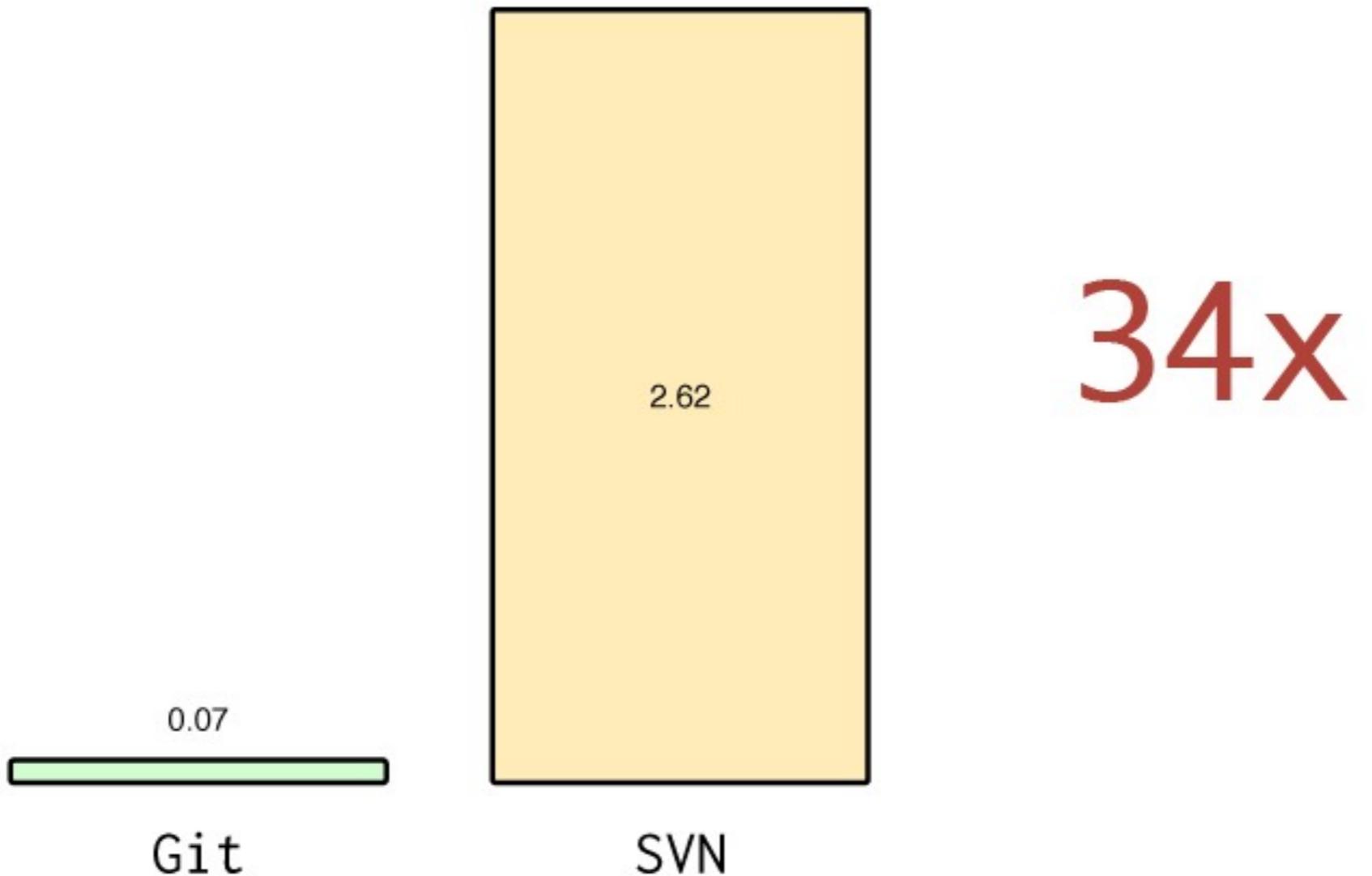
Clone/Checkout



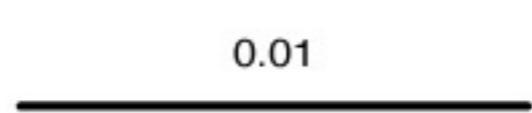
Size



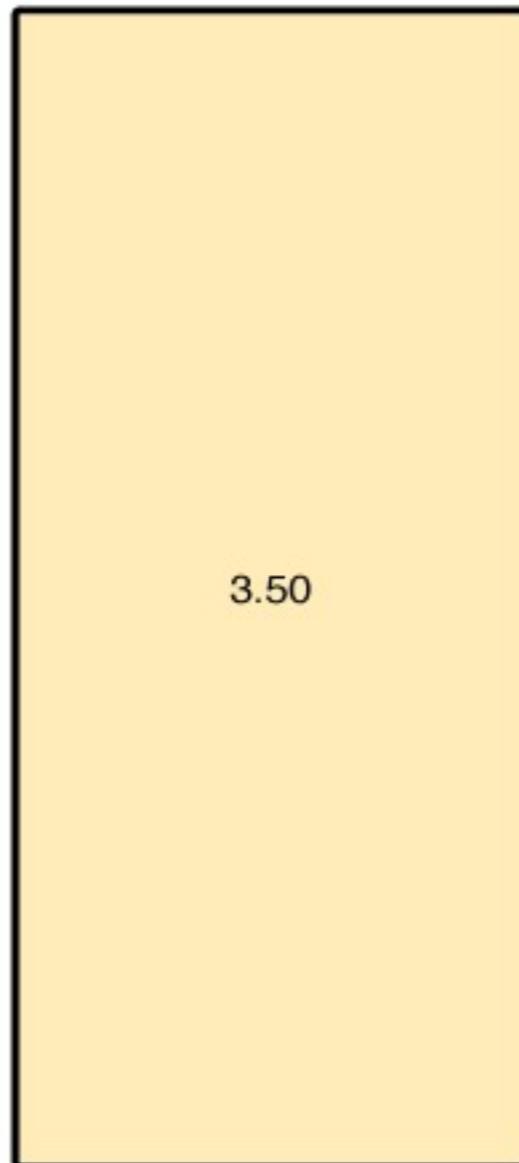
Log



Diff



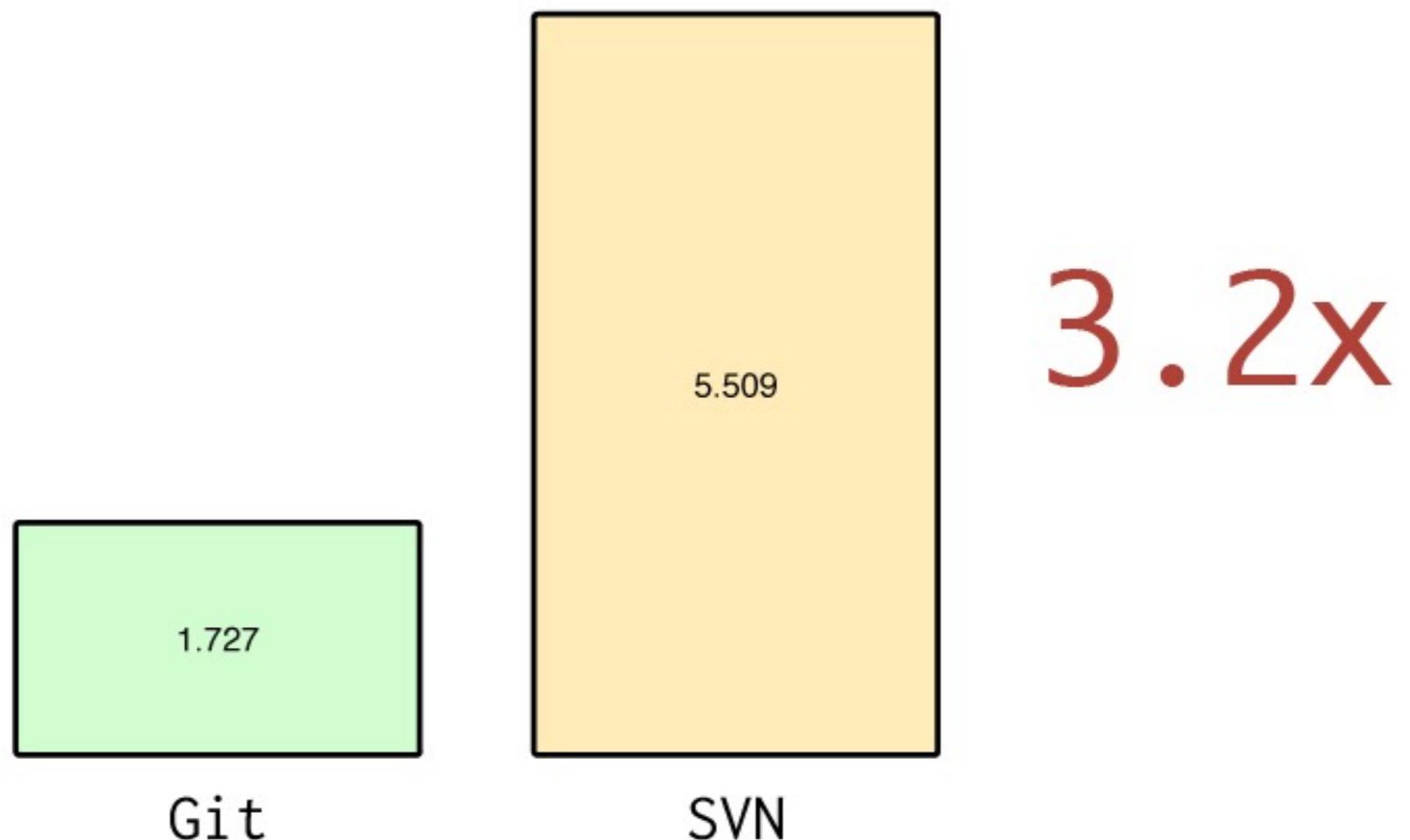
Git



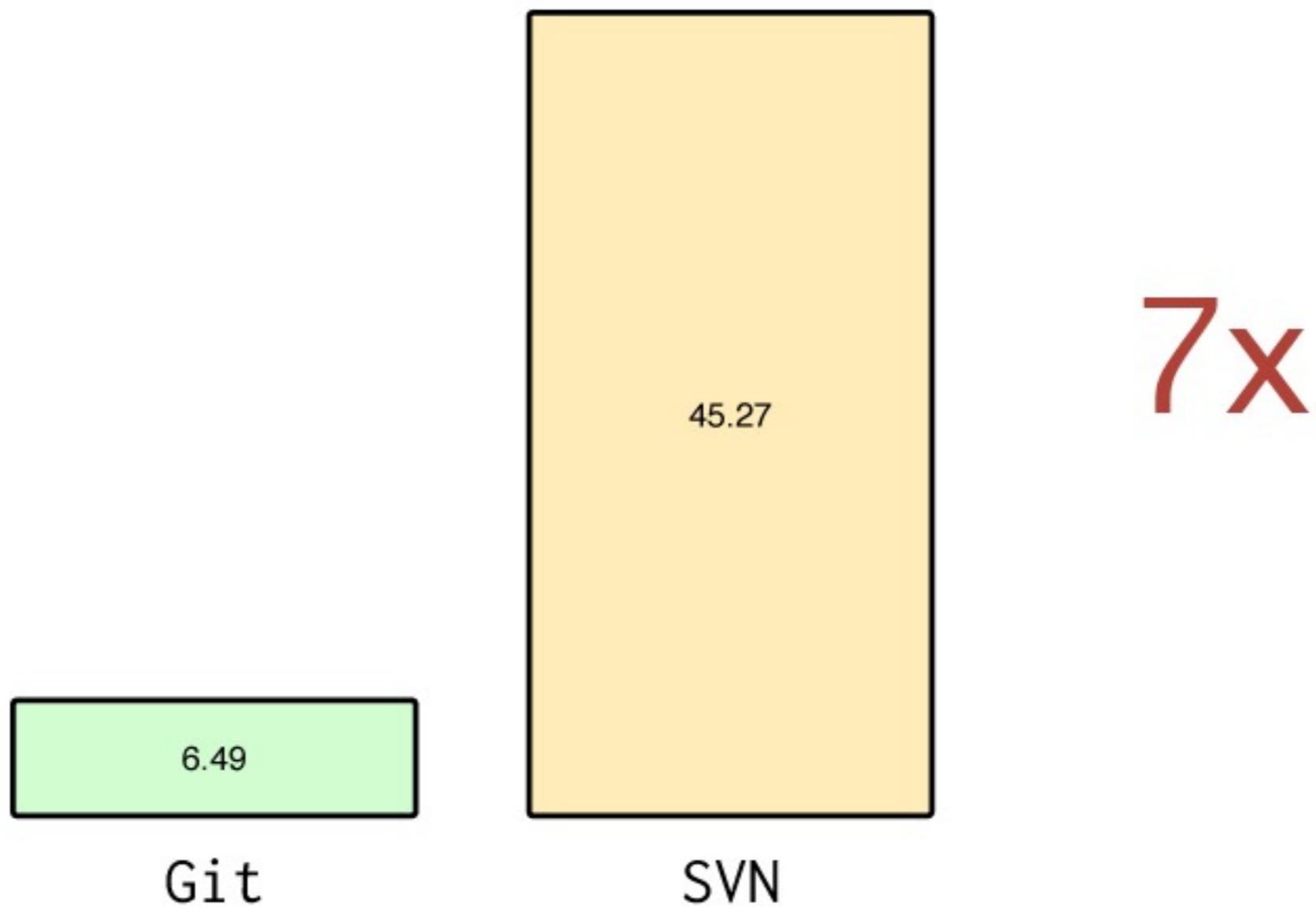
SVN

318x

Commit

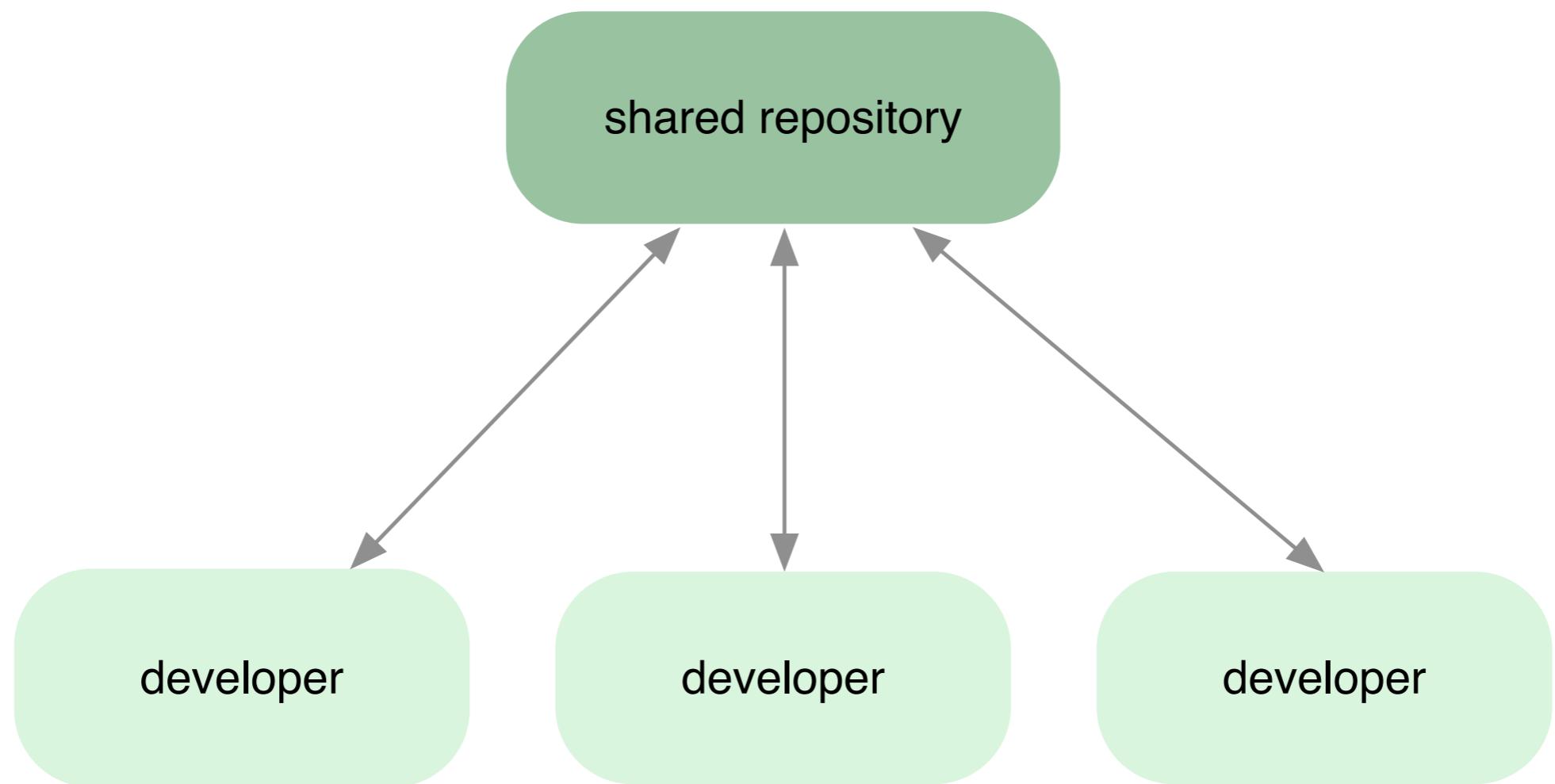


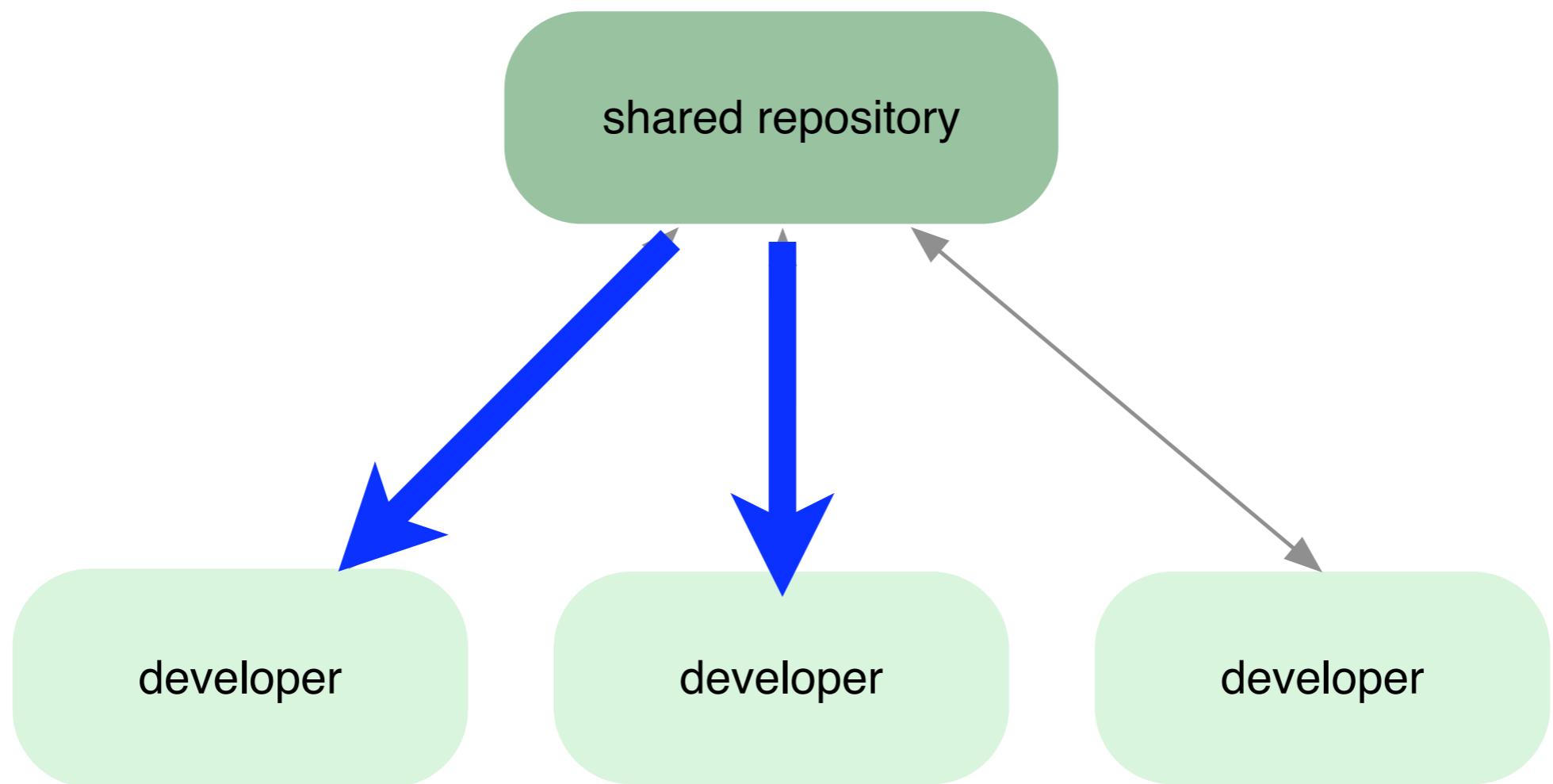
Substantial Commit

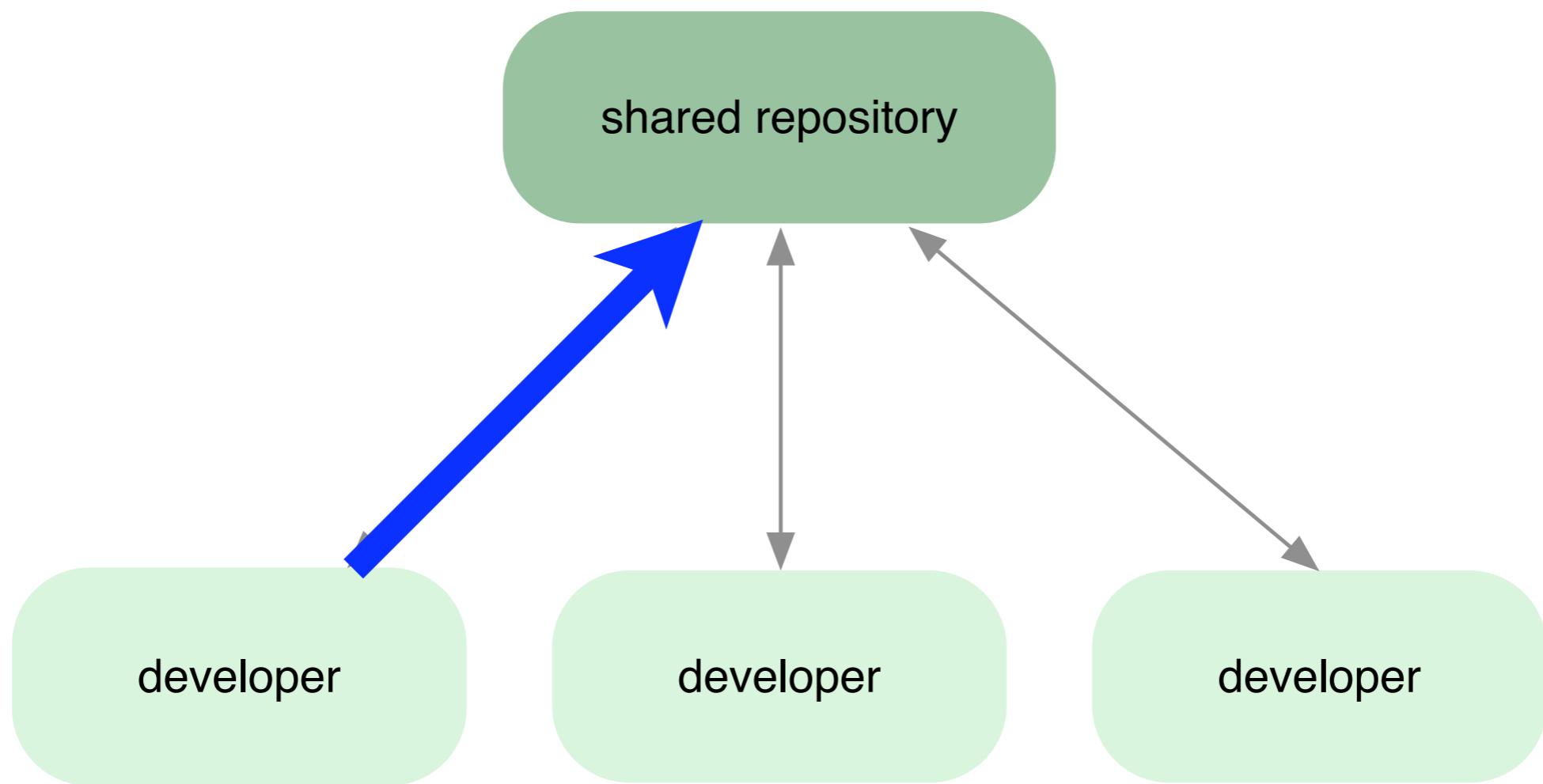


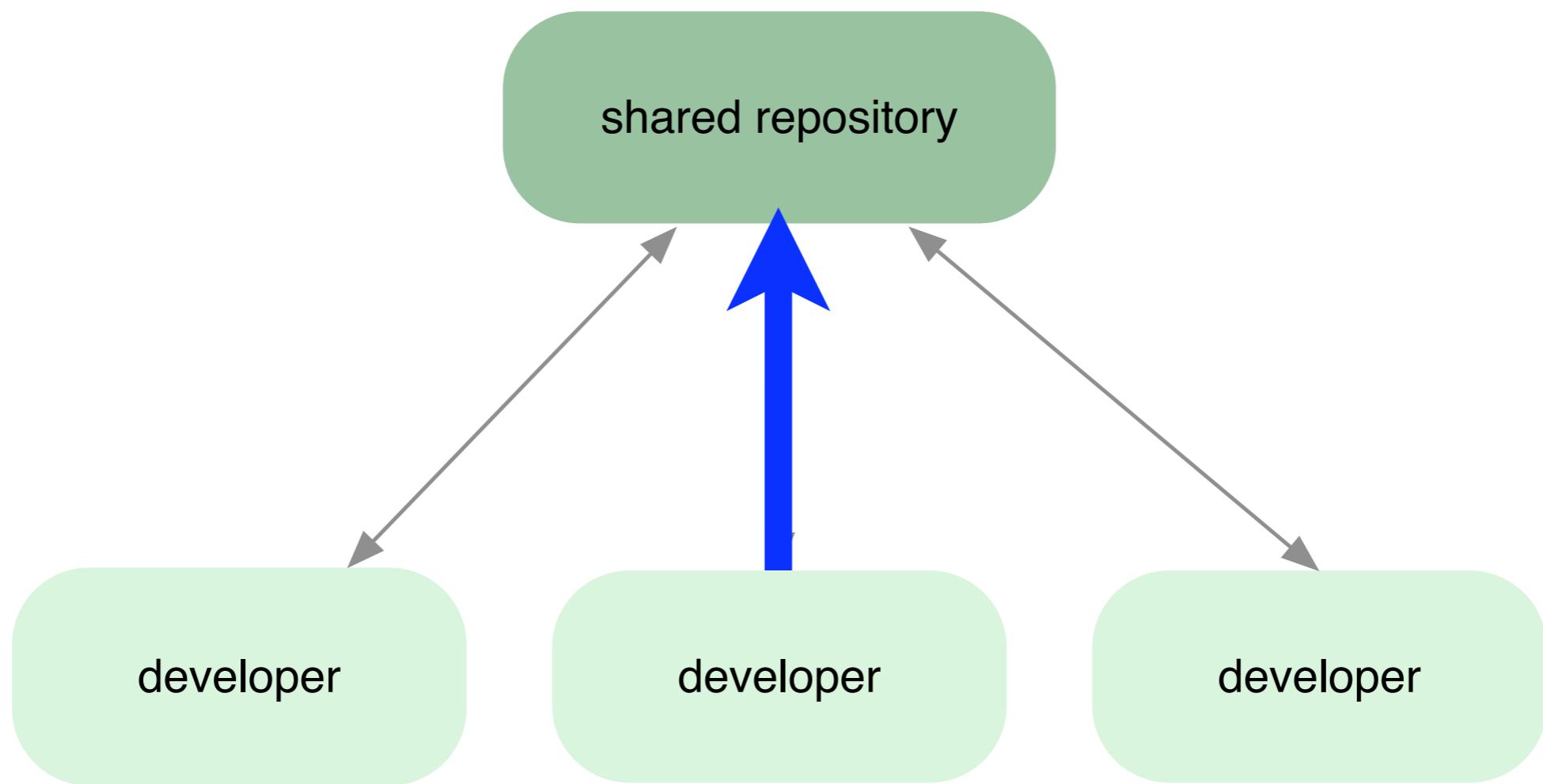
more workflows

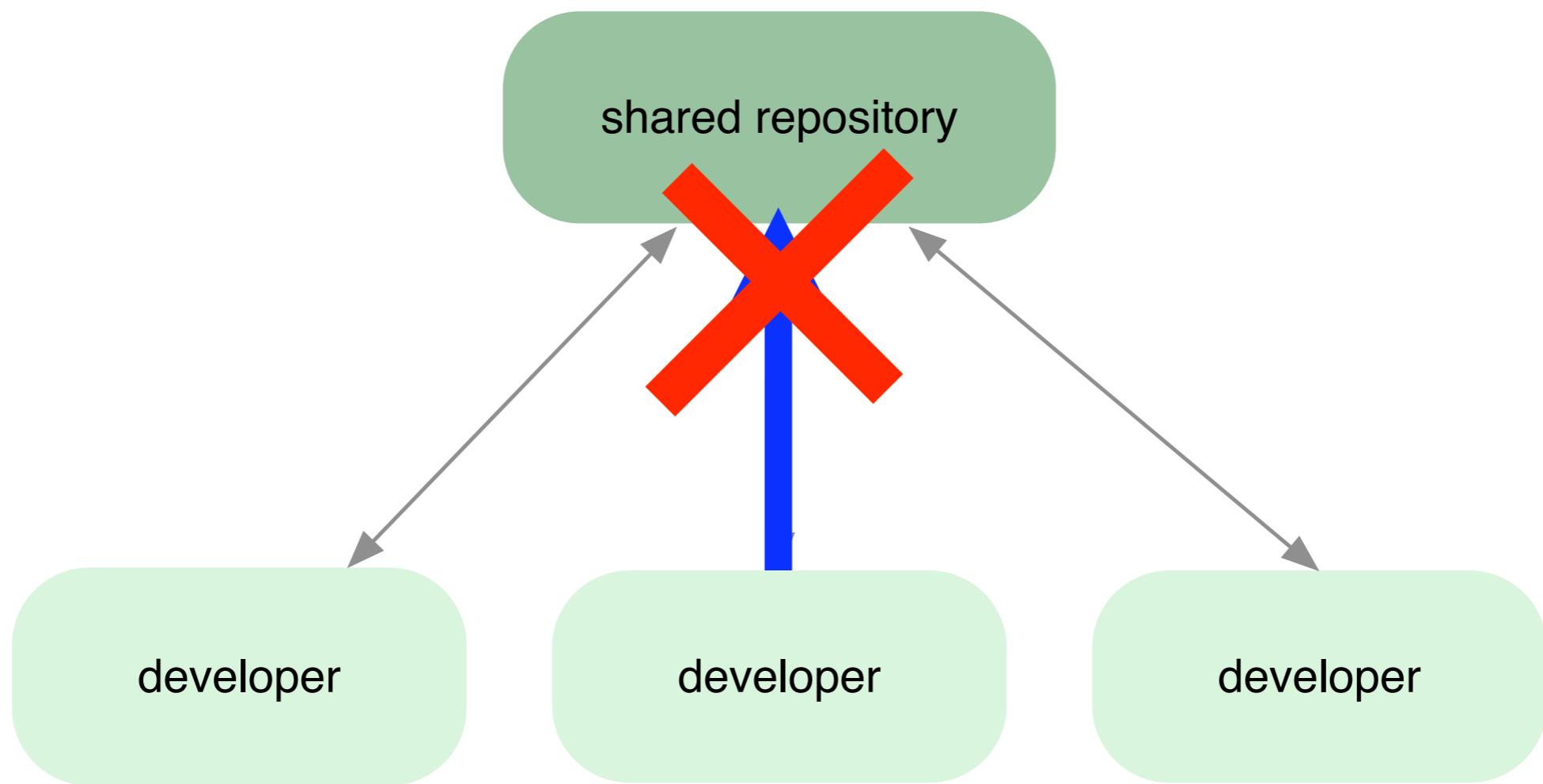
centralized

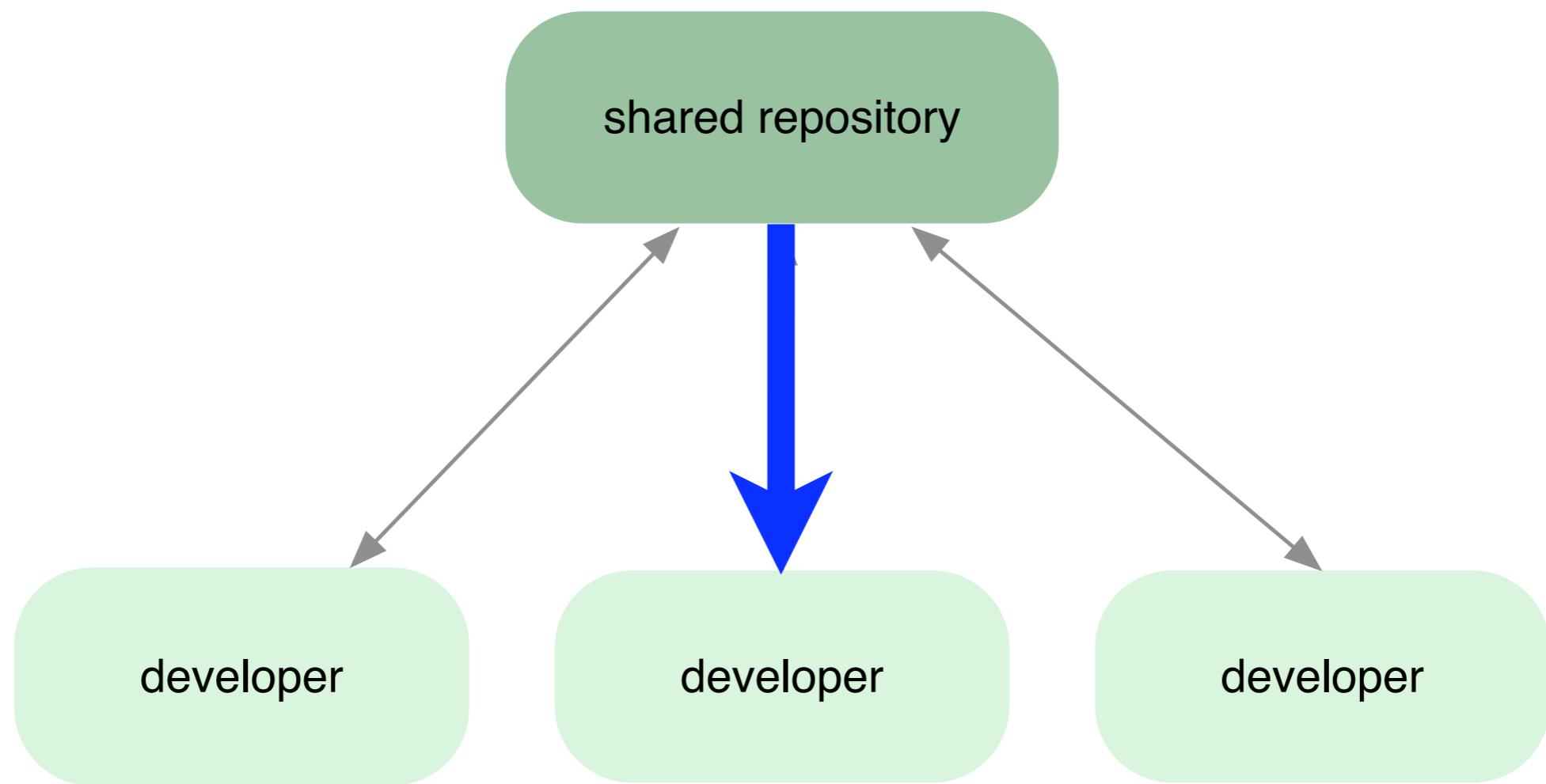


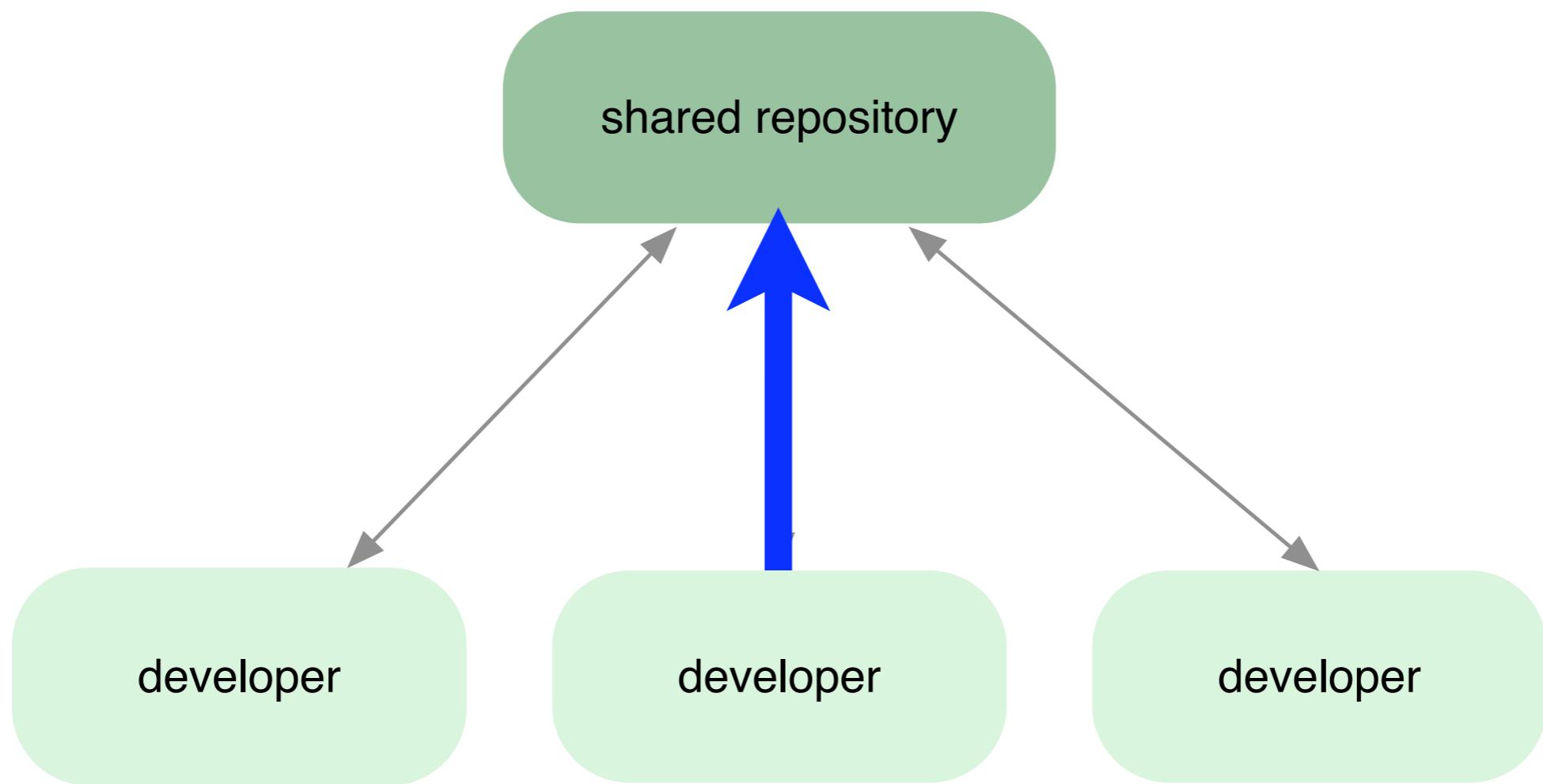




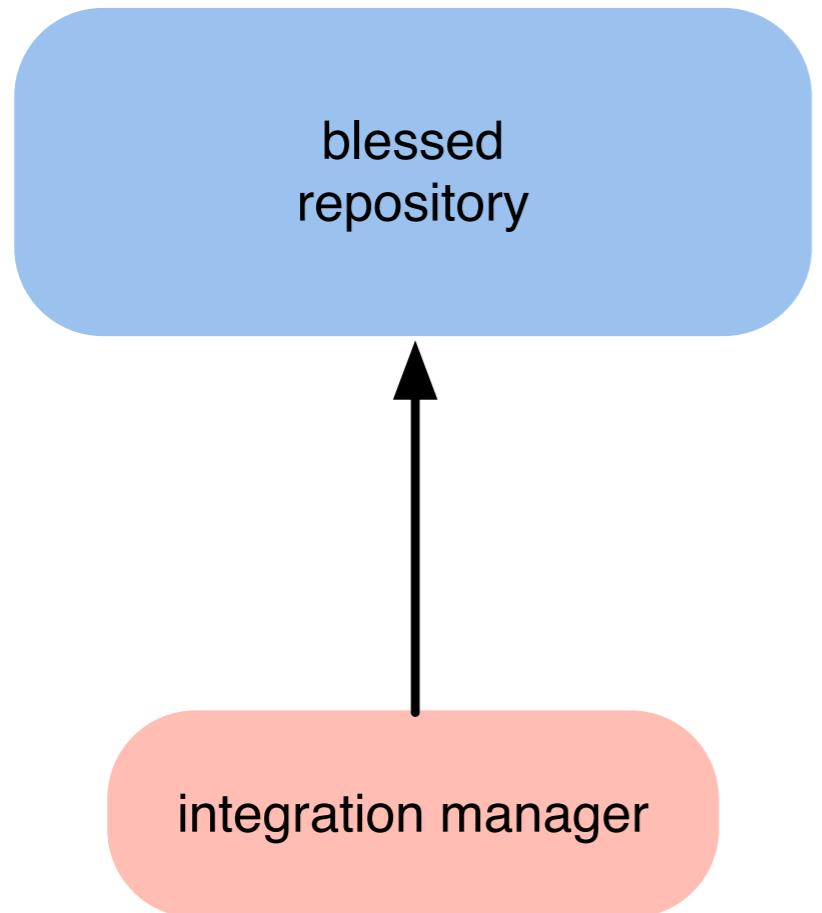


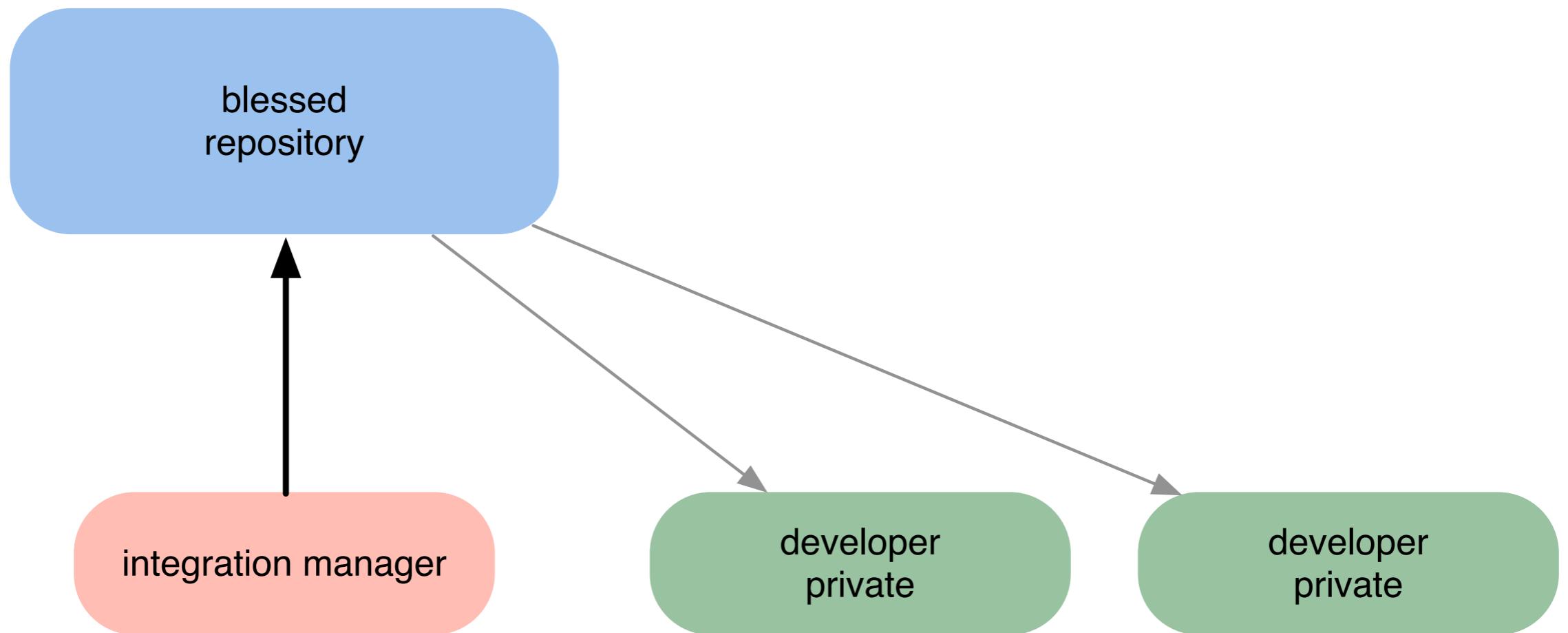


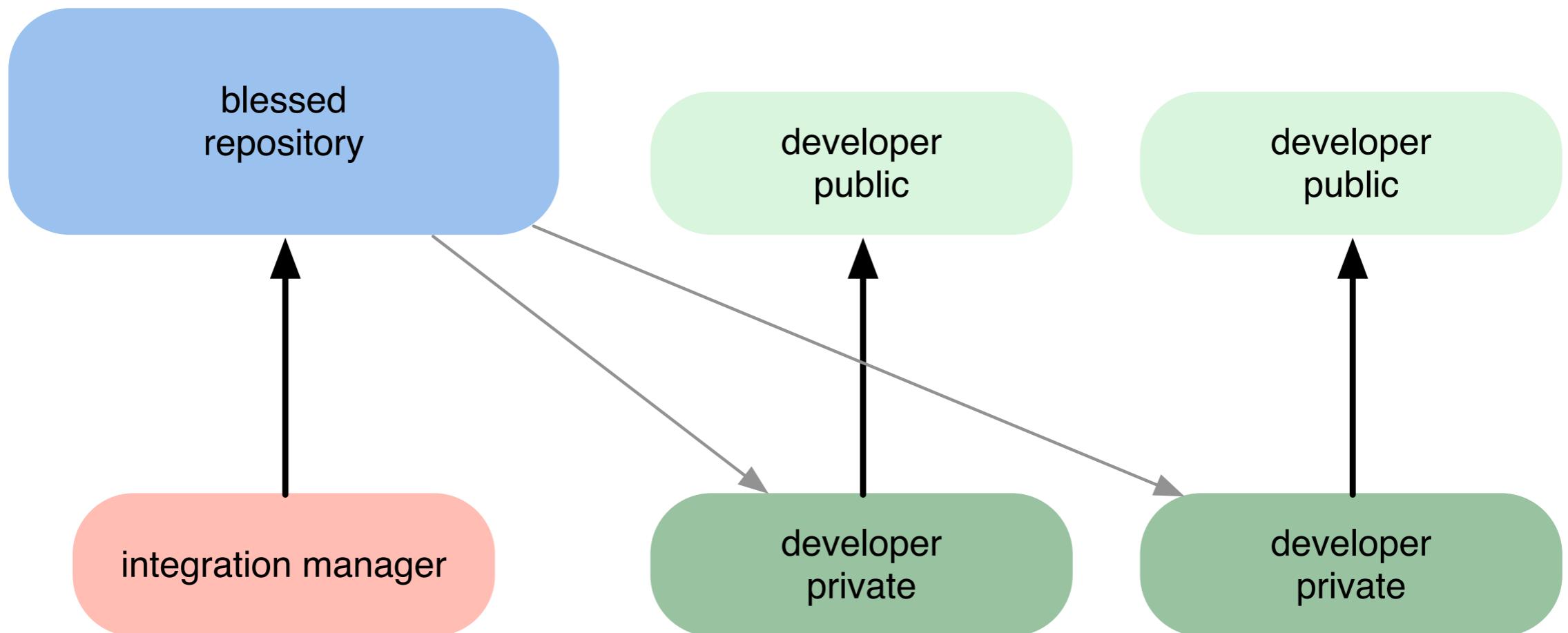


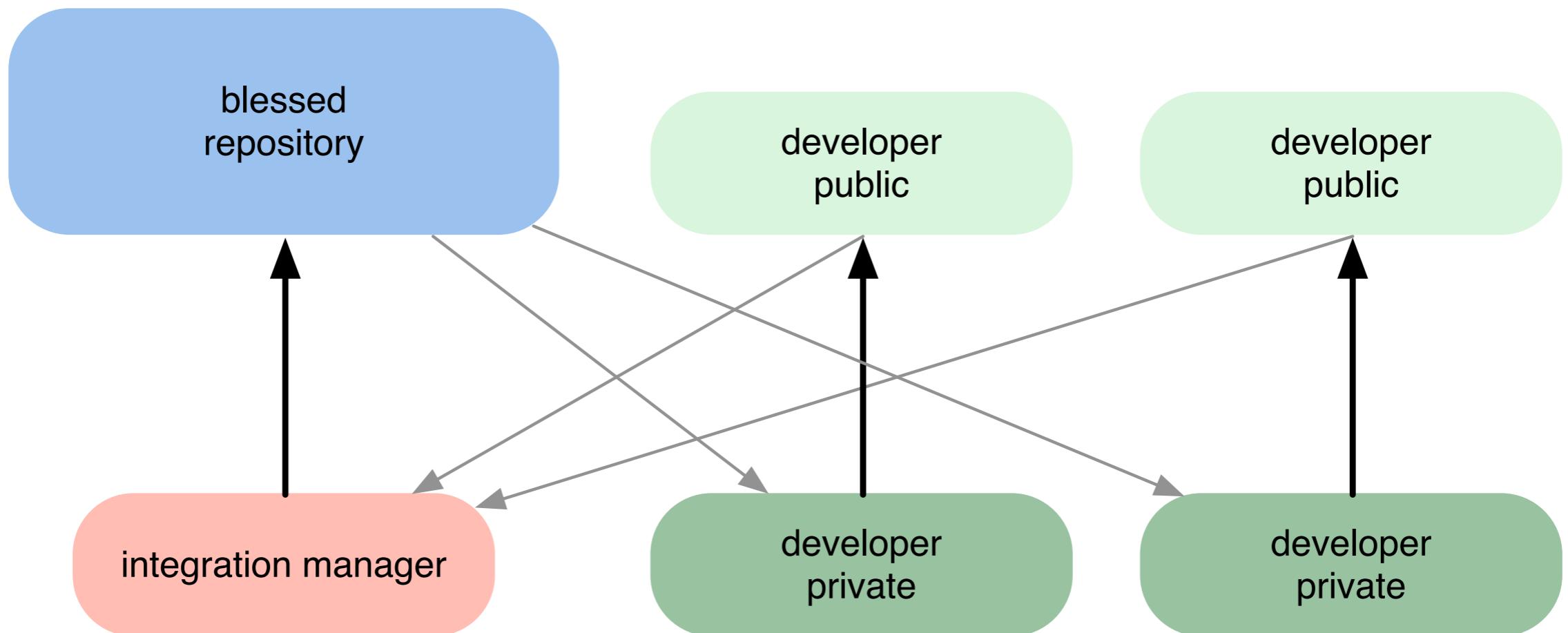


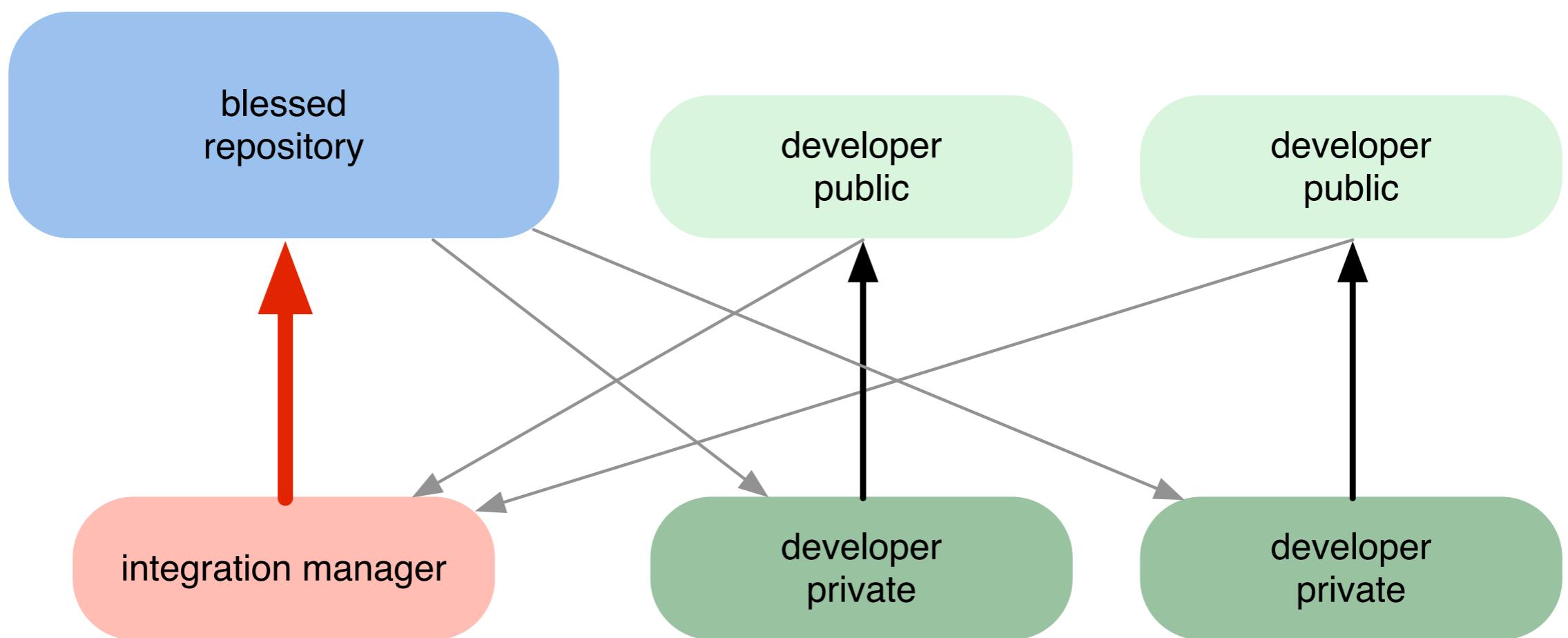
integration manager



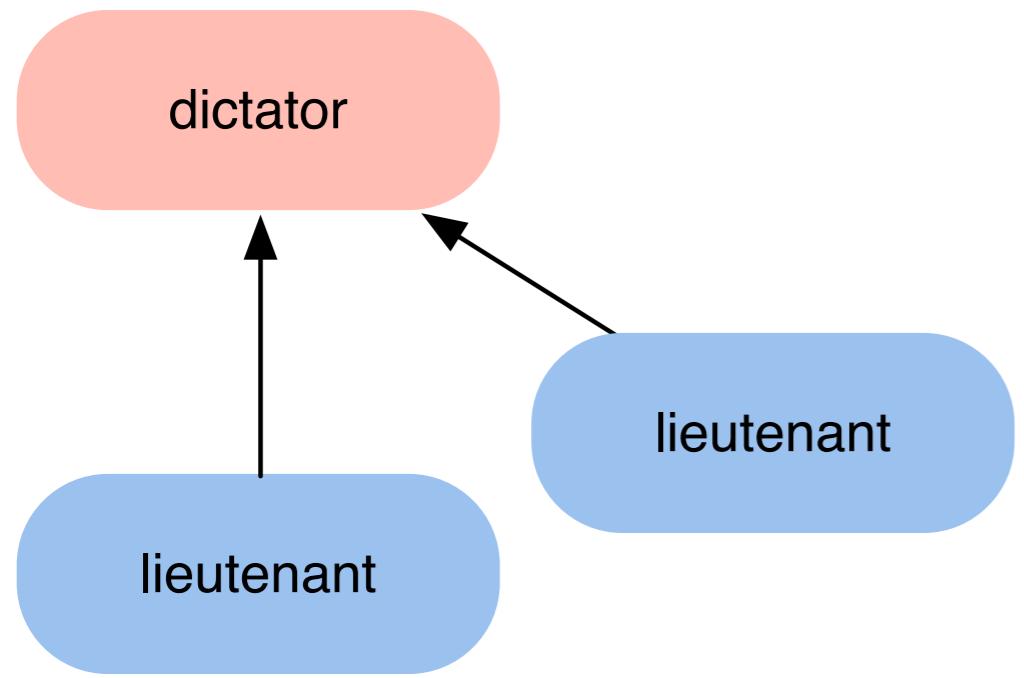


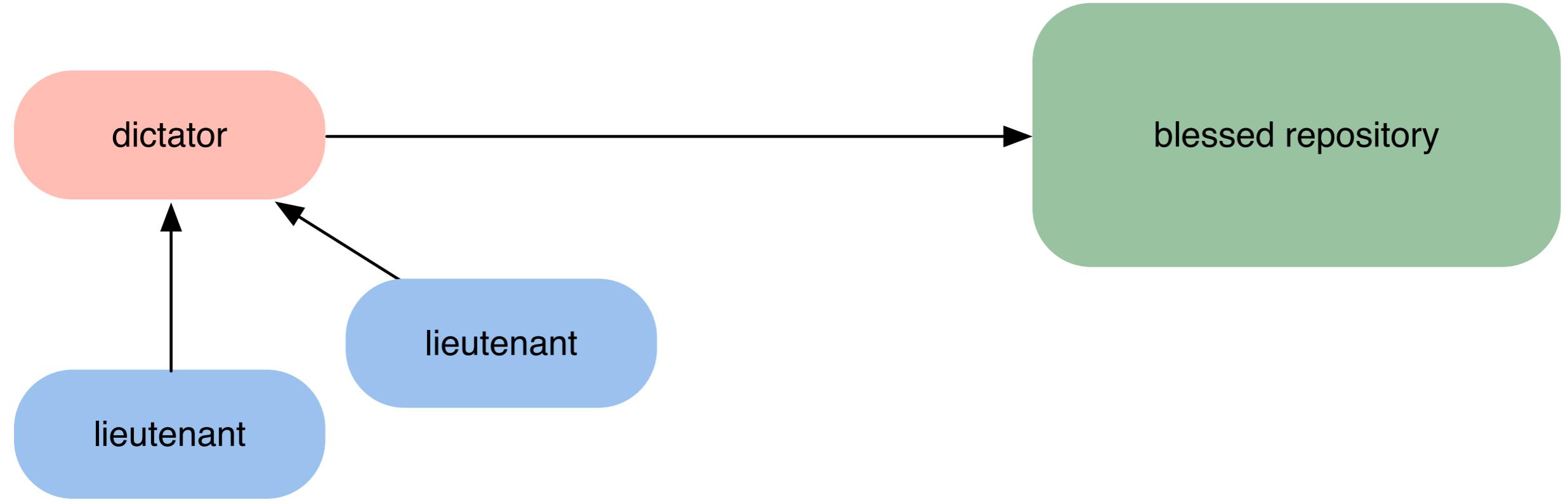


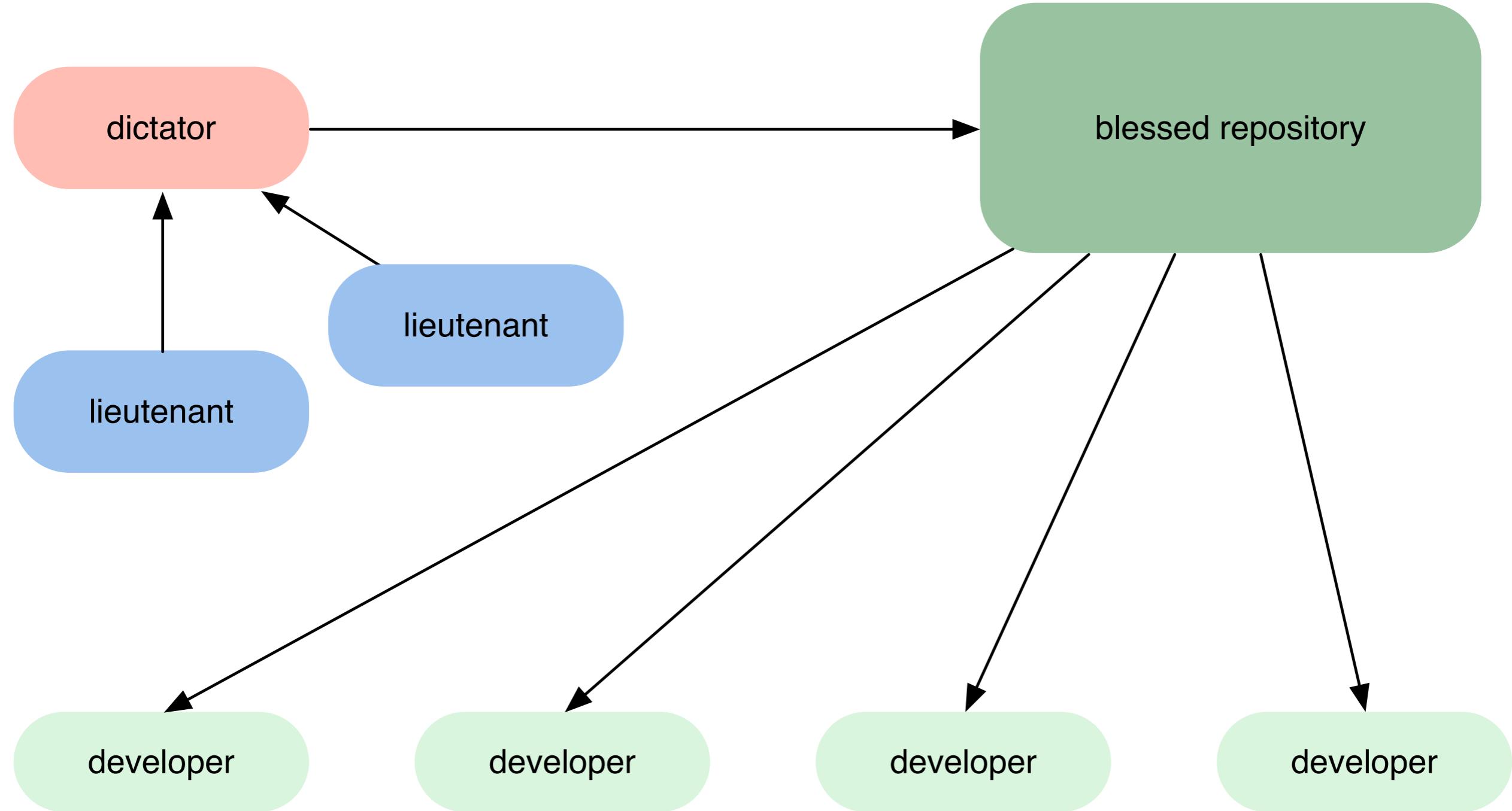


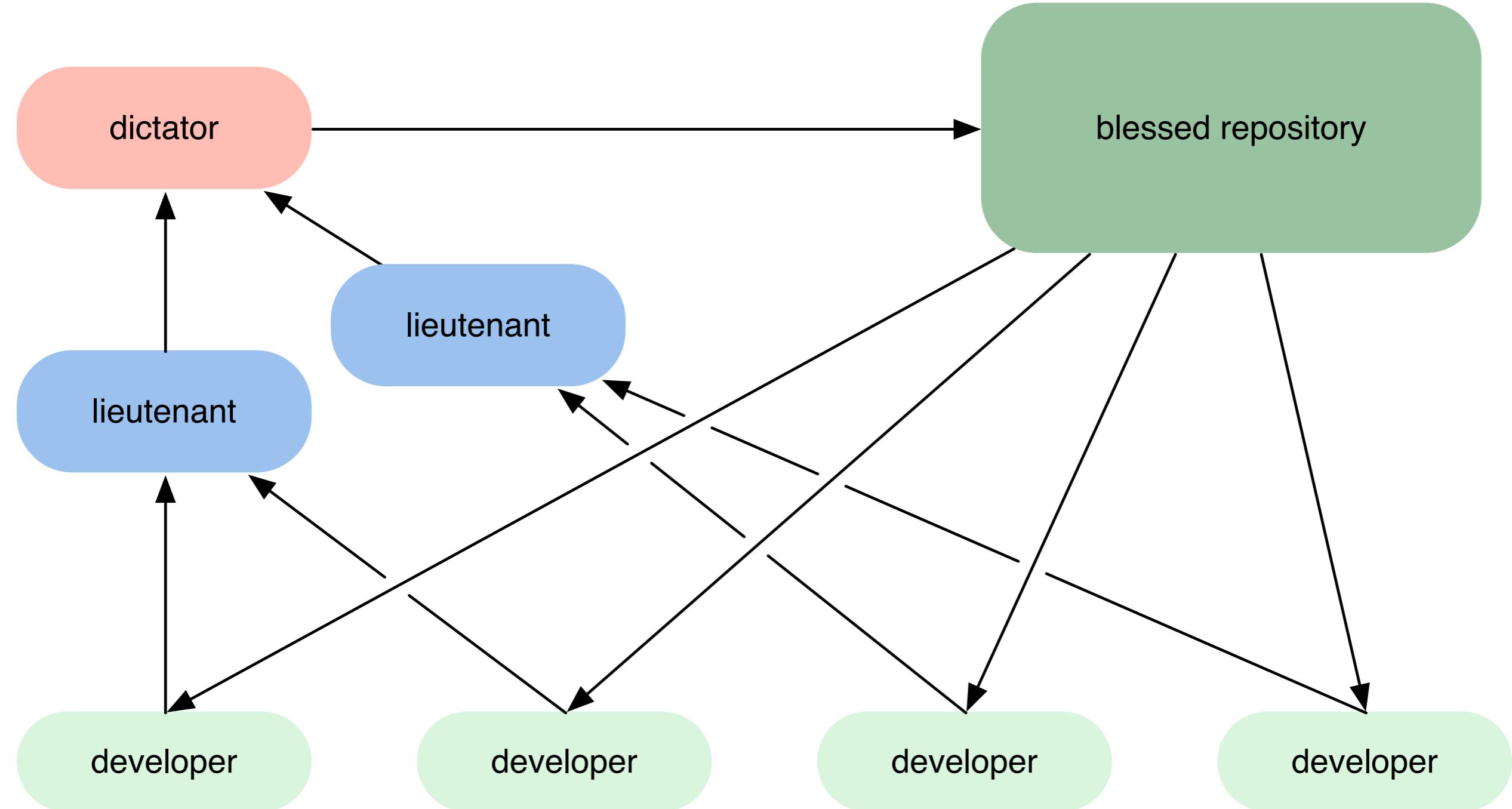


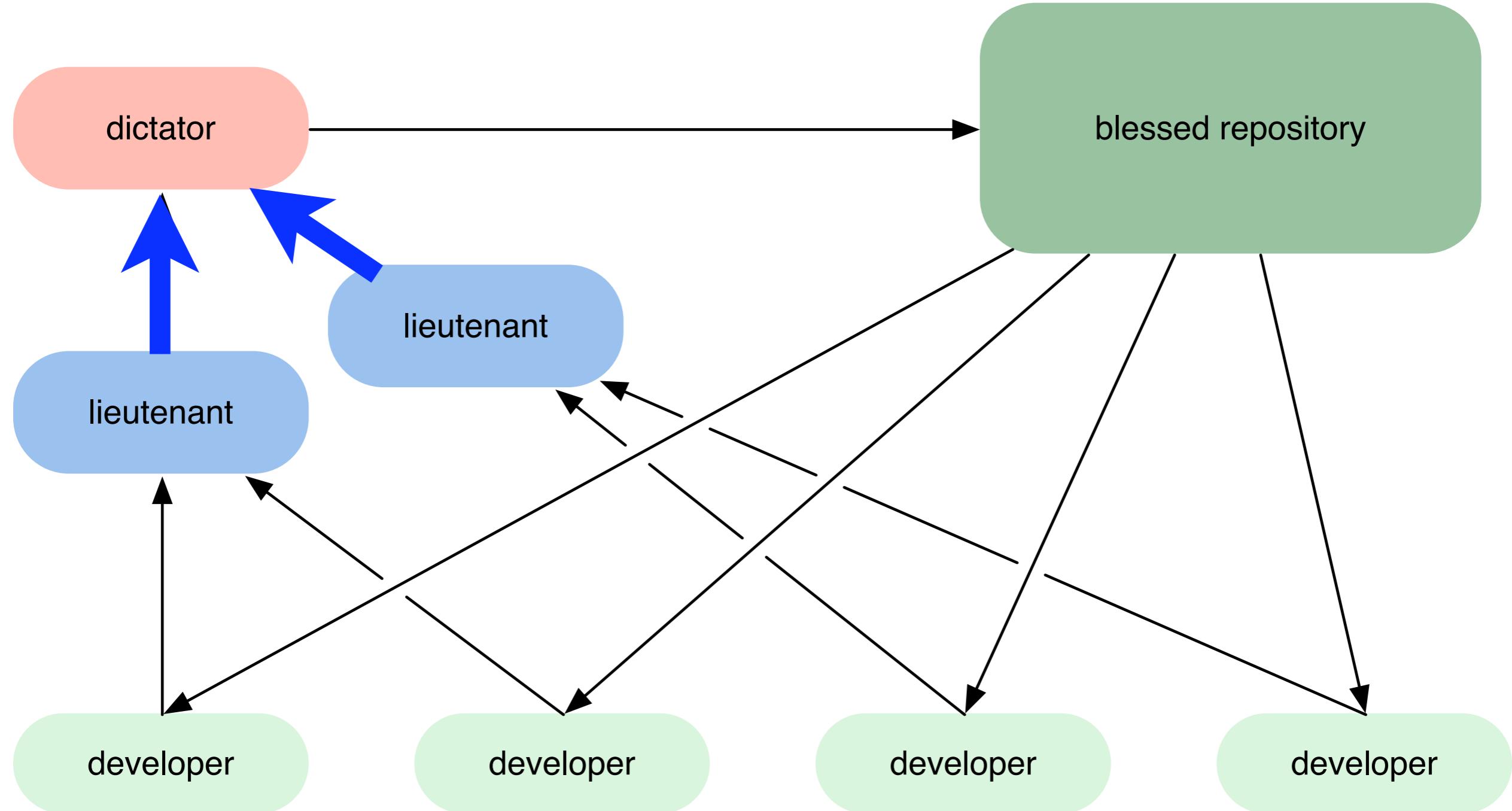
dictator / lieutenant

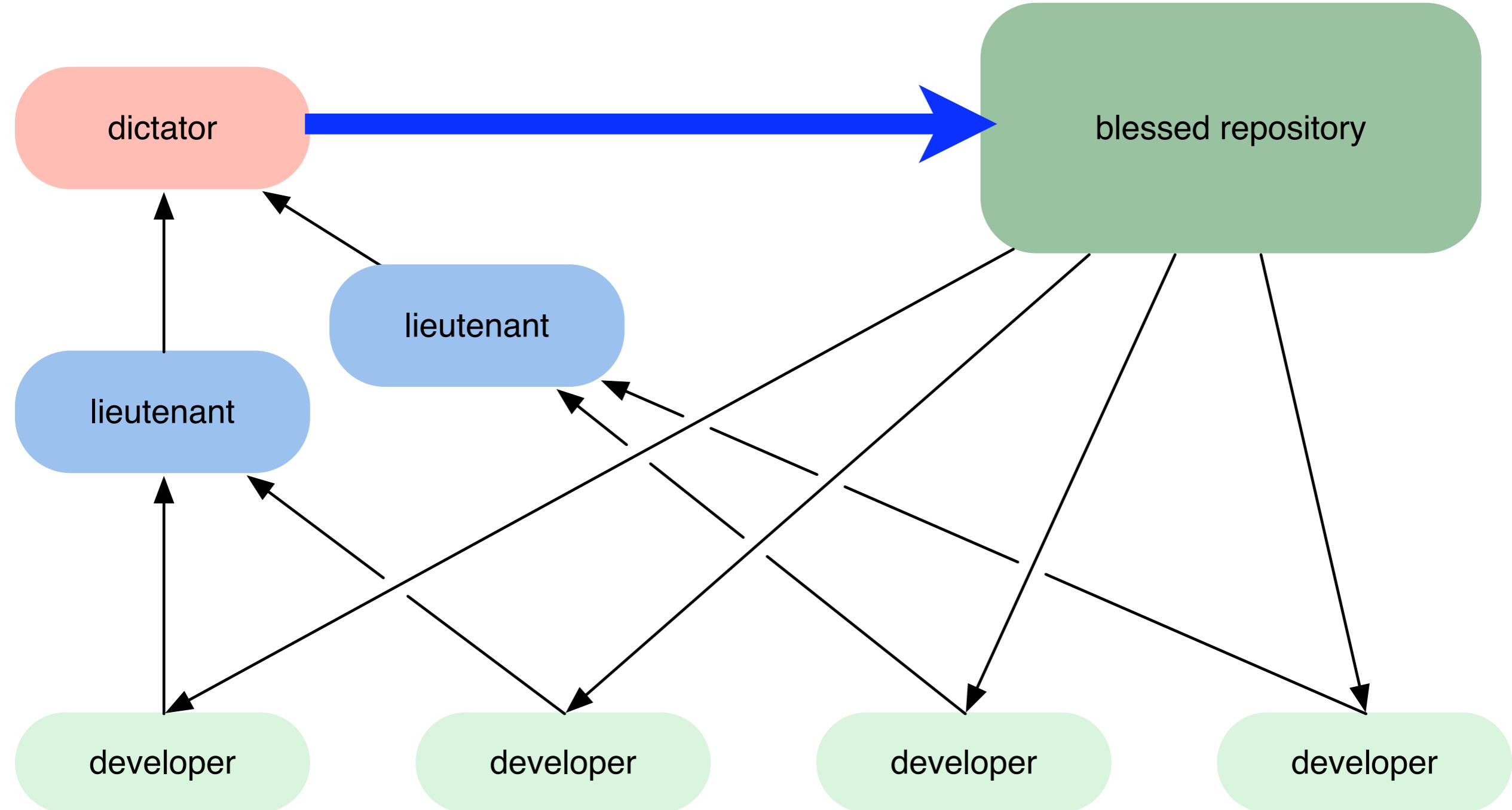


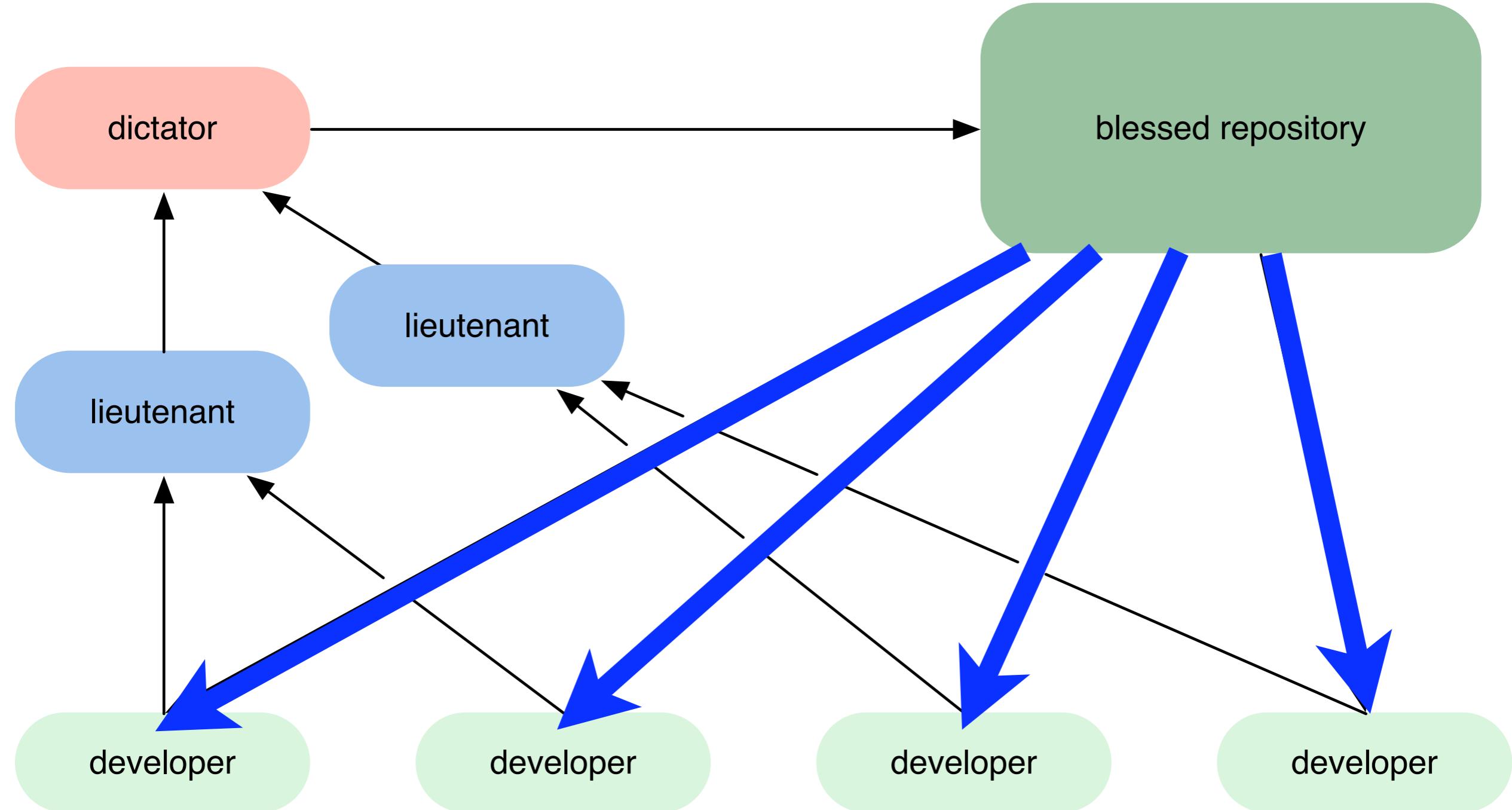


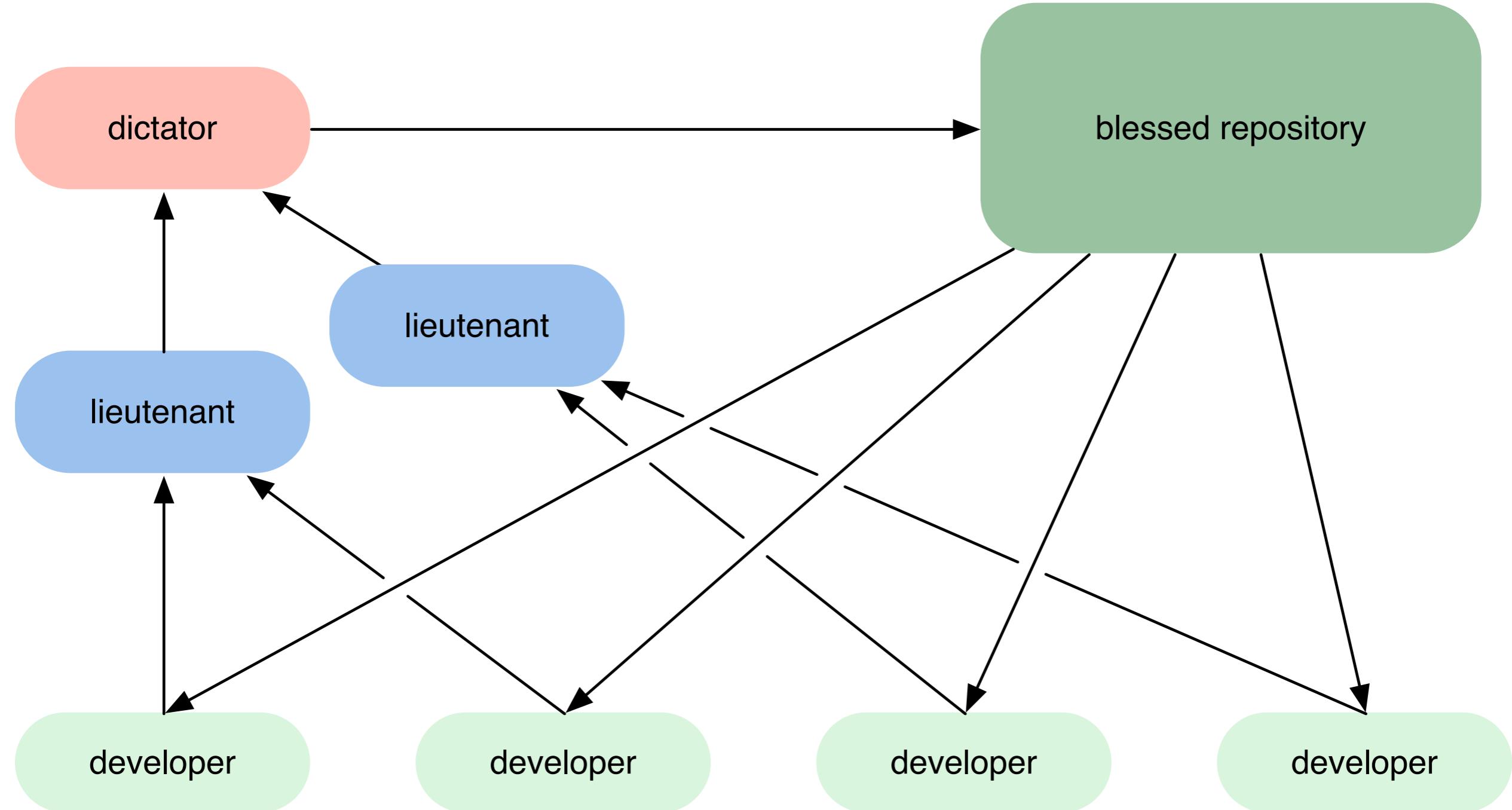












cheap local branching

cheap local branching
and easy merging

```
$ du -h testrepo/  
78M testrepo/
```

```
$ du -h testrepo/  
78M testrepo/  
$ cd testrepo/
```

```
$ du -h testrepo/  
78M testrepo/  
$ cd testrepo/  
$ time git checkout -b newbranch
```

```
$ du -h testrepo/  
78M testrepo/  
$ cd testrepo/  
$ time git checkout -b newbranch  
Switched to a new branch "newbranch"  
  
real 0m0.091s  
user 0m0.016s  
sys 0m0.023s
```

```
$ du -h testrepo/  
78M testrepo/  
$ cd testrepo/  
$ time git checkout -b newbranch  
Switched to a new branch "newbranch"
```

```
real 0m0.091s  
user 0m0.016s  
sys 0m0.023s
```

```
$ time cp -Rf testrepo r2
```

```
real 0m13.684s
user 0m0.085s
sys  0m1.215s
```

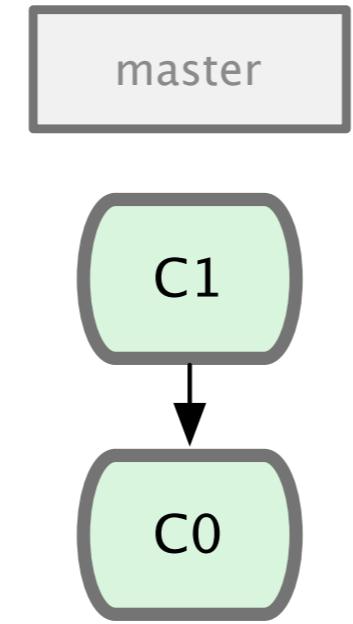
| 3.68 vs 0.09

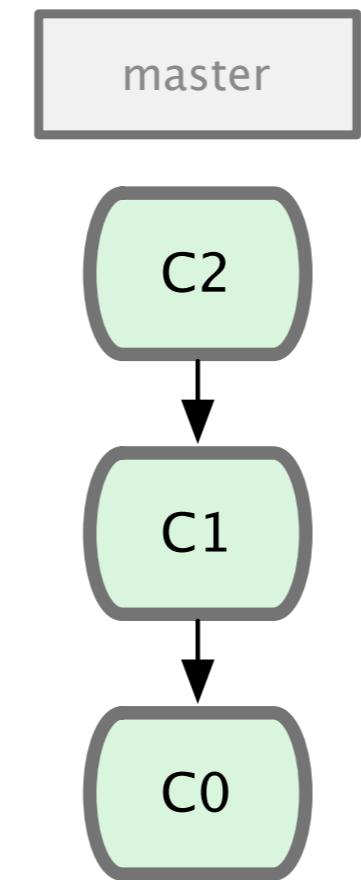
so what?

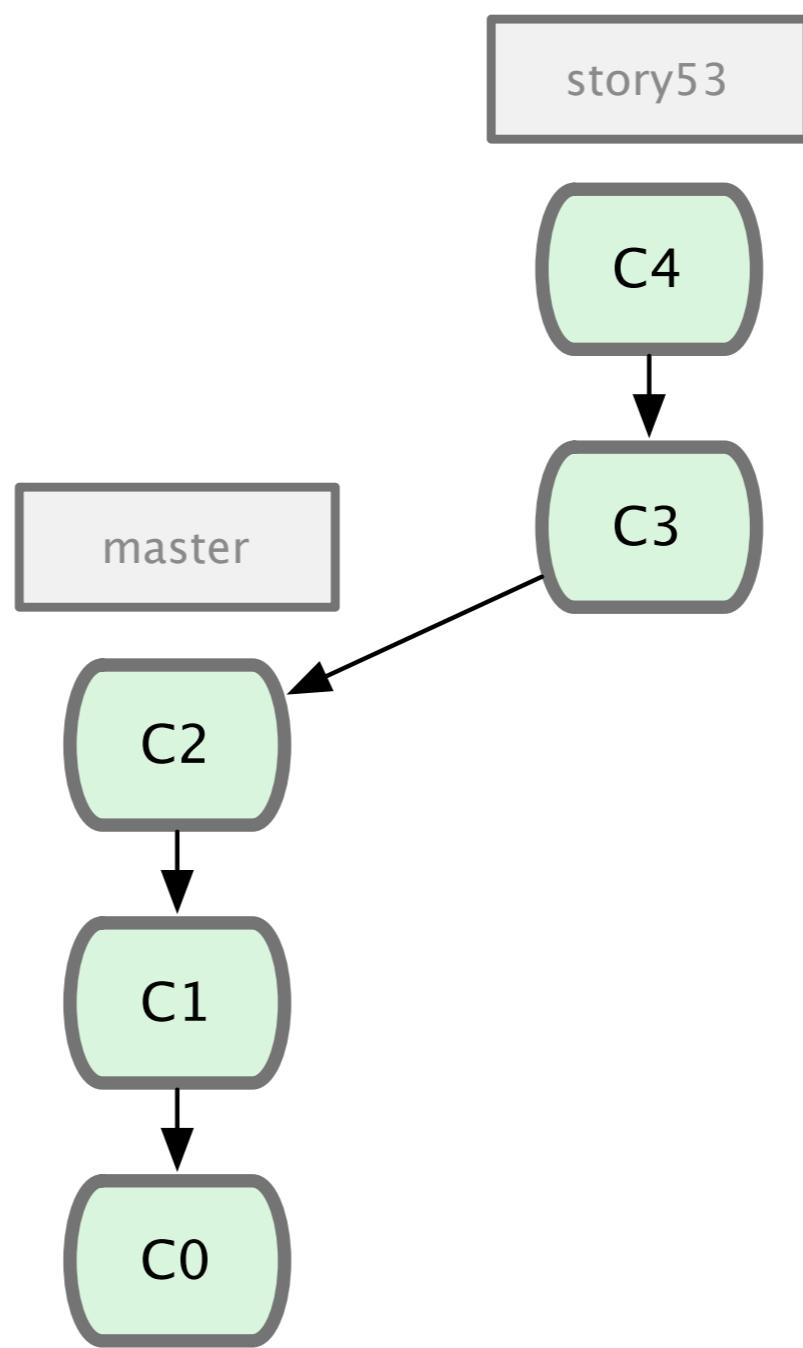
**frictionless context
switching**

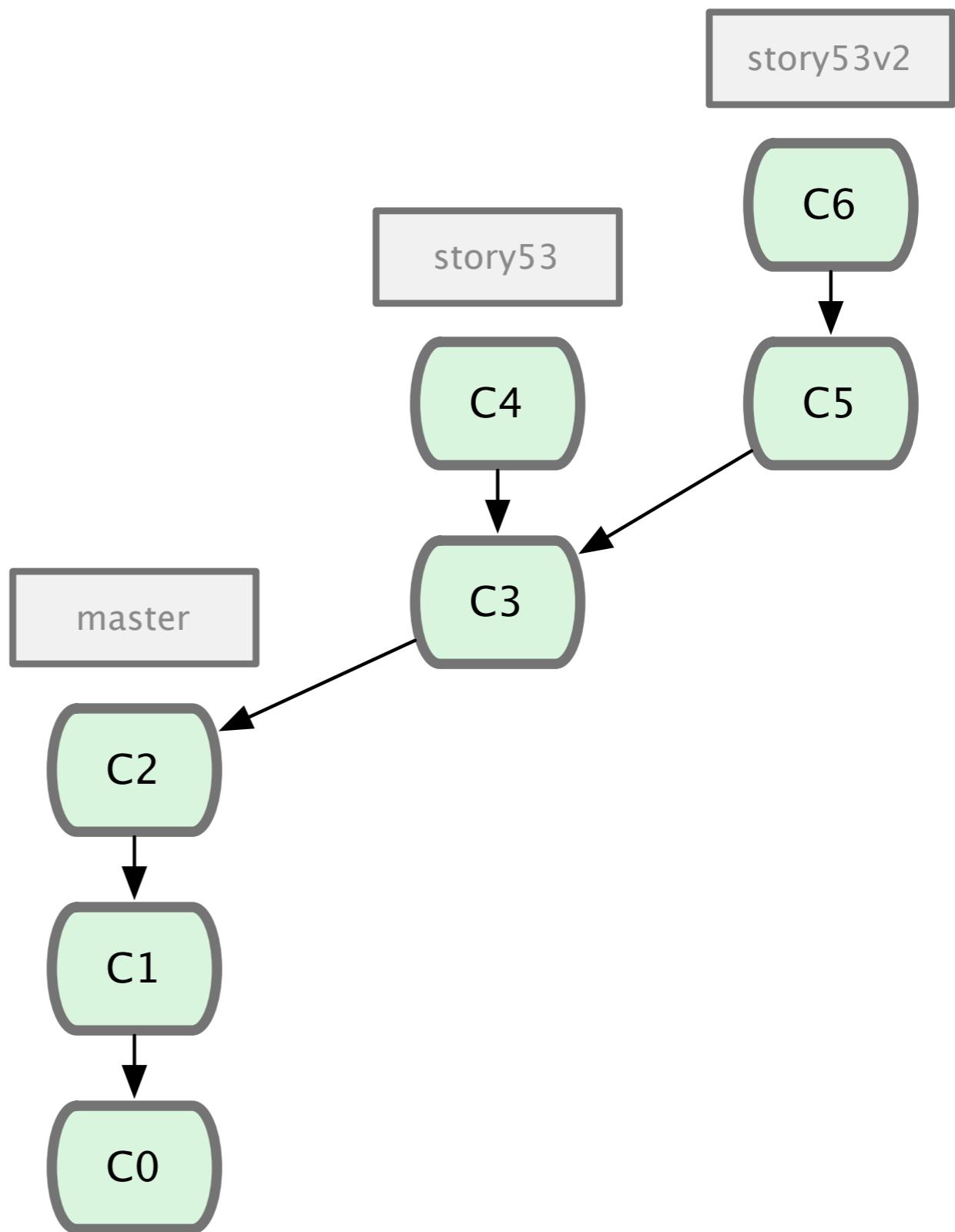
try out new ideas

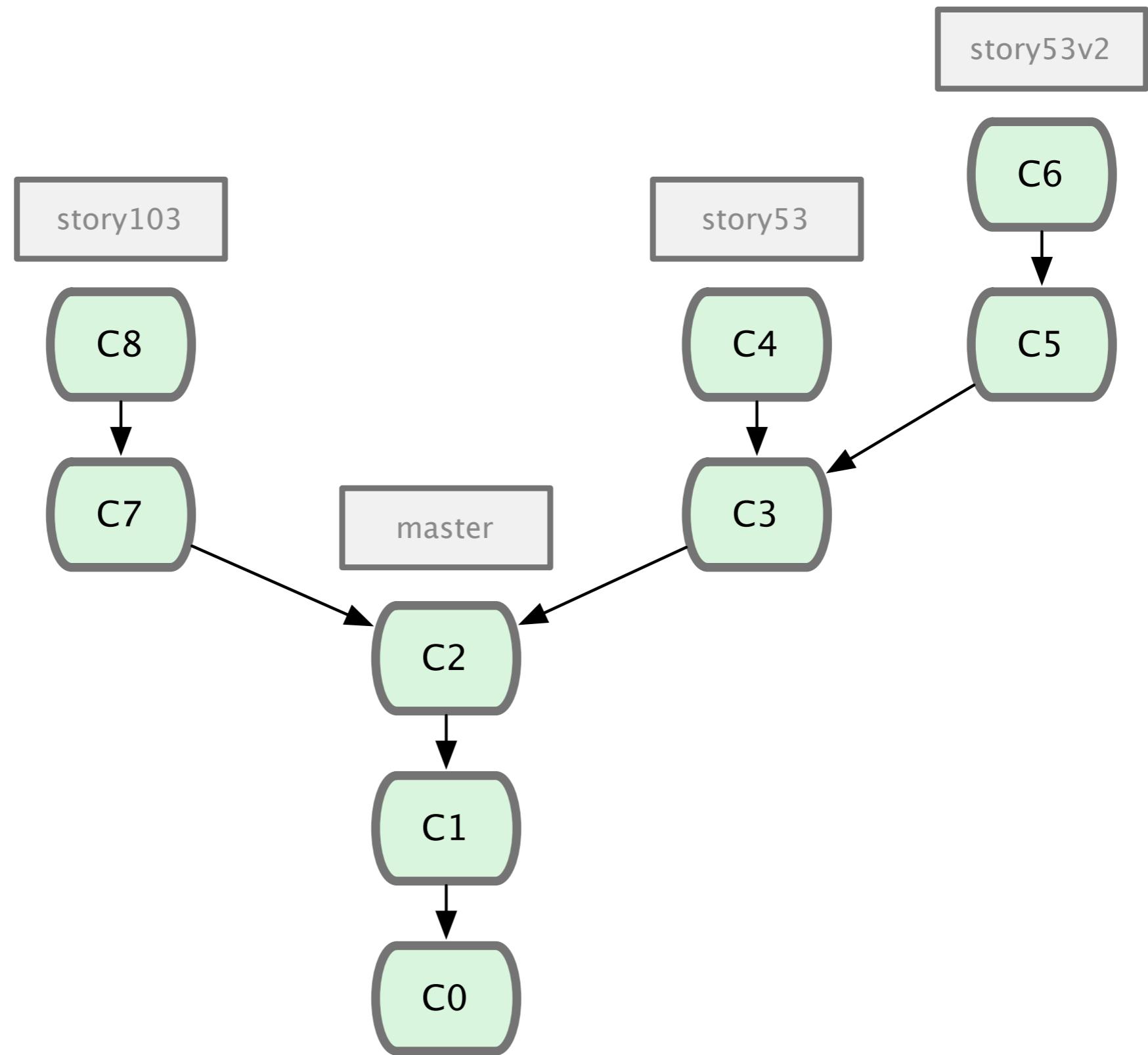
non-linear development

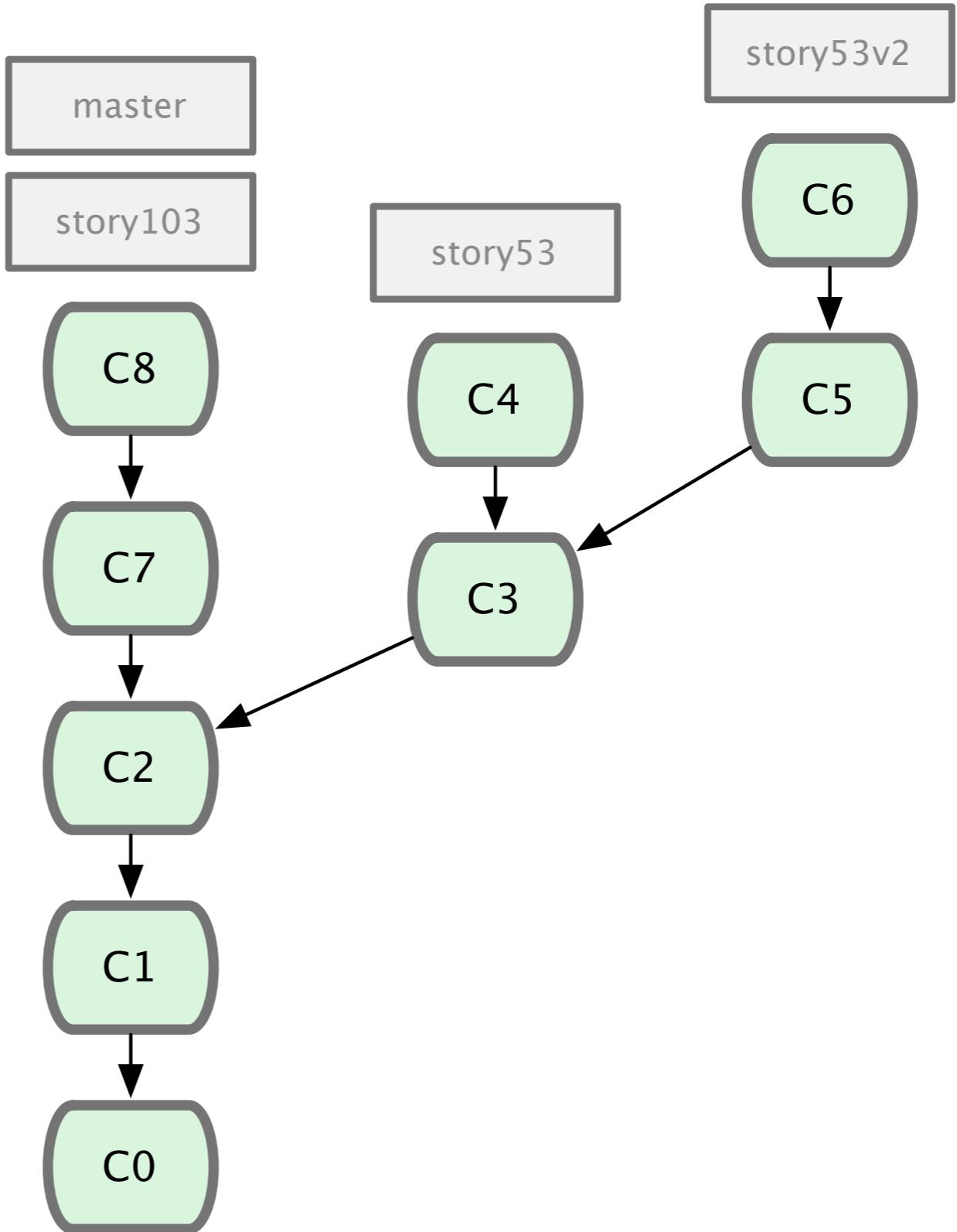


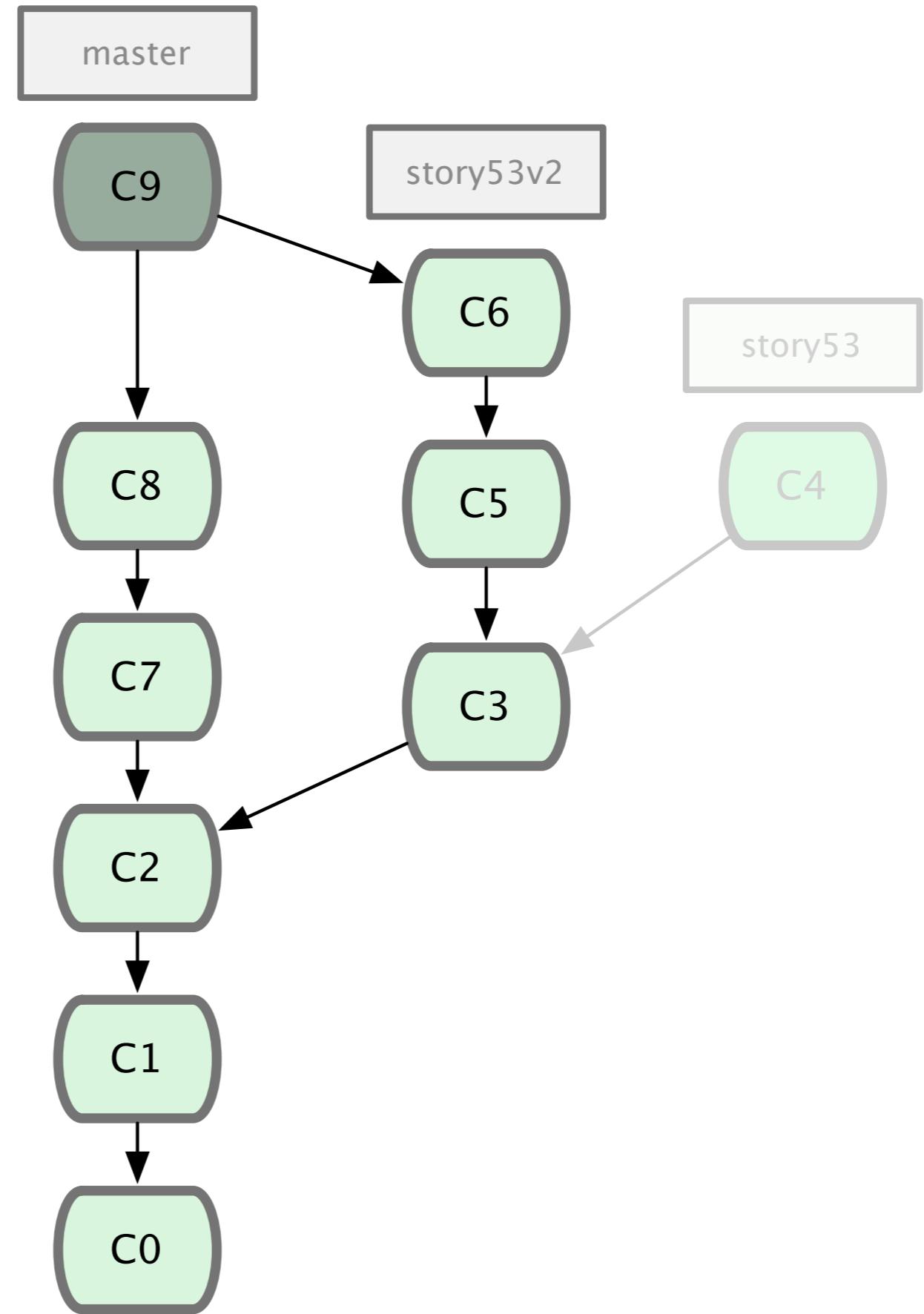




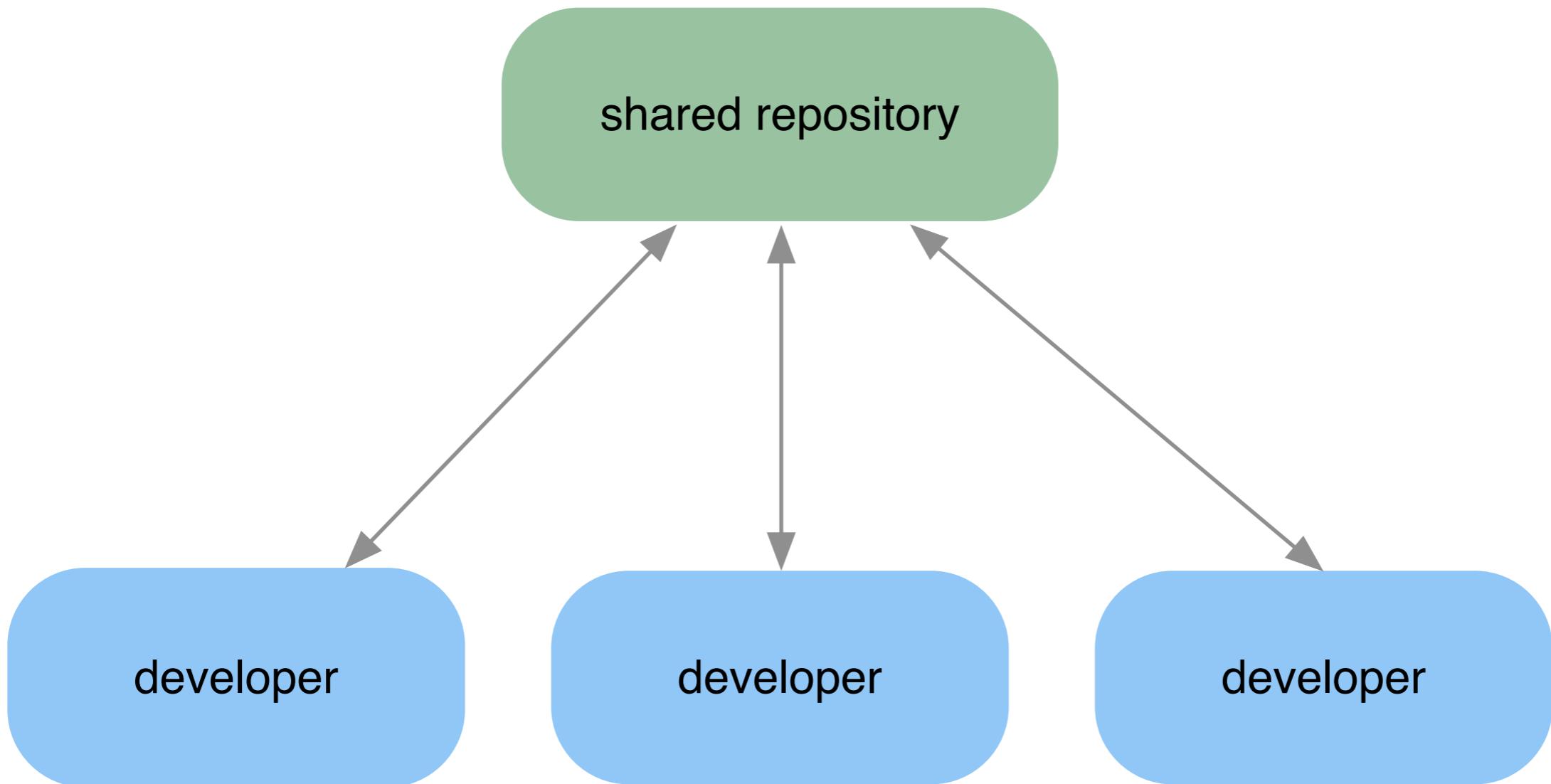


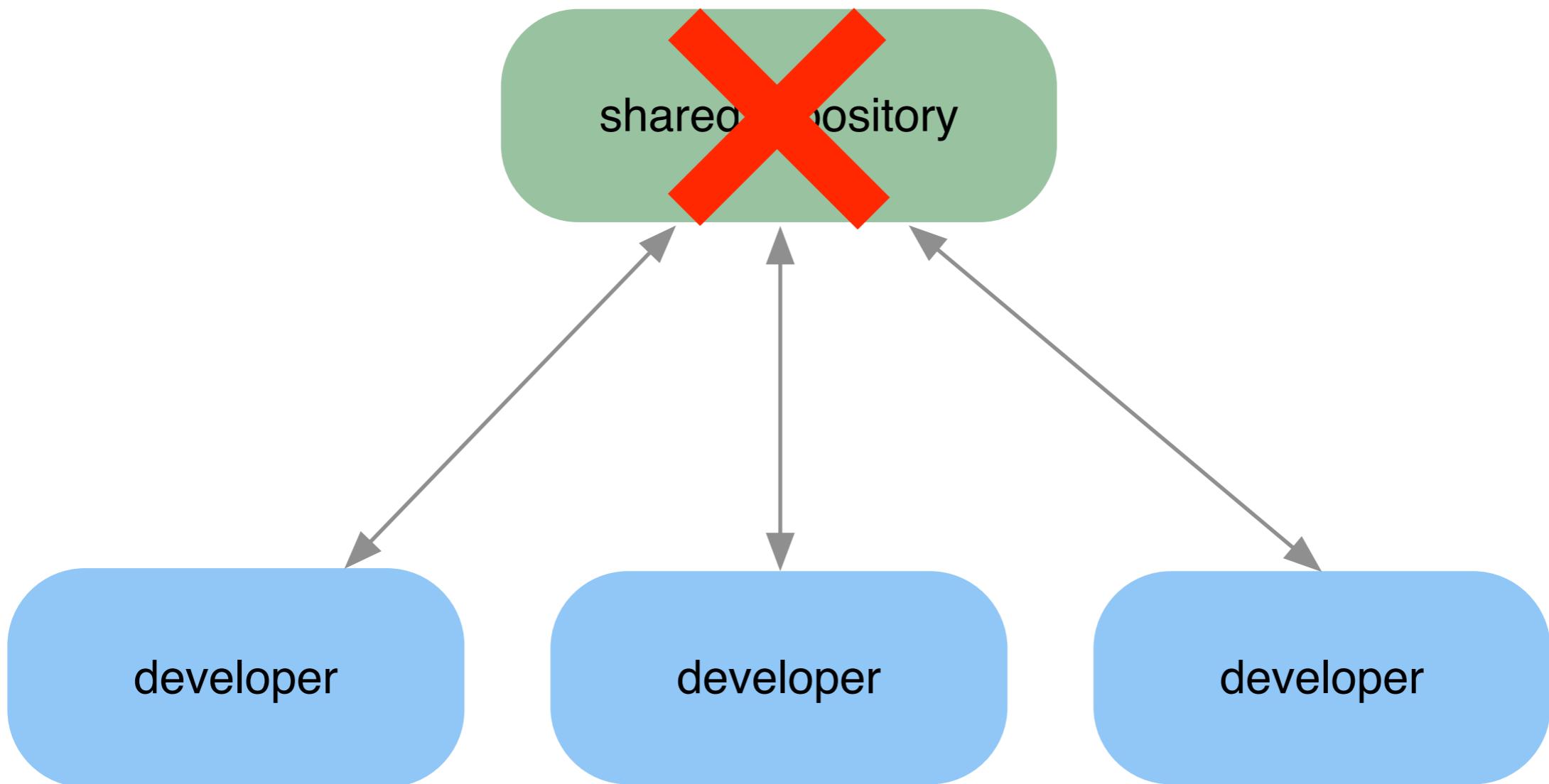


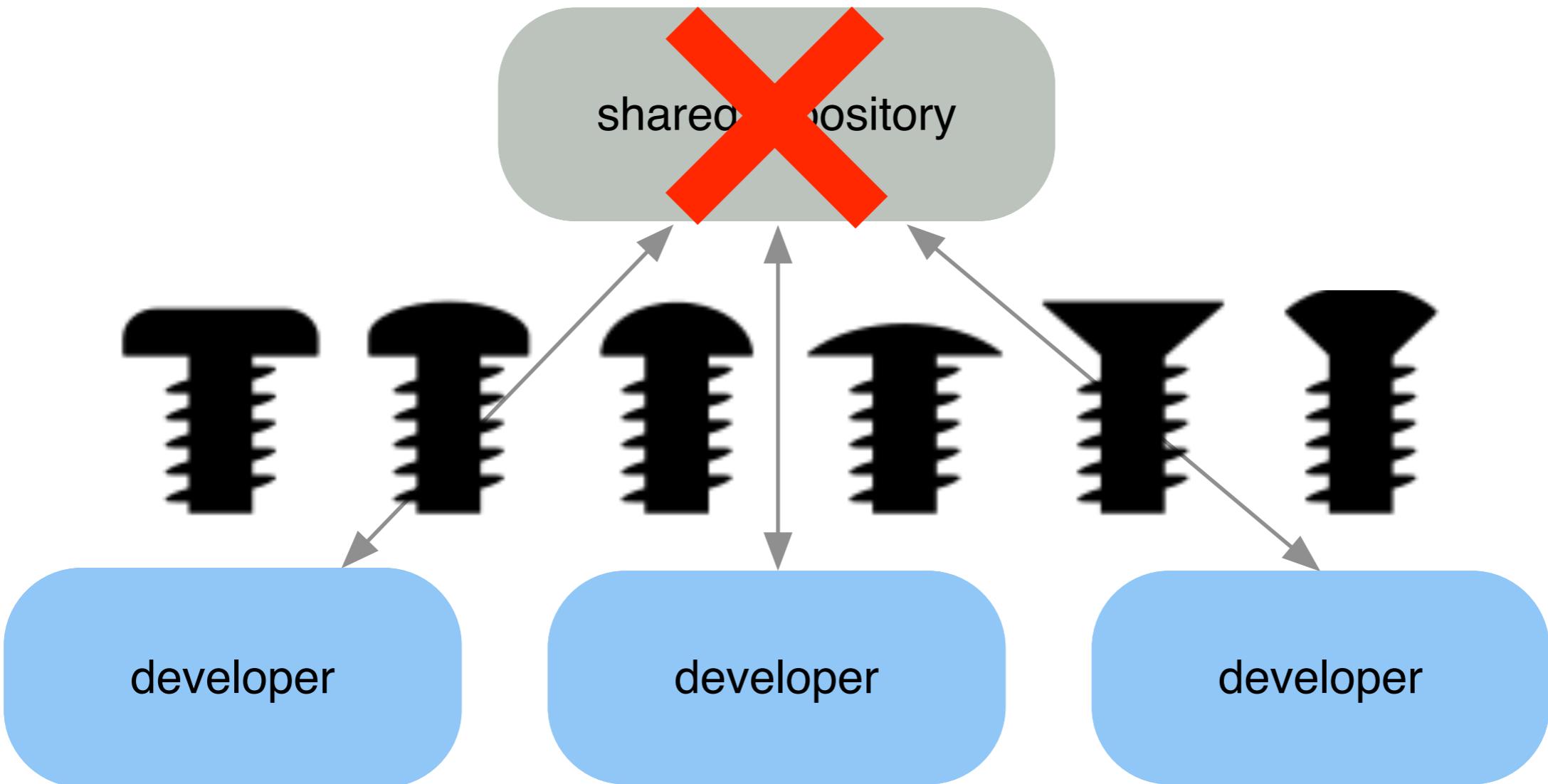




no necessary SPOF







shared repository

A central gray rounded rectangle contains the text "shared repository" in black. A large black X is drawn across the text. Three blue rounded rectangles, each containing the word "developer", are positioned below the central box, connected to it by thin lines.

developer

developer

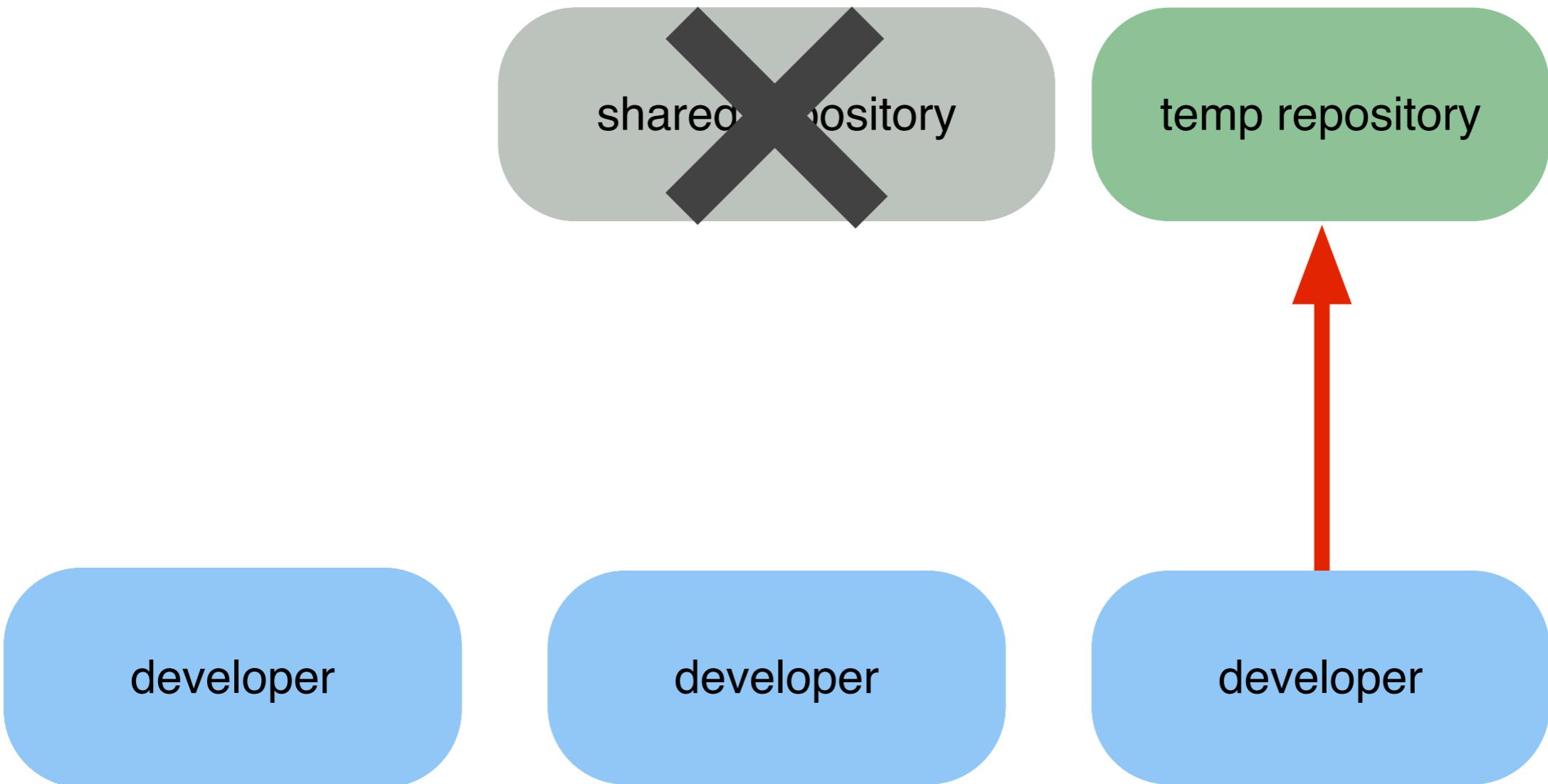
developer

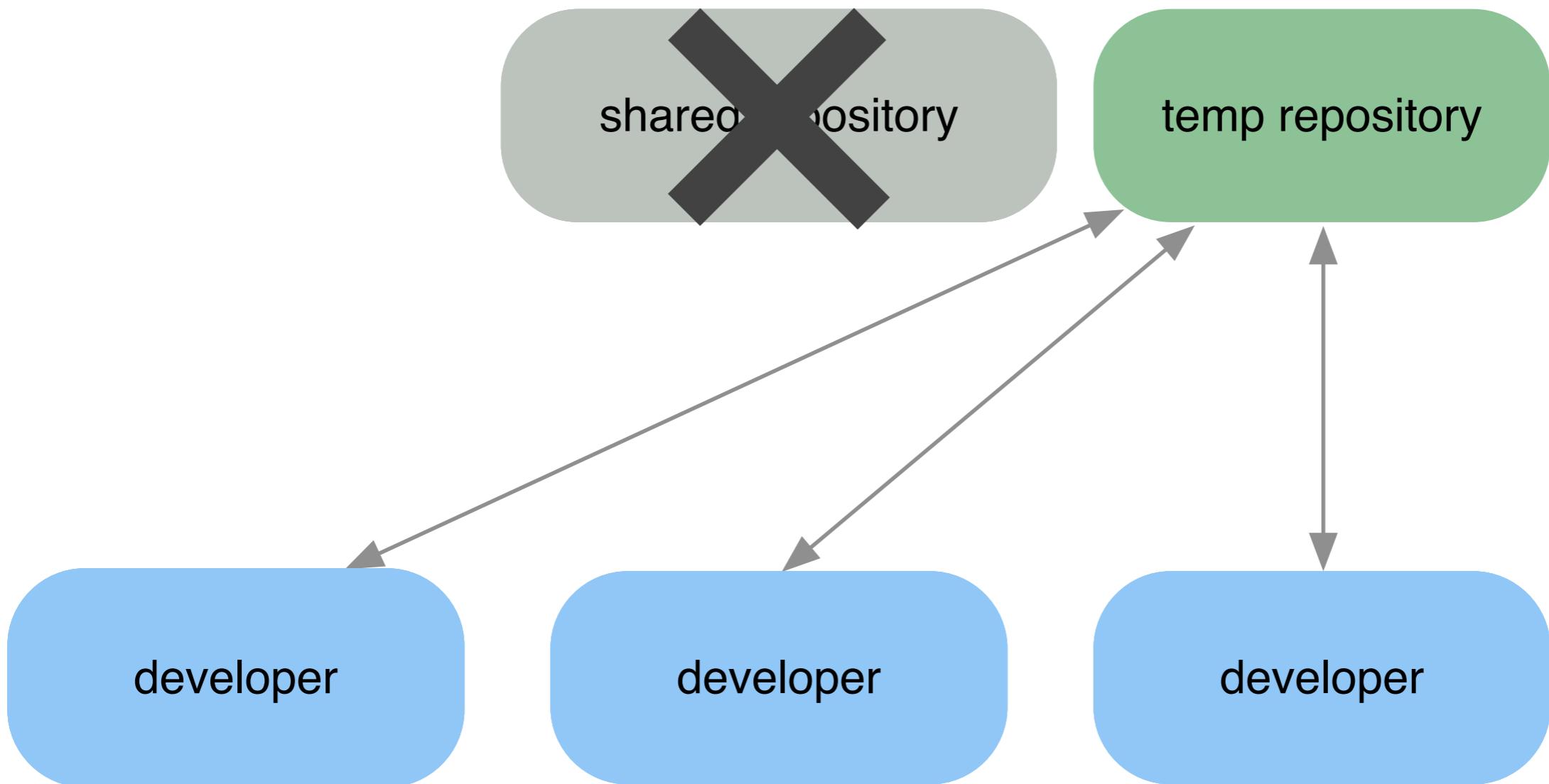
shared repository

developer

developer







How Do I Use Git?

First Steps

```
$ git config --global user.name "Scott Chacon"  
$ git config --global user.email "schacon@gmail.com"
```

Getting A Repo

git init

```
$ touch hello_world.rb
```

```
$ touch hello_world.rb  
$ git init
```

```
$ tree -a  
.  
|-- .git  
|   |-- HEAD  
|   |-- branches  
|   |-- config  
|   |-- description  
|   |-- hooks  
|   |   |-- post-commit.sample  
|   |   |-- post-receive.sample  
|   |   |-- ...  
|   |   |-- pre-rebase.sample  
|   |   `-- update.sample  
|   |-- info  
|   |   `-- exclude  
|   |-- objects  
|   |   |-- info  
|   |   `-- pack  
|   |-- refs  
|   |   |-- heads  
|   |   `-- tags  
|   `-- remotes  
`-- hello_world.rb
```

11 directories, 25 files

```
$ tree -a  
.  
|-- .git  
|   |-- HEAD  
|   |-- branches  
|   |-- config  
|   |-- description  
|   |-- hooks  
|   |   |-- post-commit.sample  
|   |   |-- post-receive.sample  
|   |   |-- ...  
|   |   |-- pre-rebase.sample  
|   |   `-- update.sample  
|   |-- info  
|   |   `-- exclude  
|   |-- objects  
|   |   |-- info  
|   |   `-- pack  
|   |-- refs  
|   |   |-- heads  
|   |   `-- tags  
|   `-- remotes  
`-- hello_world.rb
```

11 directories, 25 files

```
$ tree -a
```

```
.
|--- .git
|   |--- HEAD
|   |--- branches
|   |--- config
|   |--- description
|   |--- hooks
|   |   |--- post-commit.sample
|   |   |--- post-receive.sample
|   |   |--- ...
|   |   |--- pre-rebase.sample
|   |   `--- update.sample
|   |--- info
|   |   `--- exclude
|   |--- objects
|   |   |--- info
|   |   |--- pack
|   |--- refs
|   |   |--- heads
|   |   |--- tags
|   |--- remotes
`--- hello_world.rb
```

11 directories, 25 files

git clone

```
$ git clone
```

```
$ git clone git://github.com/schacon/grit.git
```

```
$ git clone git://github.com/schacon/grit.git mygrit
```

```
$ git clone git://github.com/schacon/grit.git mygrit
Initialized empty Git repository in /home/schacon/mygrit/.git/
remote: Counting objects: 3220, done.
remote: Compressing objects: 100% (2093/2093), done.
remote: Total 3220 (delta 1134), reused 3149 (delta 1081)
Receiving objects: 100% (3220/3220), 1.79 MiB | 357 KiB/s, done.
Resolving deltas: 100% (1134/1134), done.
```

```
$ git clone git://github.com/schacon/grit.git mygrit
Initialized empty Git repository in /home/schacon/mygrit/.git/
remote: Counting objects: 3220, done.
remote: Compressing objects: 100% (2093/2093), done.
remote: Total 3220 (delta 1134), reused 3149 (delta 1081)
Receiving objects: 100% (3220/3220), 1.79 MiB | 357 KiB/s, done.
Resolving deltas: 100% (1134/1134), done.
```

```
$ cd mygrit
$
```

```
$ git clone git://github.com/schacon/grit.git mygrit
Initialized empty Git repository in /home/schacon/mygrit/.git/
remote: Counting objects: 3220, done.
remote: Compressing objects: 100% (2093/2093), done.
remote: Total 3220 (delta 1134), reused 3149 (delta 1081)
Receiving objects: 100% (3220/3220), 1.79 MiB | 357 KiB/s, done.
Resolving deltas: 100% (1134/1134), done.
```

```
$ cd mygrit
$ ls
API.txt      Manifest.txt README.txt benchmarks.rb examples lib
History.txt   PURE_TODO Rakefile benchmarks.txt grit.gemspec test
$
```

A Basic Workflow

A Basic Workflow

Edit files

Stage the changes

Review your changes

Commit the changes

working directory

index

repository

working directory

a working copy
of your project

index

repository

working directory

index

object database

repository

working directory

index

“staging”

repository

A Basic Workflow

Edit files

Stage the changes

Review your changes

Commit the changes

\$

\$ find .

```
$ find .  
./app.yaml  
./index.yaml  
./main.py
```

\$

```
$ vim main.py
```

```
$ vim main.py
```

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hello world!')

    def main():
        application = webapp.WSGIApplication([(('/', MainHandler)],
                                              debug=True)
        wsgiref.handlers.CGIHandler().run(application)

if __name__ == '__main__':
    main()
~
~
"main.py" 16L, 402C
```

```
$ vim main.py
```

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

# this program prints out 'hello world'

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hello world!')

    def main():
        application = webapp.WSGIApplication([(('/', MainHandler)],
                                              debug=True)
        wsgiref.handlers.CGIHandler().run(application)

    if __name__ == '__main__':
        main()
~  
~  
"main.py" 16L, 402C
```

```
$ git status
```

```
$ git status
```

```
# On branch master
```

```
# Changed but not updated:
```

```
#   (use "git add <file>..." to update what
```

```
#
```

```
# modified:    main.py
```

```
#
```

```
no changes added to commit (use "git add" and
```

```
$ git status
```

```
# On branch master
```

```
# Changed but not updated:
```

```
#   (use "git add <file>..." to update what
```

```
#
```

```
# modified:    main.py
```

```
#
```

```
no changes added to commit (use "git add" and
```

```
$ git status
```

```
# On branch master
```

```
# Changed but not updated:
```

```
#   (use "git add <file>..." to update what
```

```
#
```

```
# modified:    main.py
```

```
#
```

```
no changes added to commit (use "git add" and
```

```
$ git status
```

```
# On branch master
```

```
# Changed but not STAGED:
```

```
#   (use "git add <file>..." to update what
```

```
#
```

```
# modified:    main.py
```

```
#
```

```
no changes added to commit (use "git add" and
```

A Basic Workflow

Edit files

Stage the changes

Review your changes

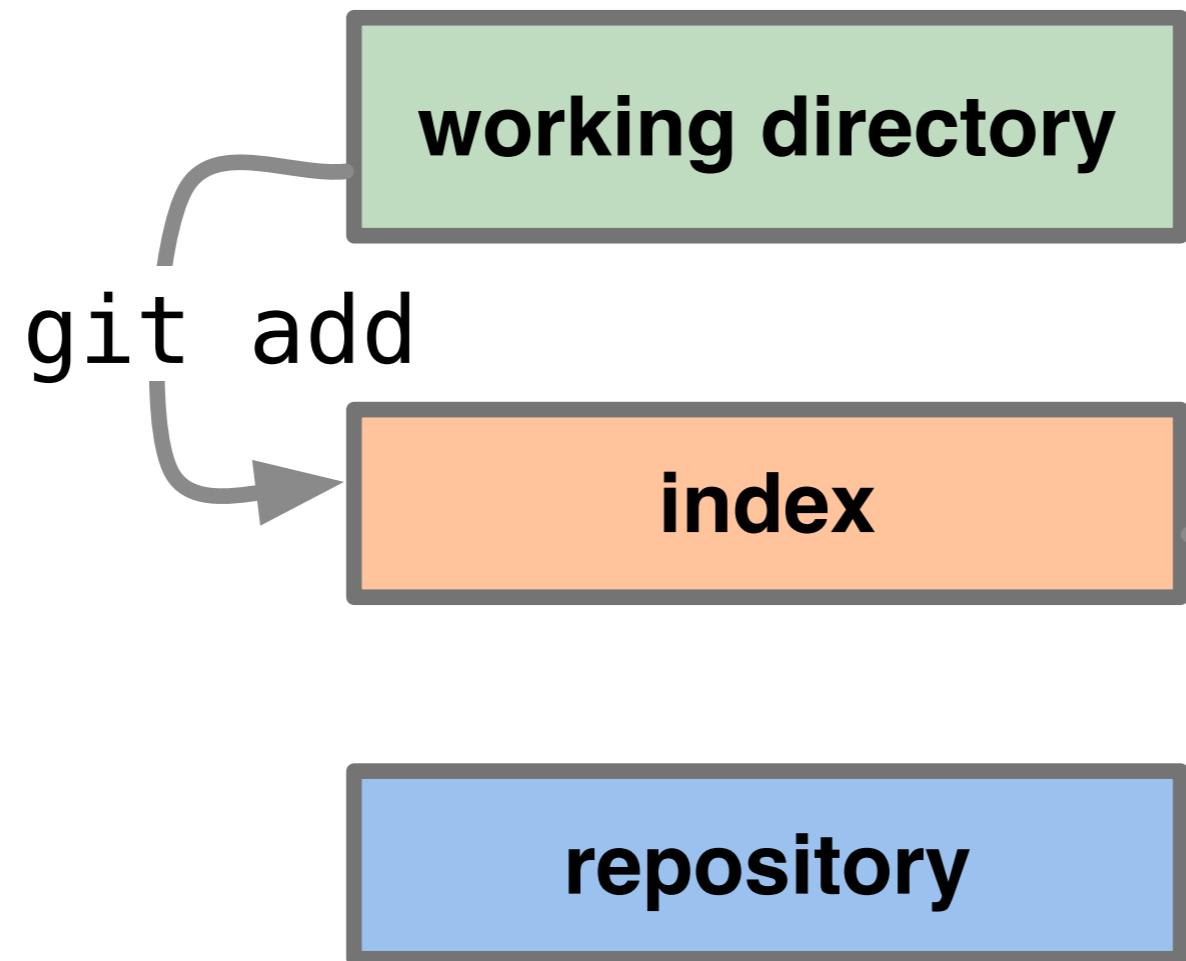
Commit the changes

git add

working directory

index

repository



\$

```
$ git add main.py
```

```
$ git add main.py
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>
#
# modified:    main.py
#
```

```
$ git add main.py
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>
#
# modified:    main.py
#
```

```
$ git add main.py
$ git status
# On branch master
# Changes THAT ARE STAGED :
#   (use "git reset HEAD <file>
#
# modified:    main.py
#
```

\$

```
$ vim app.yaml
```

```
$ vim app.yaml
```

```
application: chacon
version: 1
runtime: python
api_version: 1
```

```
handlers:
- url: .*
  script: main.py
```

```
~
```

```
~
```

```
~
```

```
"app.yaml" 8L, 101C
```

```
$ vim app.yaml
```

```
application: chacon
version: █
runtime: python
api_version: 1
```

```
handlers:
- url: .*
  script: main.py
```

```
~
```

```
~
```

```
~
```

```
"app.yaml" 8L, 101C
```

```
$ vim app.yaml
```

```
application: chacon
version: 2
runtime: python
api_version: 1
```

```
handlers:
- url: .*
  script: main.py
```

```
~
```

```
~
```

```
~
```

```
"app.yaml" 8L, 101C
```

\$

```
$ git status
```

\$ git status

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:   main.py
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be com
#
# modified:   app.yaml
#
```

\$

```
$ vim main.py
```

```
$ vim main.py
```

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

# this program prints out 'hello world'

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hello world!')

def main():
    application = webapp.WSGIApplication(['/'], MainHandler),
                                         debug=True)
    wsgiref.handlers.CGIHandler().run(application)

if __name__ == '__main__':
    main()
~
```

```
$ vim main.py
```

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

# this program prints out 'hello world'

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hola █ orld!')

def main():
    application = webapp.WSGIApplication([('/', MainHandler)],
                                          debug=True)
    wsgiref.handlers.CGIHandler().run(application)

if __name__ == '__main__':
    main()
~
~
"main.py" 16L, 402C
```

```
$ vim main.py
```

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

# this program prints out 'hello world'

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hola Mundo')

def main():
    application = webapp.WSGIApplication(['/'], MainHandler),
                                         debug=True)
    wsgiref.handlers.CGIHandler().run(application)

if __name__ == '__main__':
    main()
~  
~  
"main.py" 16L, 402C
```

\$

```
$ git status
```

```
$ git status
```

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:   main.py
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be com
#
# modified:   app.yaml
# modified:   main.py
#
```

\$ git status

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:   main.py
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be com
#
# modified:   app.yaml
# modified:   main.py
#
```

Staged



\$ git status

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:   main.py
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be com
#
# modified:   app.yaml
# modified:   main.py
#
```

Unstaged



\$ git status

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   main.py
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be com
#
#       modified:   app.yaml
#       modified:   main.py
#
```



Staged

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

# this program prints out 'hello world'

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hello World!')

def main():
    application =
webapp.WSGIApplication([(' '/',
debug=True)

wsgiref.handlers.CGIHandler().run(applicat

if __name__ == '__main__':
    main()
```

In Working Directory

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

# this program prints out 'hello world'

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hola Mundo!')

def main():
    application =
webapp.WSGIApplication([(' '/',
debug=True)

wsgiref.handlers.CGIHandler().run(applicat

if __name__ == '__main__':
    main()
```

Staged

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

# this program prints out 'hello world'

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hello World!')

    def main():
        application =
webapp.WSGIApplication([('/' debug=True)

wsgiref.handlers.CGIHandler().run(application)

if __name__ == '__main__':
    main()
```

In Working Directory

```
#!/usr/bin/env python
import wsgiref.handlers
from google.appengine.ext import webapp

# this program prints out 'hello world'

class MainHandler(webapp.RequestHandler):

    def get(self):
        self.response.out.write('Hola Mundo!')

    def main():
        application =
webapp.WSGIApplication([('/' debug=True)

wsgiref.handlers.CGIHandler().run(application)

if __name__ == '__main__':
    main()
```

You have to stage a file
after you edit it

You have to stage a file
after you edit it

You have to stage a file
after you edit it

\$

```
$ git add app.yaml main.py
```

```
$ git add app.yaml main.py
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:    app.yaml
# modified:    main.py
#
```

interesting tidbit

git add -p

A Basic Workflow

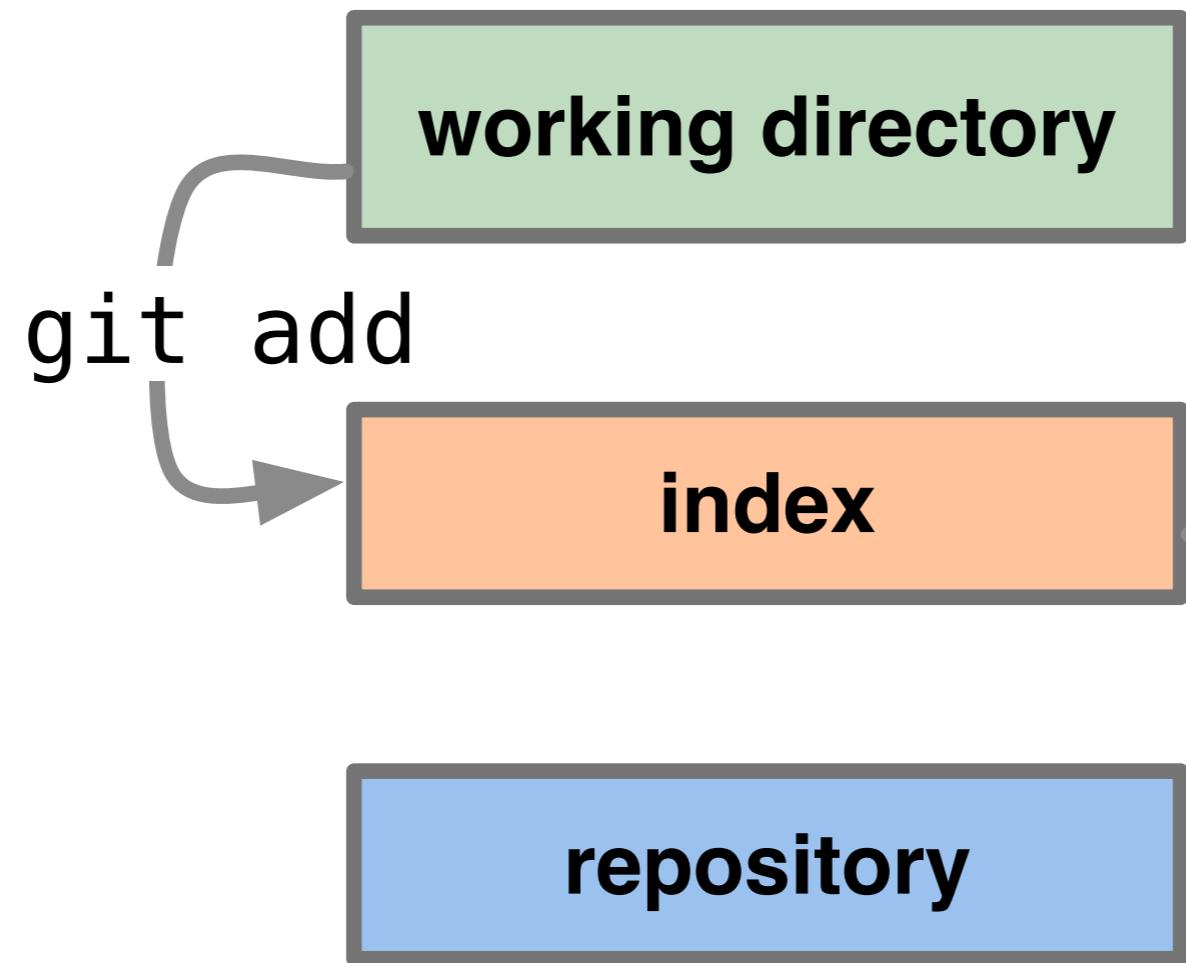
Edit files

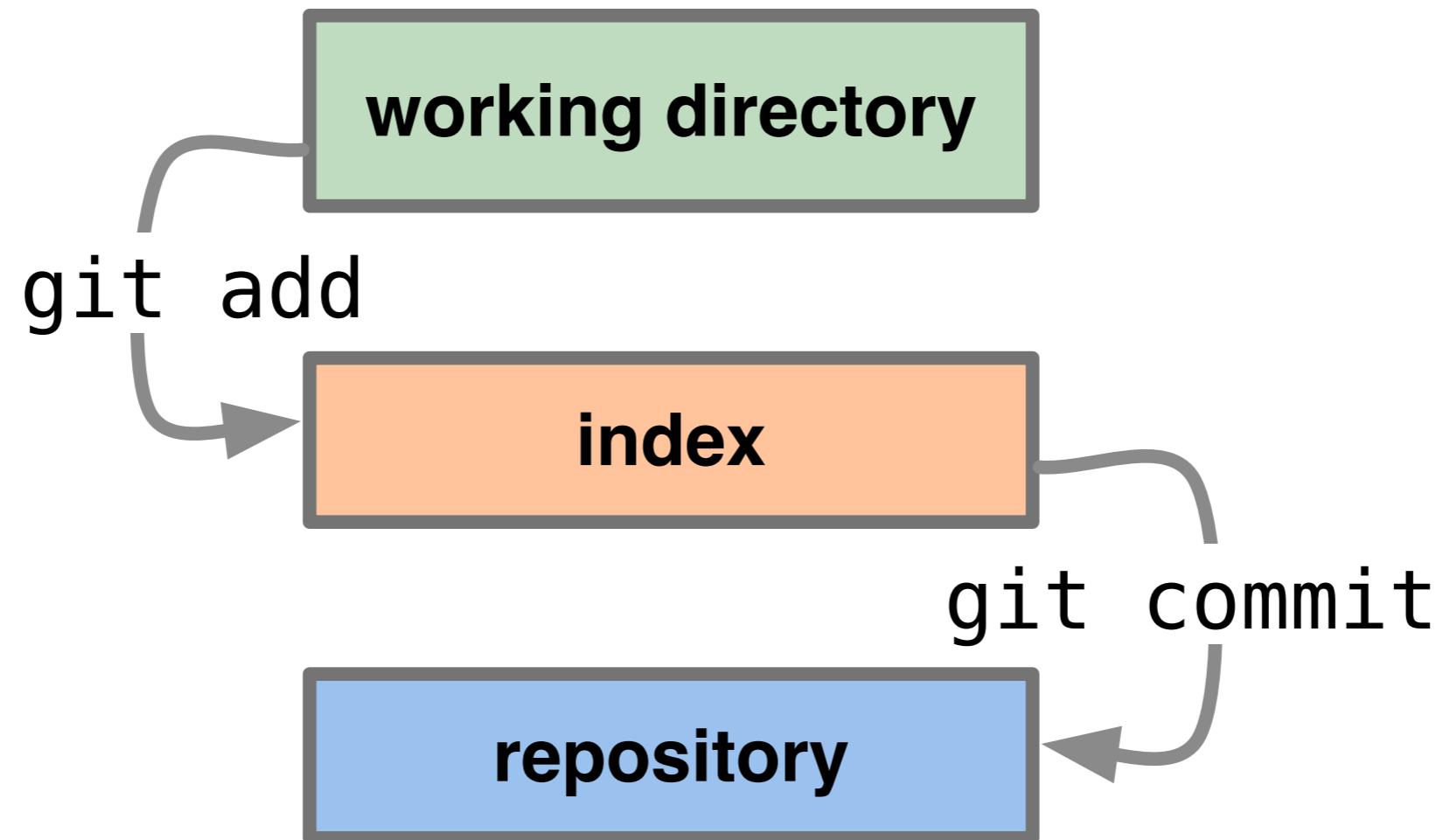
Stage the changes

Review your changes

Commit the changes

git commit





\$

```
$ git commit
```

```
$ git commit
```

\$ git commit

```
descriptive commit message █  
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the commit.  
# On branch master  
# Changes to be committed:  
#   (use "git reset HEAD <file>..." to unstage)  
#  
#       modified:   app.yaml  
#       modified:   main.py  
#  
~  
~  
~  
~  
~  
".git/COMMIT_EDITMSG" 10L, 279C
```

```
$ git commit
```

```
Created commit 77d3001: descriptive commit message  
2 files changed, 4 insertions(+), 2 deletions(-)
```

A Basic Workflow

A Basic Workflow

Edit files

vim / emacs / etc

A Basic Workflow

Edit files

`vim / emacs / etc`

Stage the changes

`git add (file)`

A Basic Workflow

Edit files

`vim / emacs / etc`

Stage the changes

`git add (file)`

Review your changes

`git status`

A Basic Workflow

Edit files

`vim / emacs / etc`

Stage the changes

`git add (file)`

Review your changes

`git status`

Commit the changes

`git commit`

cheating...

A Basic Workflow

A Basic Editor Workflow

Edit files

`vim / emacs / etc`

A Basic Git Workflow

Edit files

`vim / emacs / etc`

Stage & commit the changes

`git commit -a`

What's going on here?

```
$ git commit
```

```
Created commit 77d3001: descriptive commit message  
2 files changed, 4 insertions(+), 2 deletions(-)
```

```
$ git commit
```

```
Created commit 77d3001: descriptive commit message  
2 files changed, 4 insertions(+), 2 deletions(-)
```

77d3001

77d3001

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

```
tree c4ec543b0322744e55c5efc9b6c4e449d398dbff
parent a149e2160b3f7573768cdc2fce24d0881f3577e1
author Scott Chacon <schacon@gmail.com> 1223402504 -0700
committer Scott Chacon <schacon@gmail.com> 1223402504 -0700
```

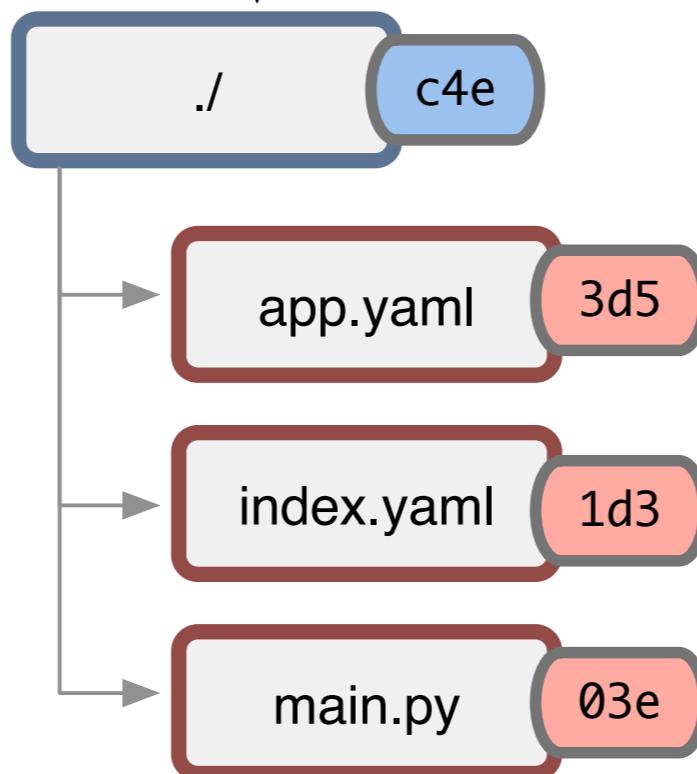
descriptive commit message

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

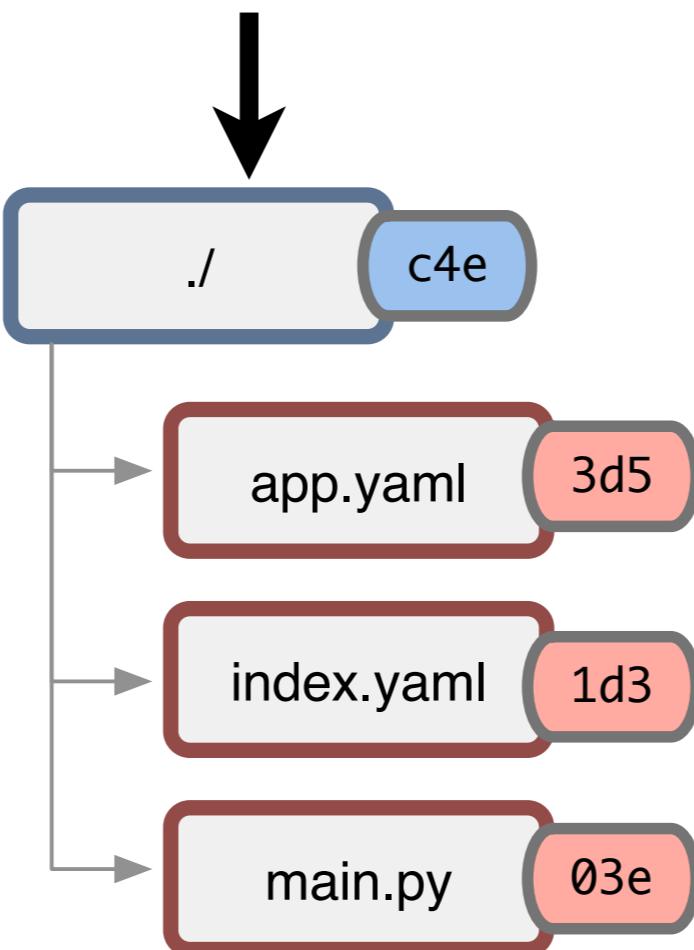
commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	



commit	size
tree	2de54
parent	38def
author	Scott
committer	Scott
this is the previous commit and I am very proud of it	

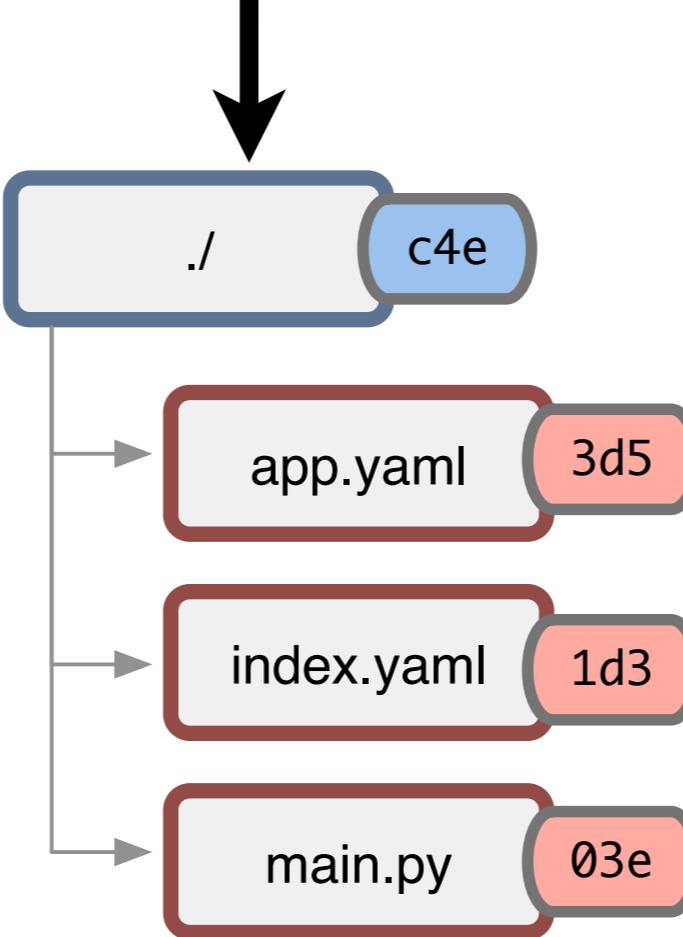
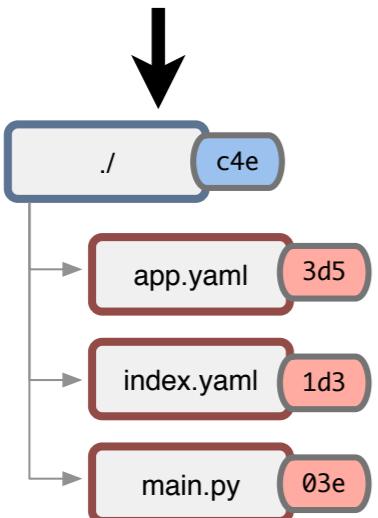
←

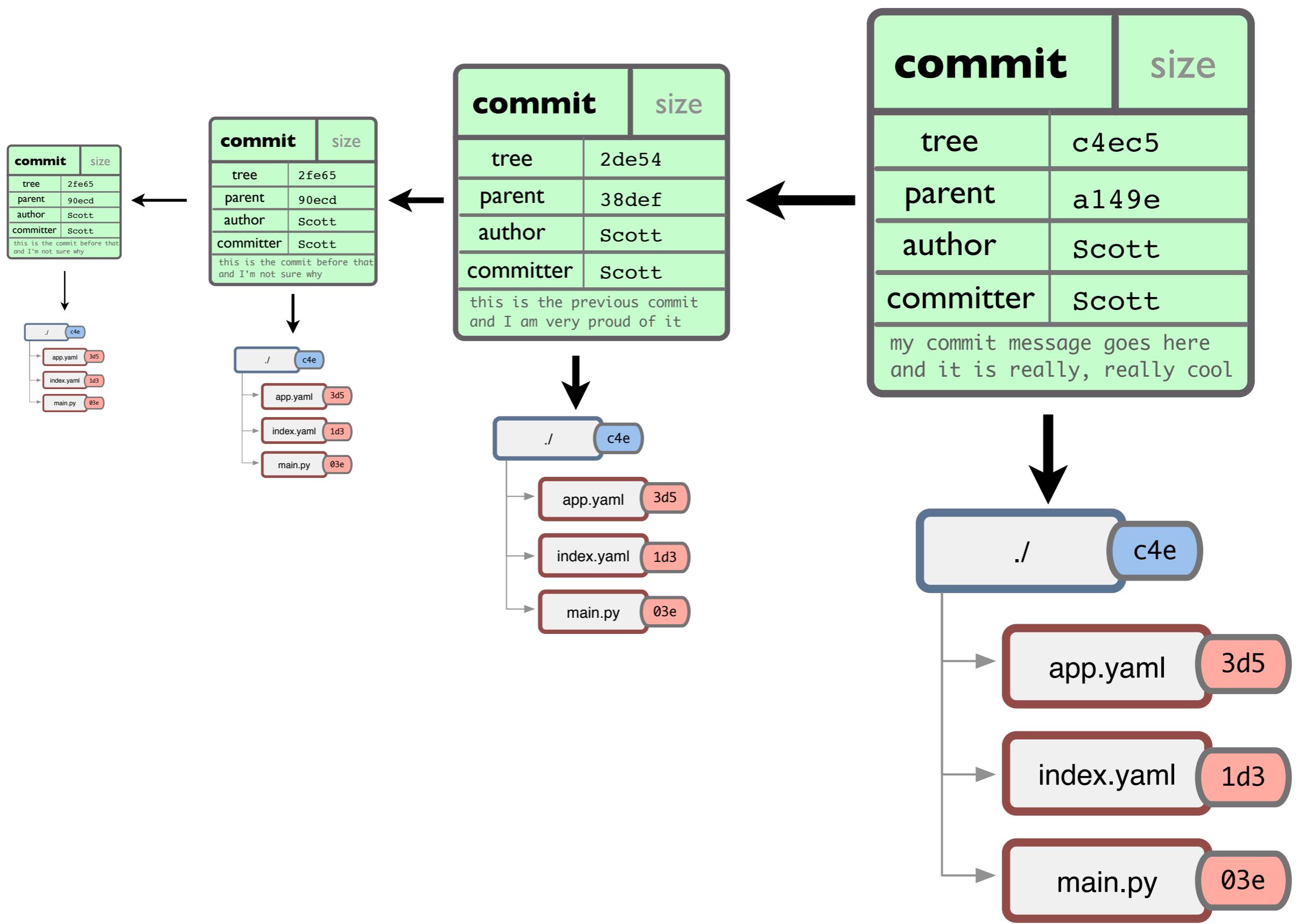
commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	

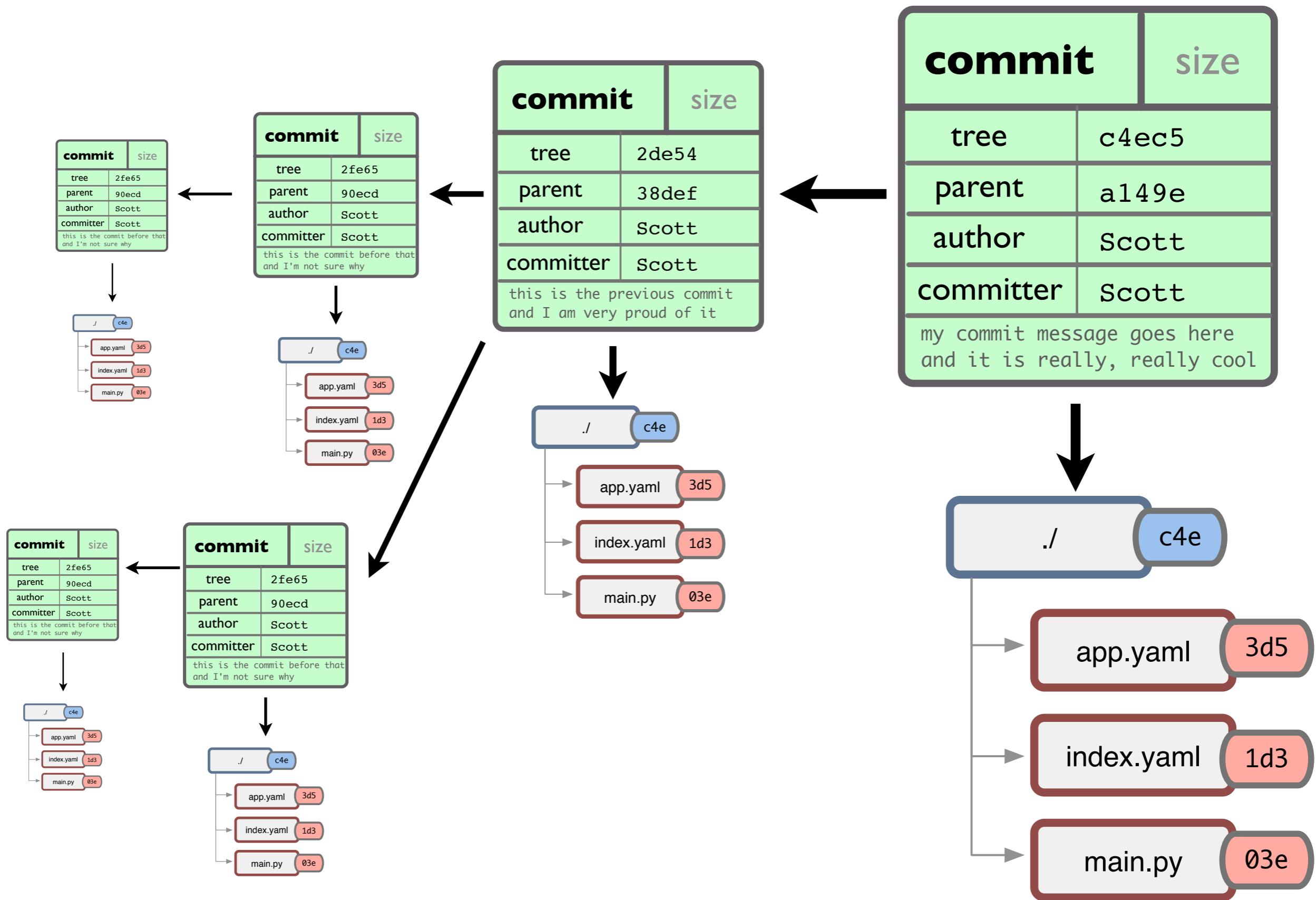


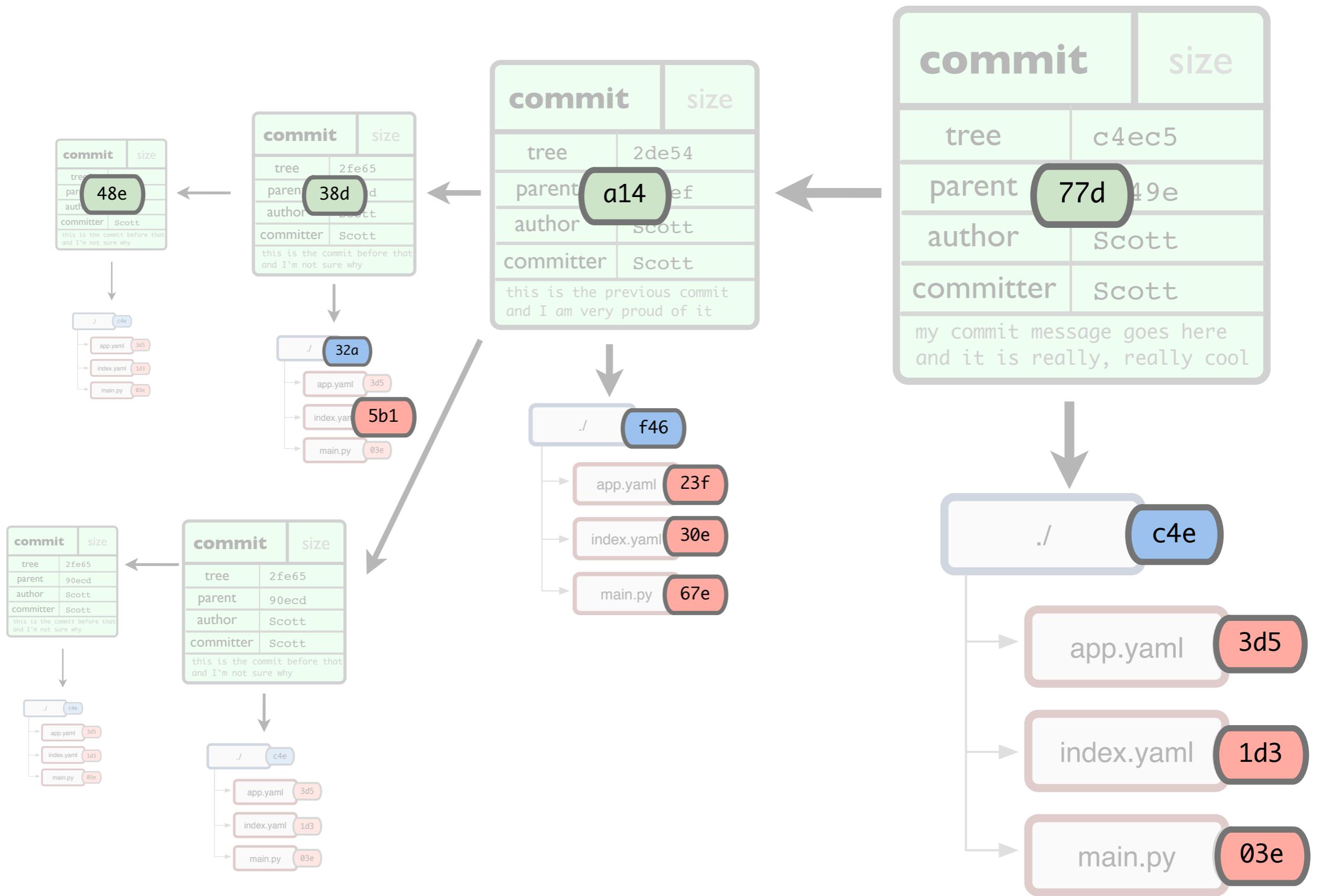
commit	
size	
tree	2de54
parent	38def
author	Scott
committer	Scott
this is the previous commit and I am very proud of it	

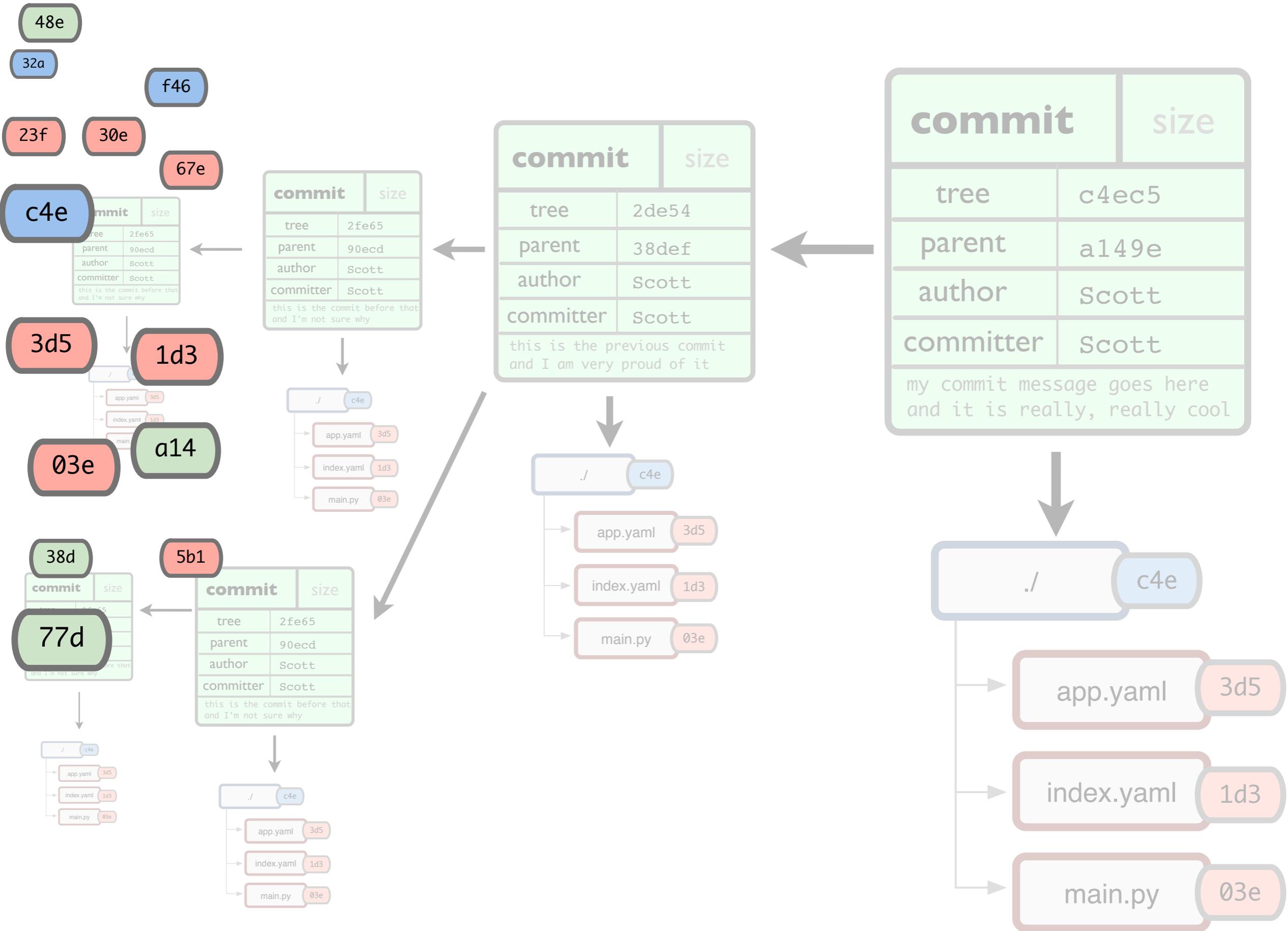
commit	
size	
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	











Repository

5b1	32a	c36
1d3	f46	3d4
ffe	6fe	ae6
38d	23f	03e
254	30e	5b1
a14	67e	1d3
3d5	735	d23
c4e	c4e	48e
77d	de3	2d3

Repository

5b1	32a	c36
1d3	f46	3d4
ffe	6fe	ae6
38d	23f	03e
254	30e	5b1
a14	67e	1d3
3d5	735	d23
c4e	c4e	48e
77d	de3	2d3

git checkout ae635

Repository

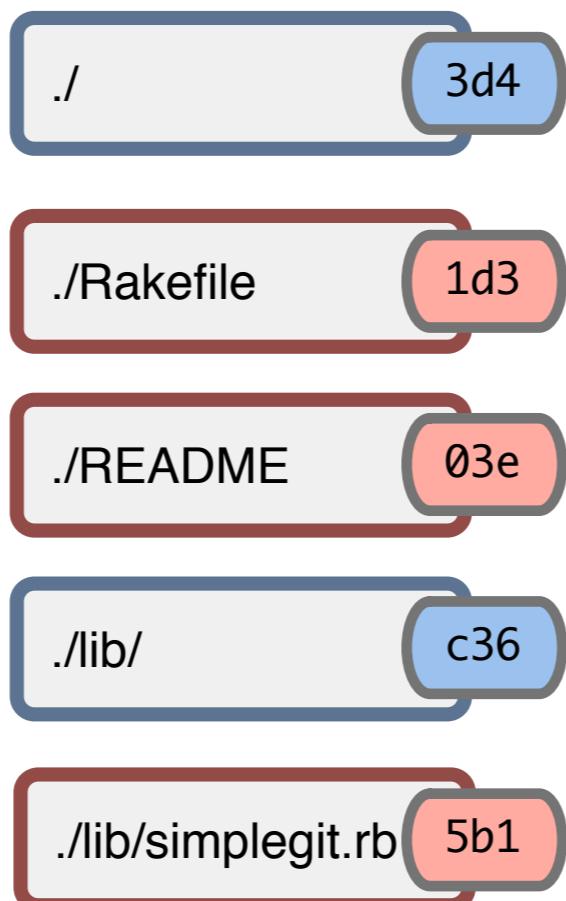
5b1	32a	c36
1d3	f46	3d4
ffe	6fe	ae6
38d	23f	03e
254	30e	5b1
a14	67e	1d3
3d5	735	d23
c4e	c4e	48e
77d	de3	2d3

git checkout ae635

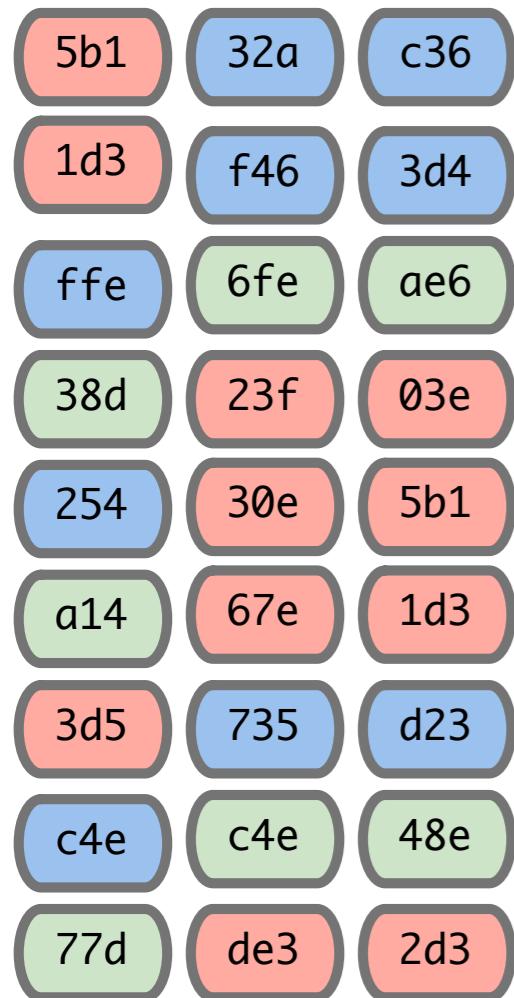
Repository



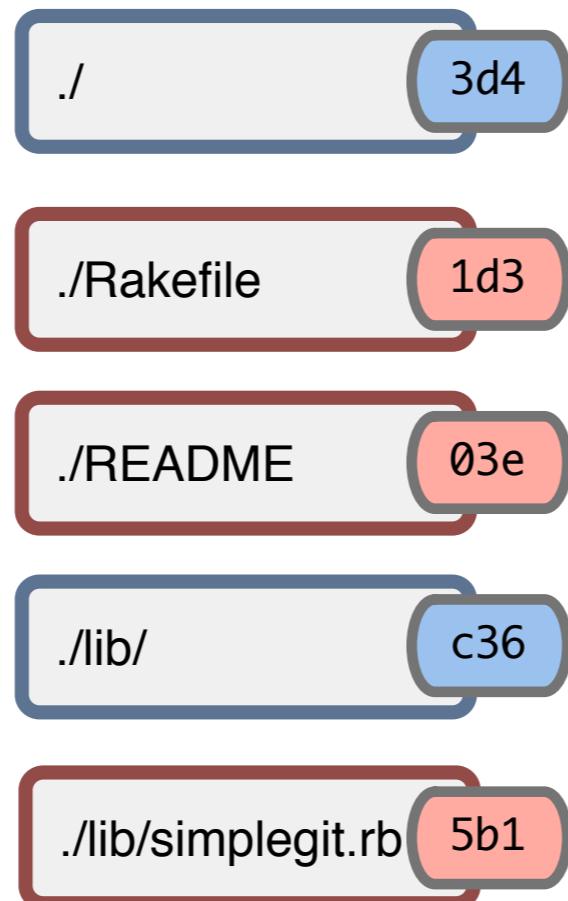
Index



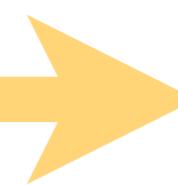
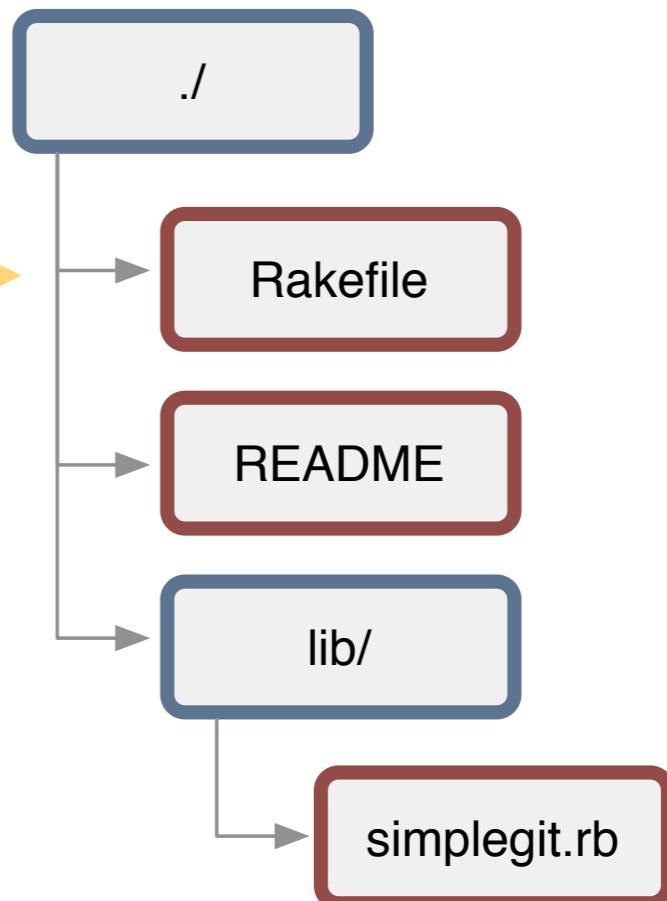
Repository



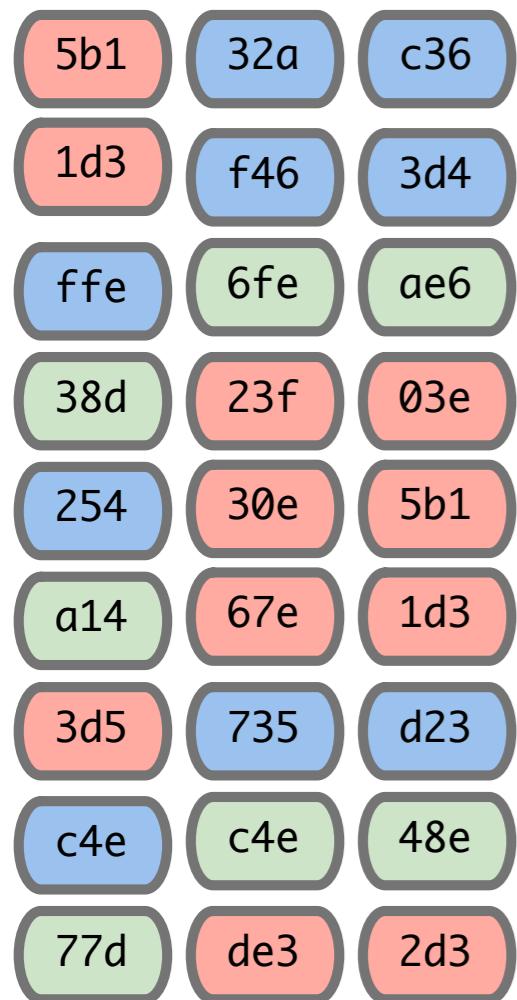
Index



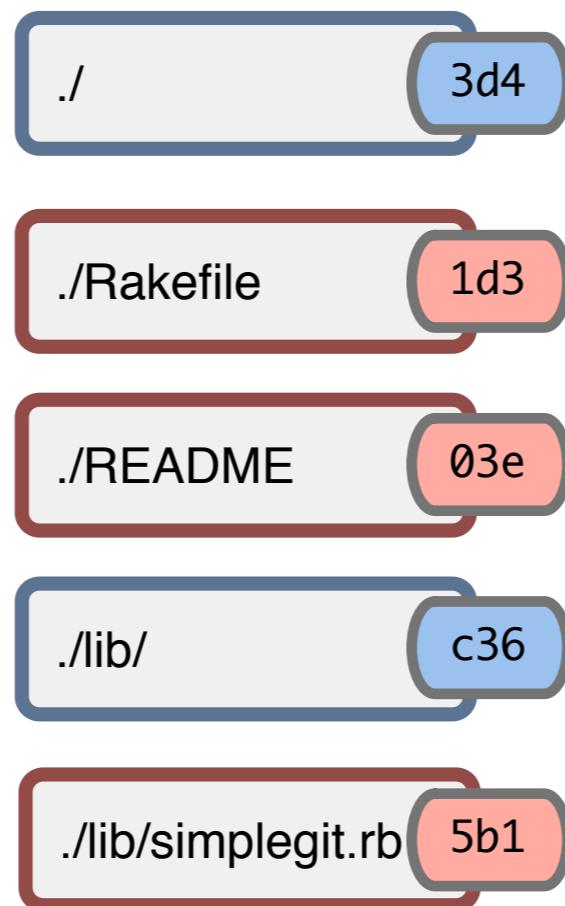
Working Directory



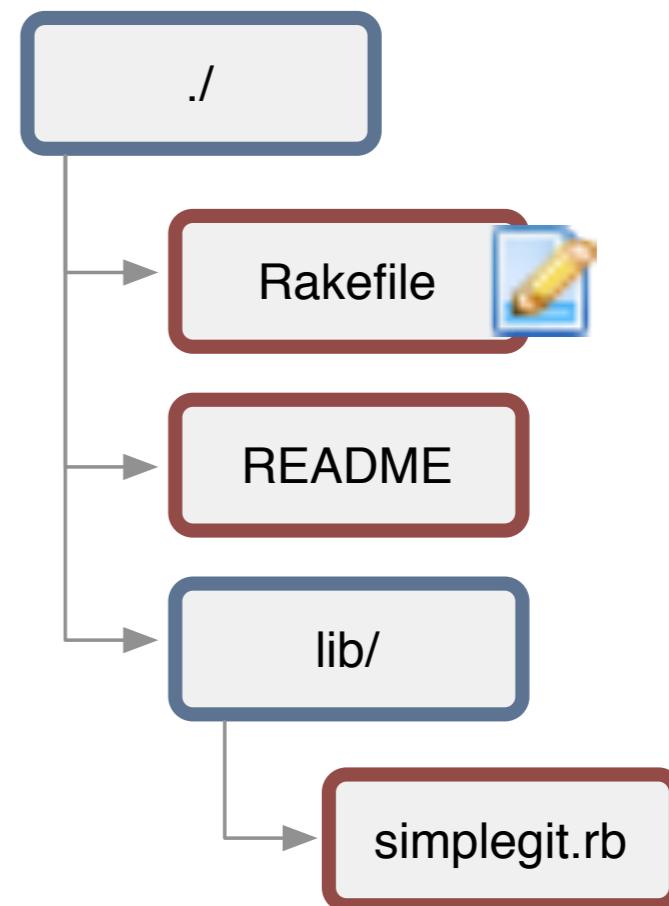
Repository



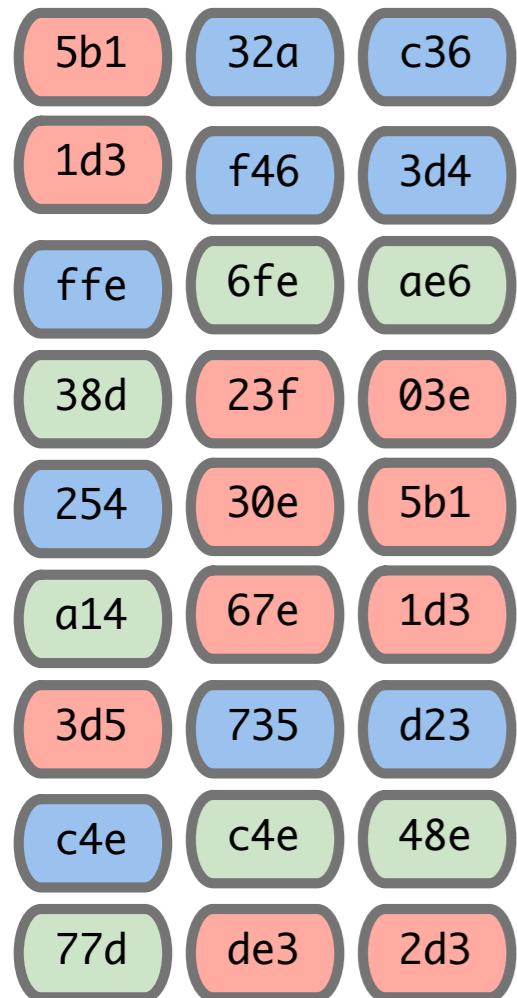
Index



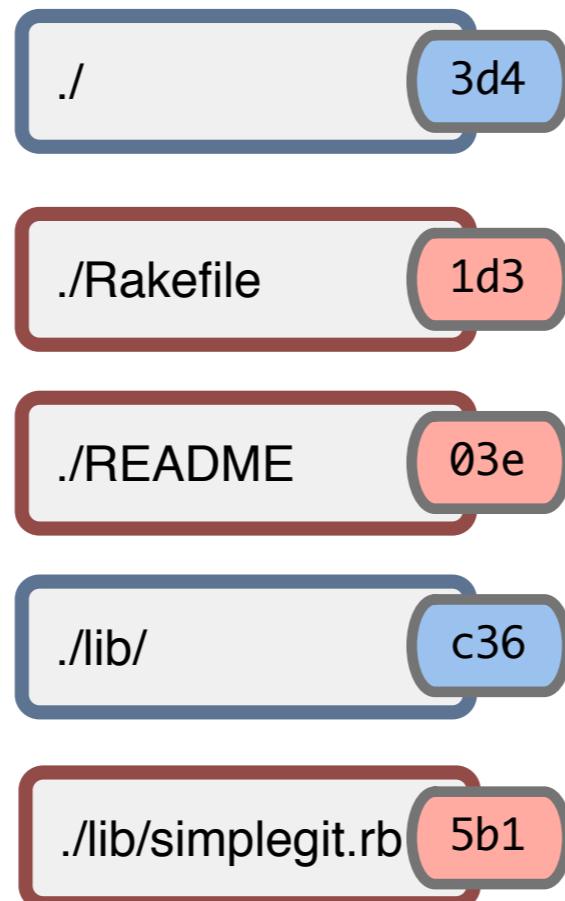
Working Directory



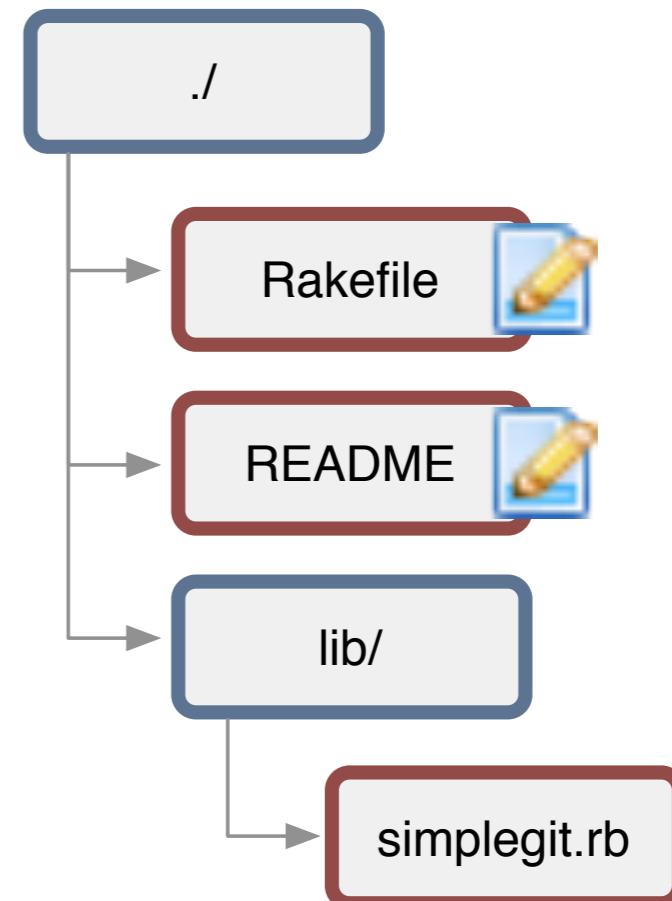
Repository



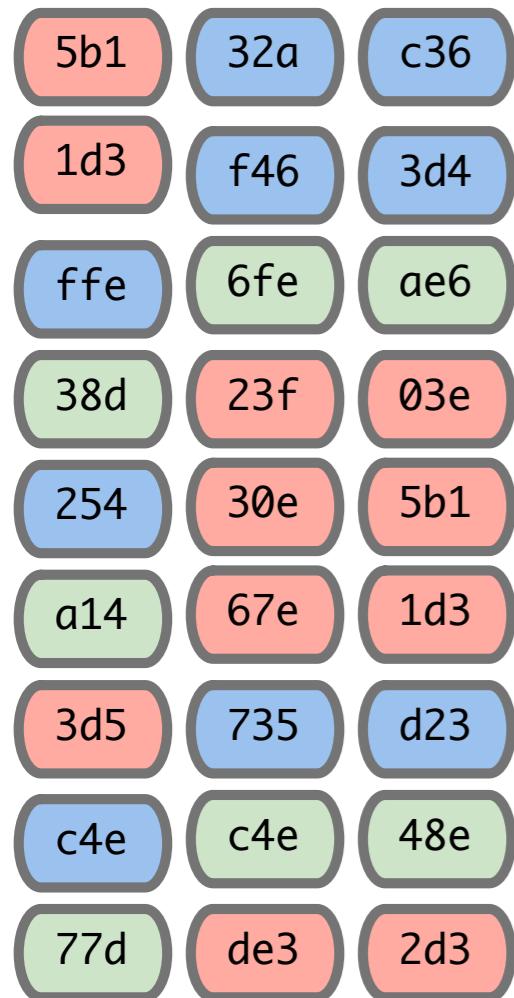
Index



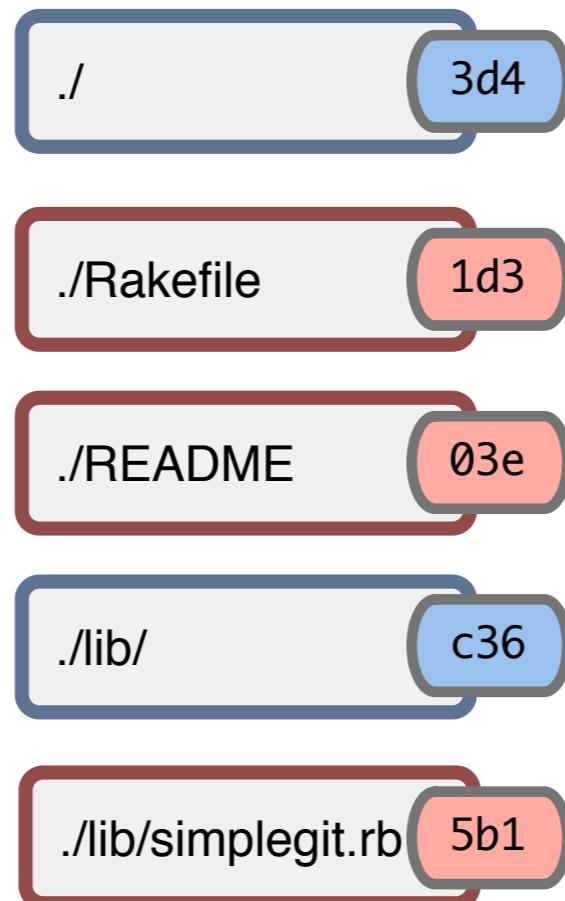
Working Directory



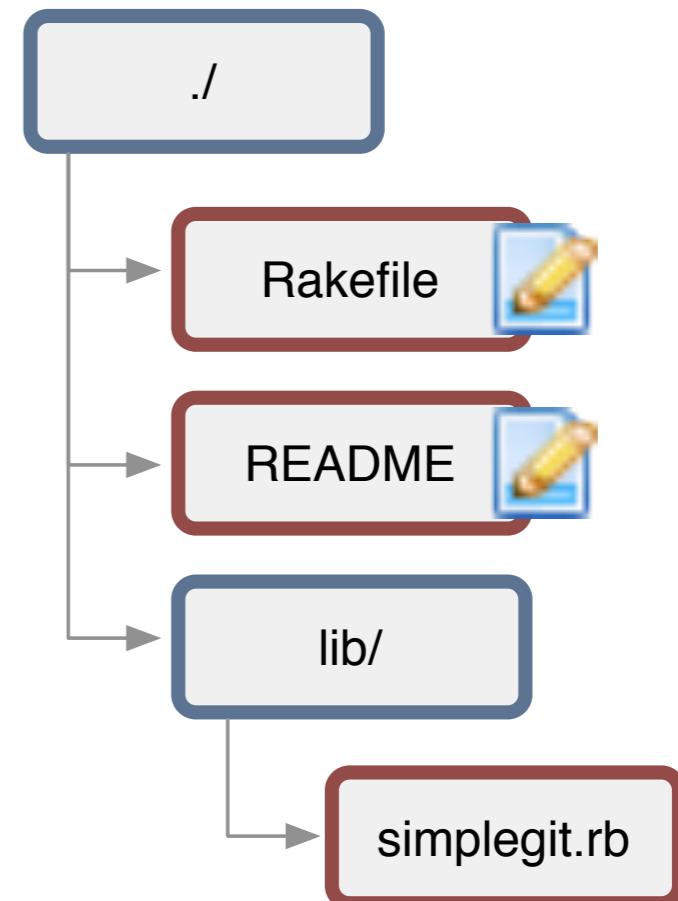
Repository



Index

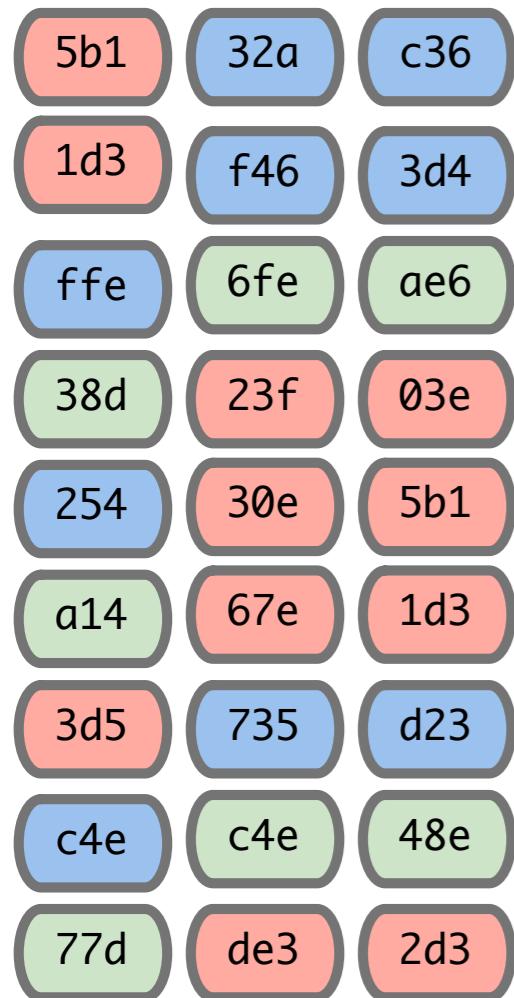


Working Directory

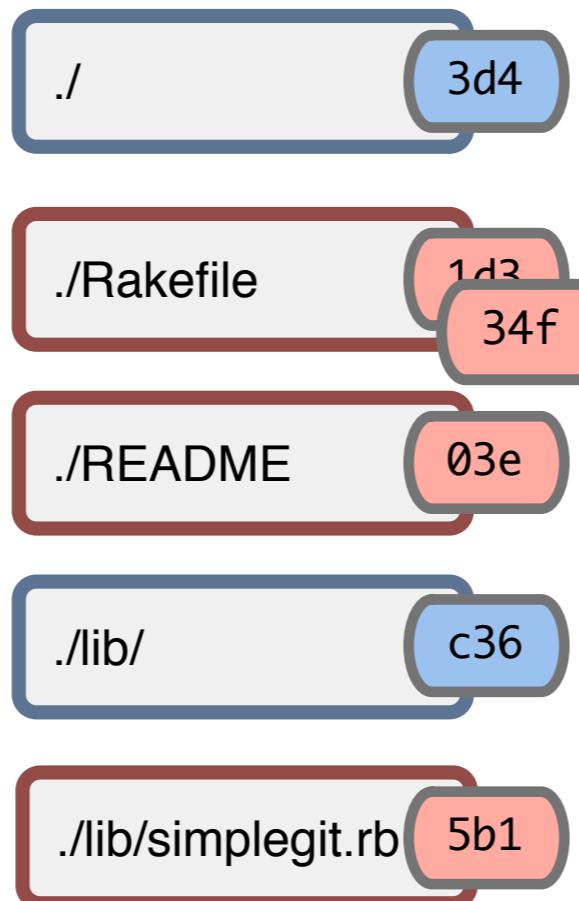


git add

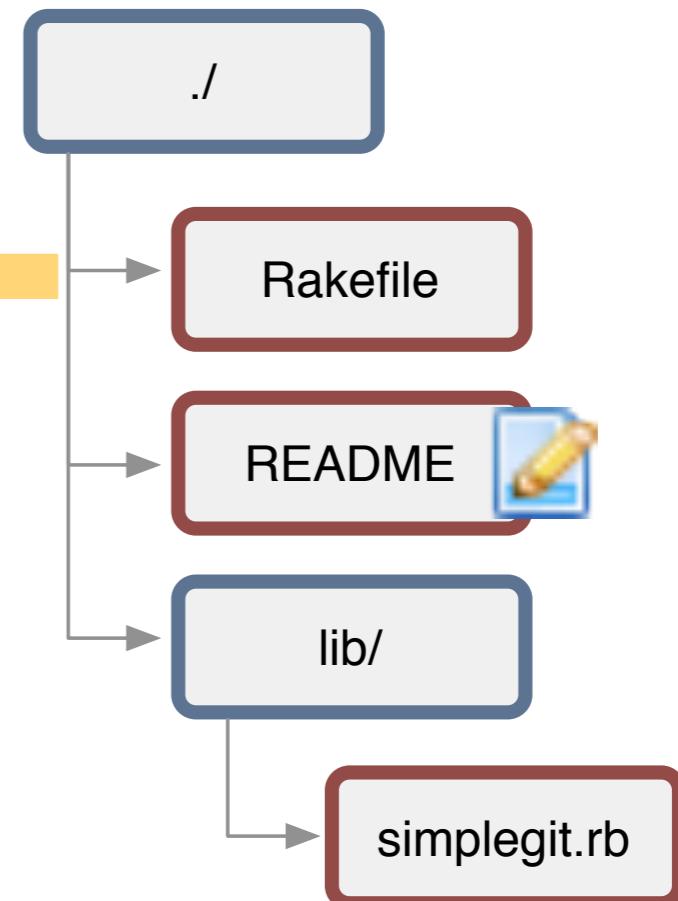
Repository



Index



Working Directory

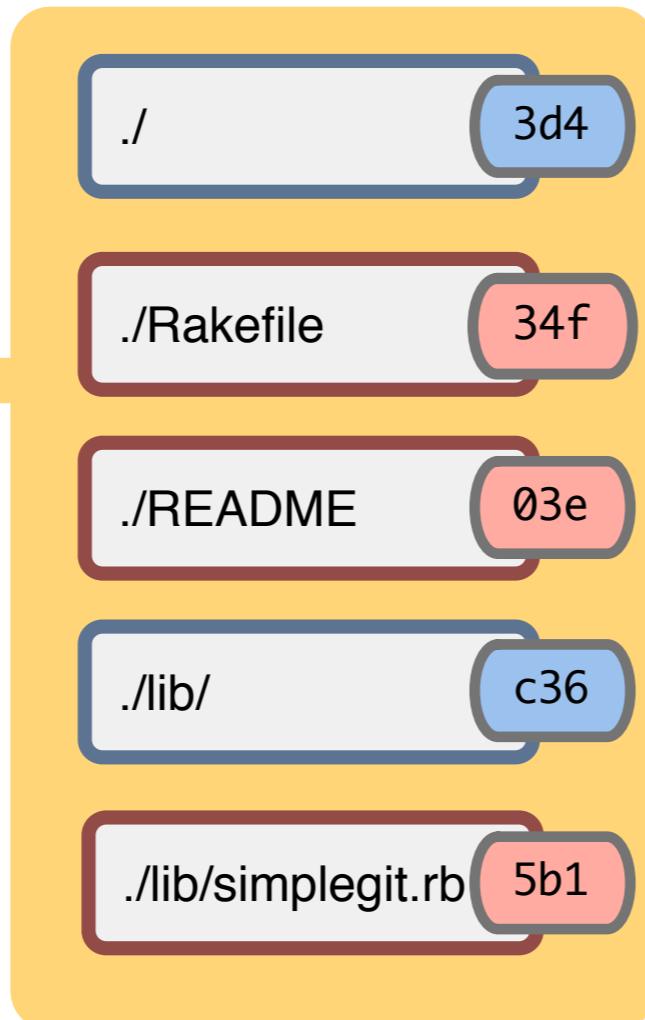


git add

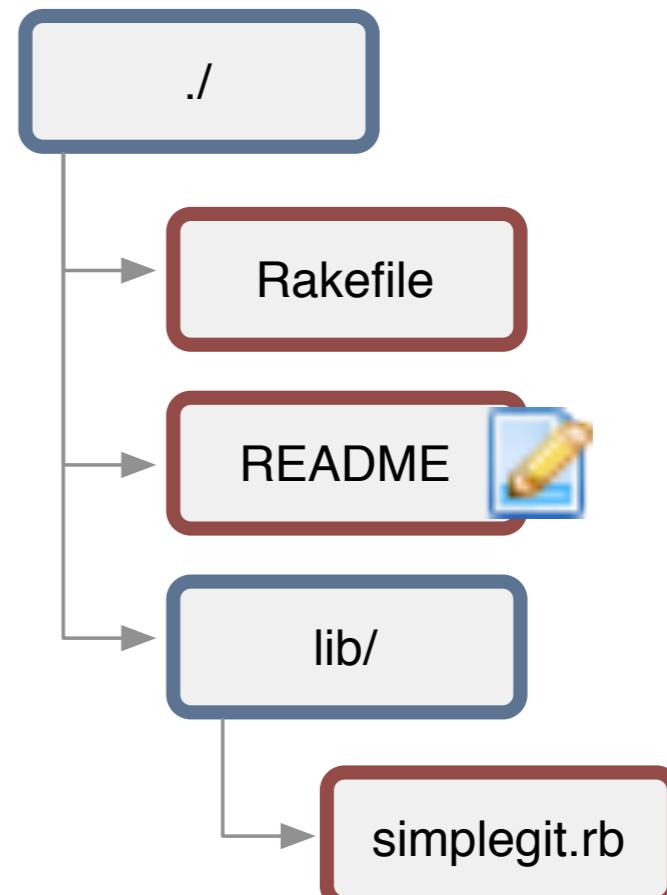
Repository



Index



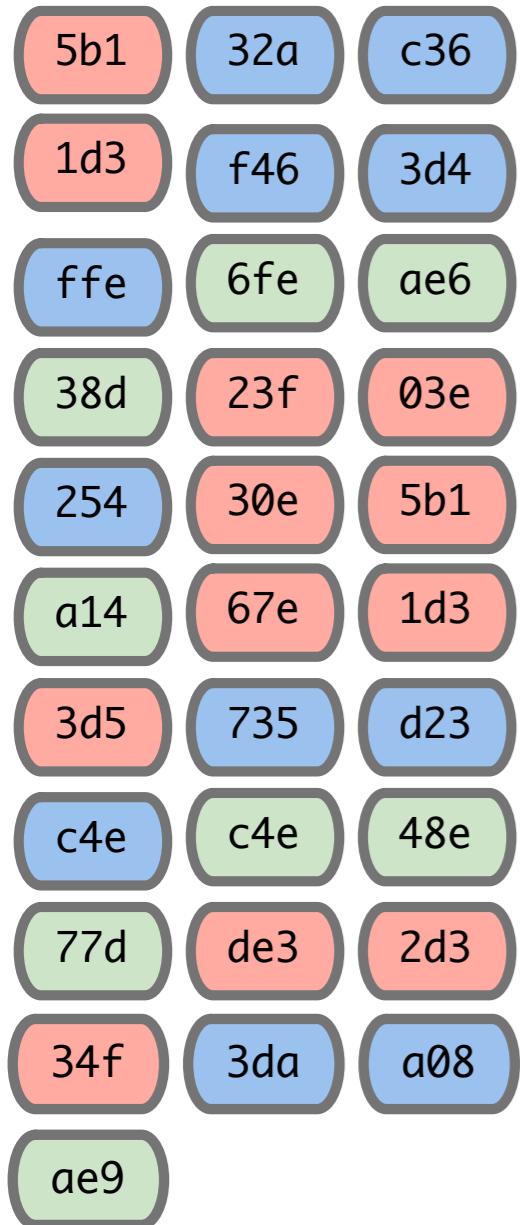
Working Directory



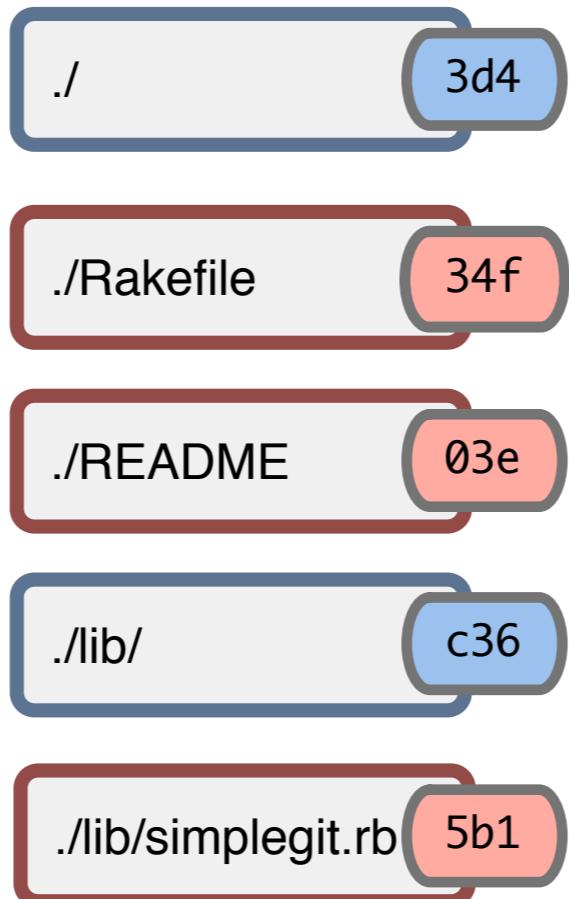
git commit



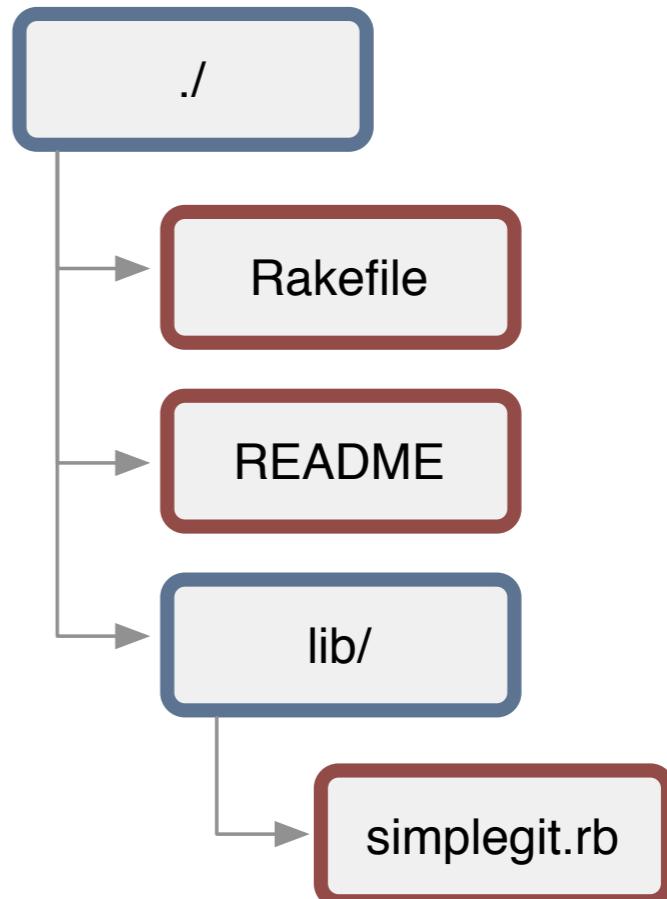
Repository



Index



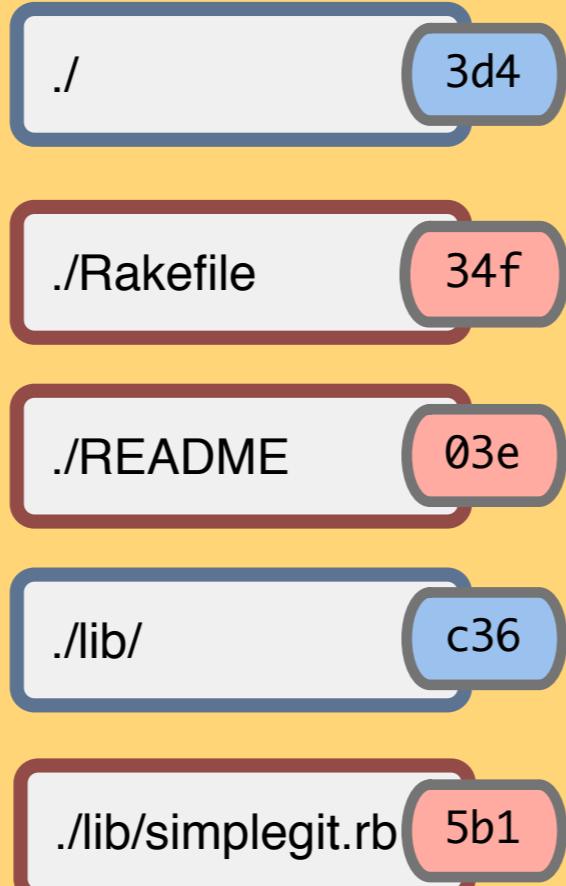
Working Directory



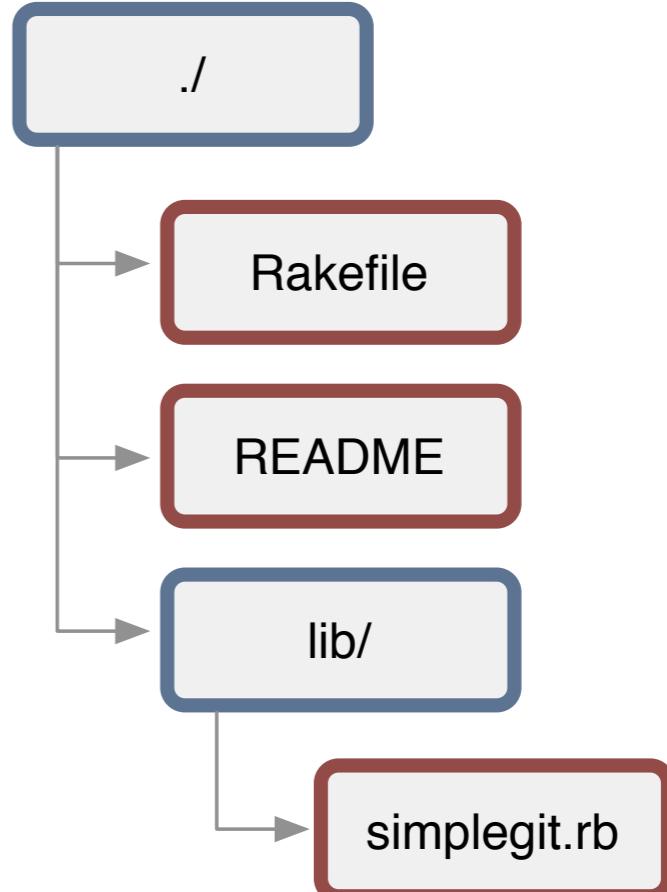
Repository



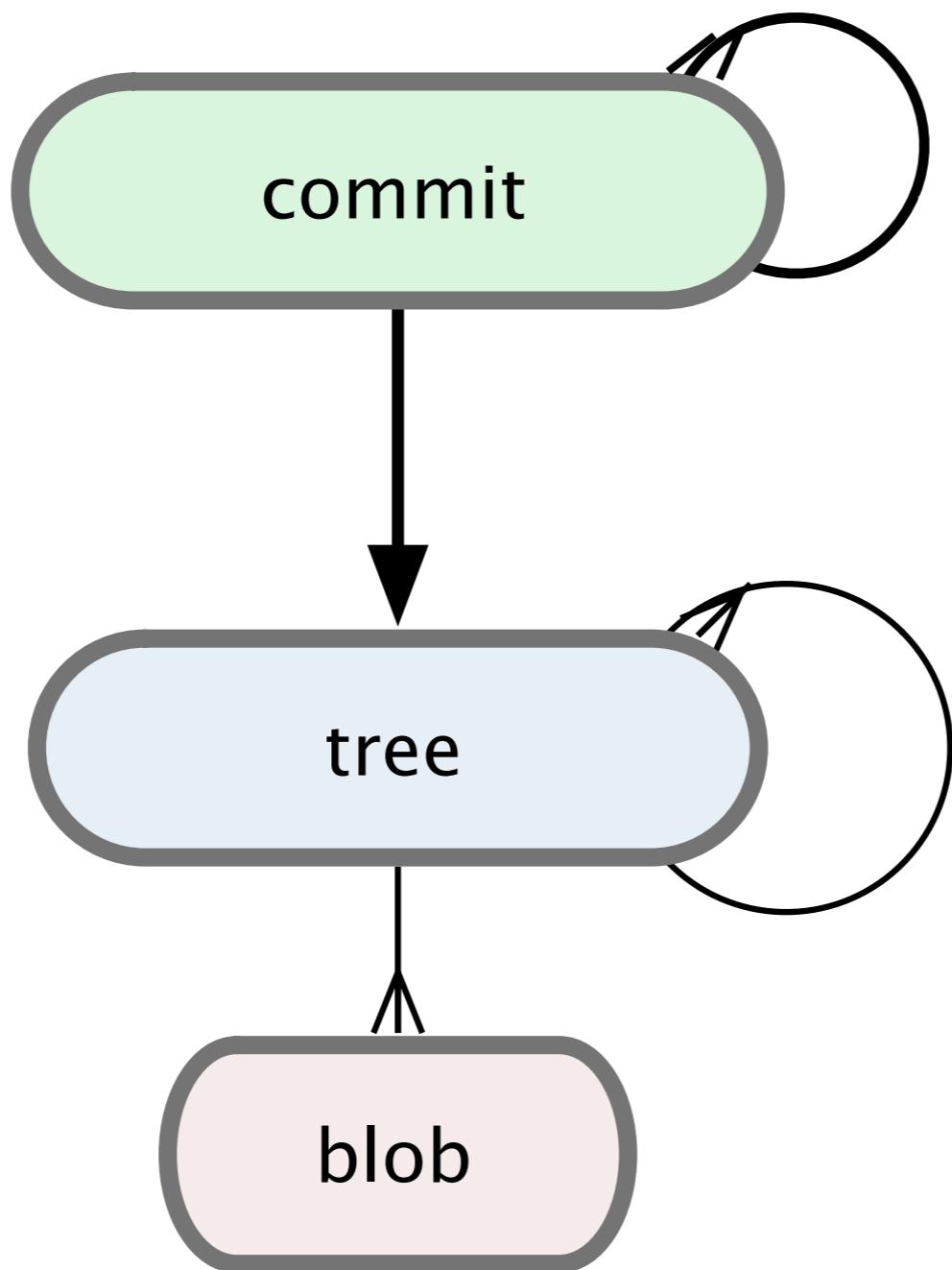
Index



Working Directory



object model



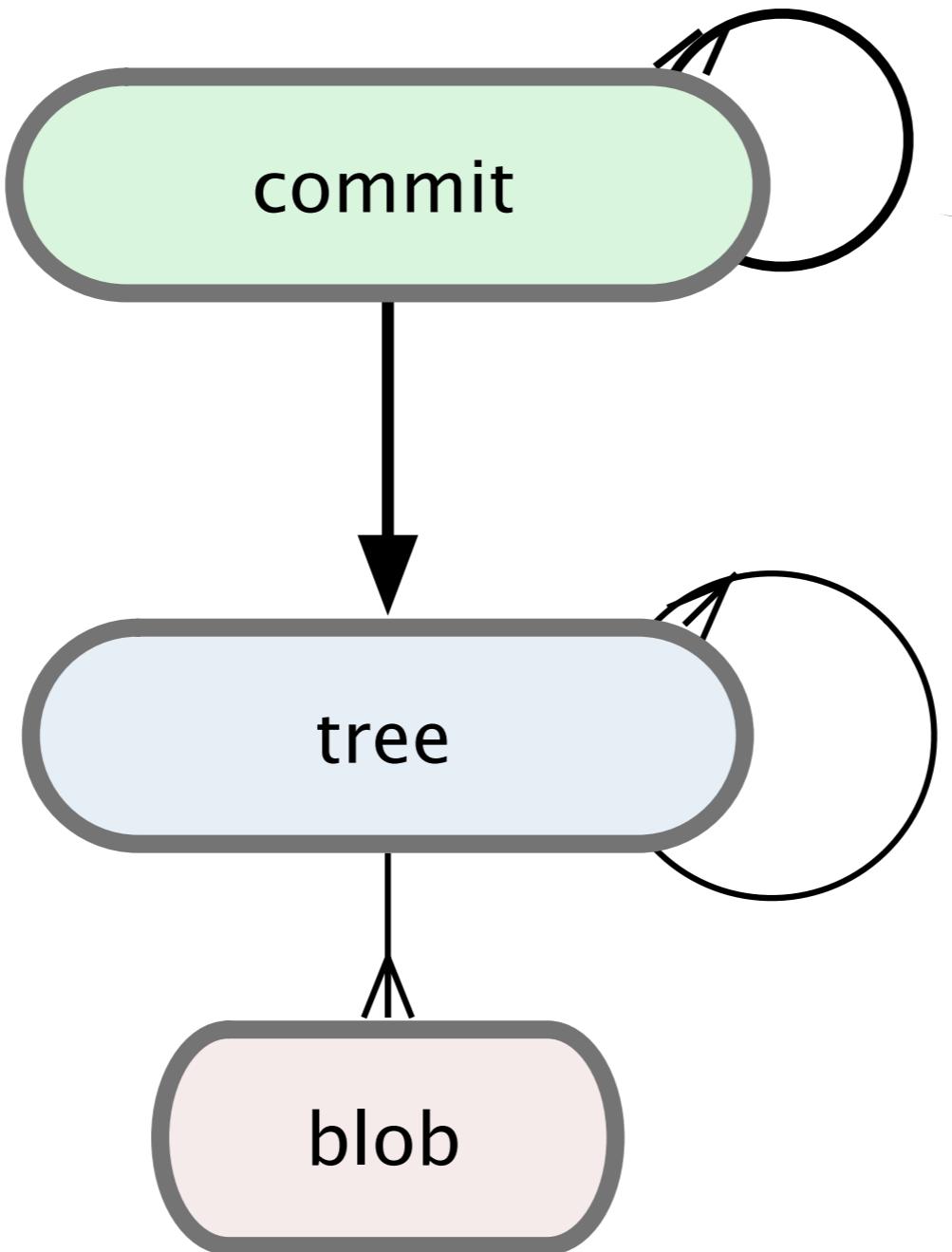
Branching and Merging

object model

pointer to a
snapshot

directory list

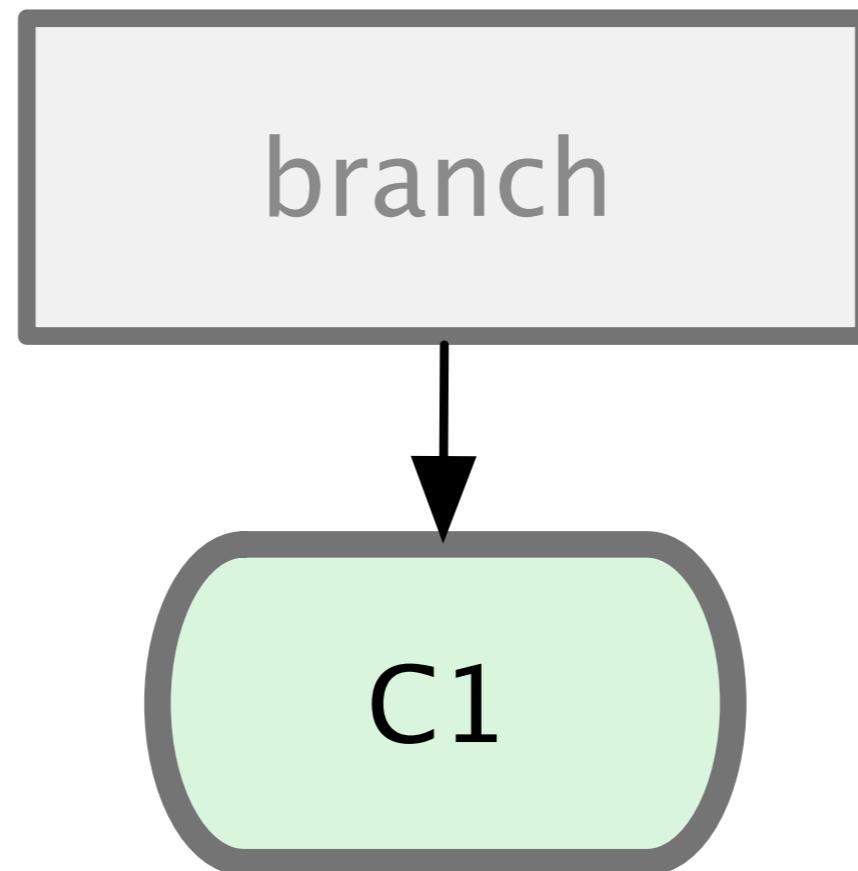
file contents



branches

branches

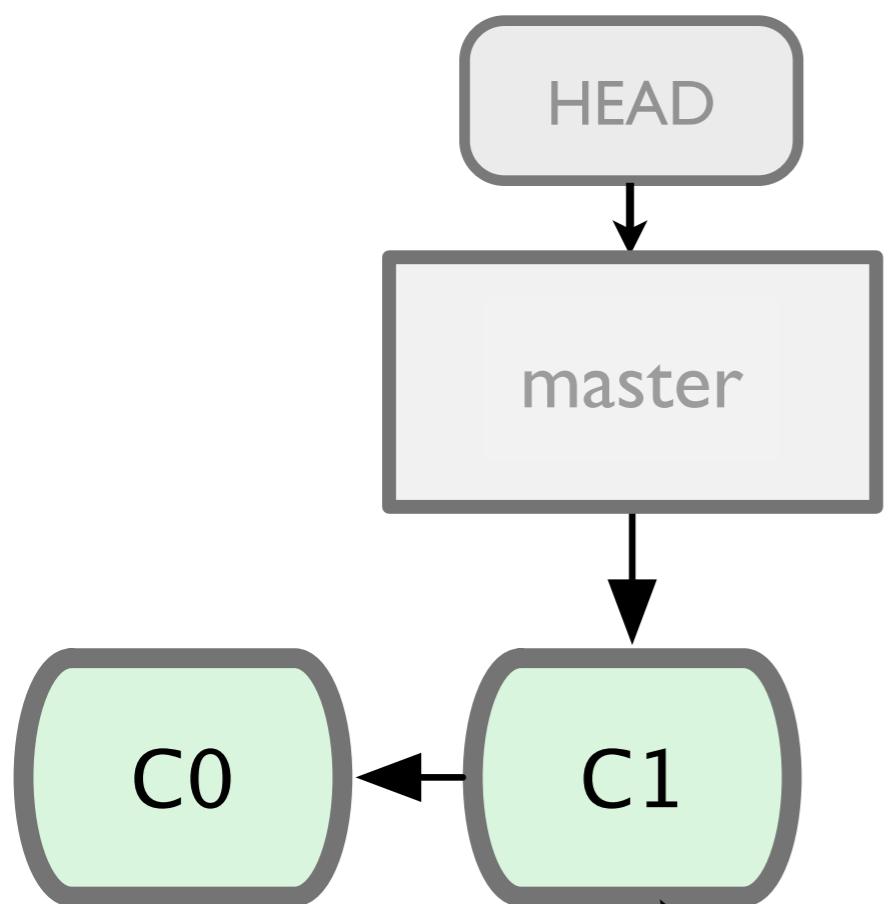
lightweight, movable
pointers to a commit

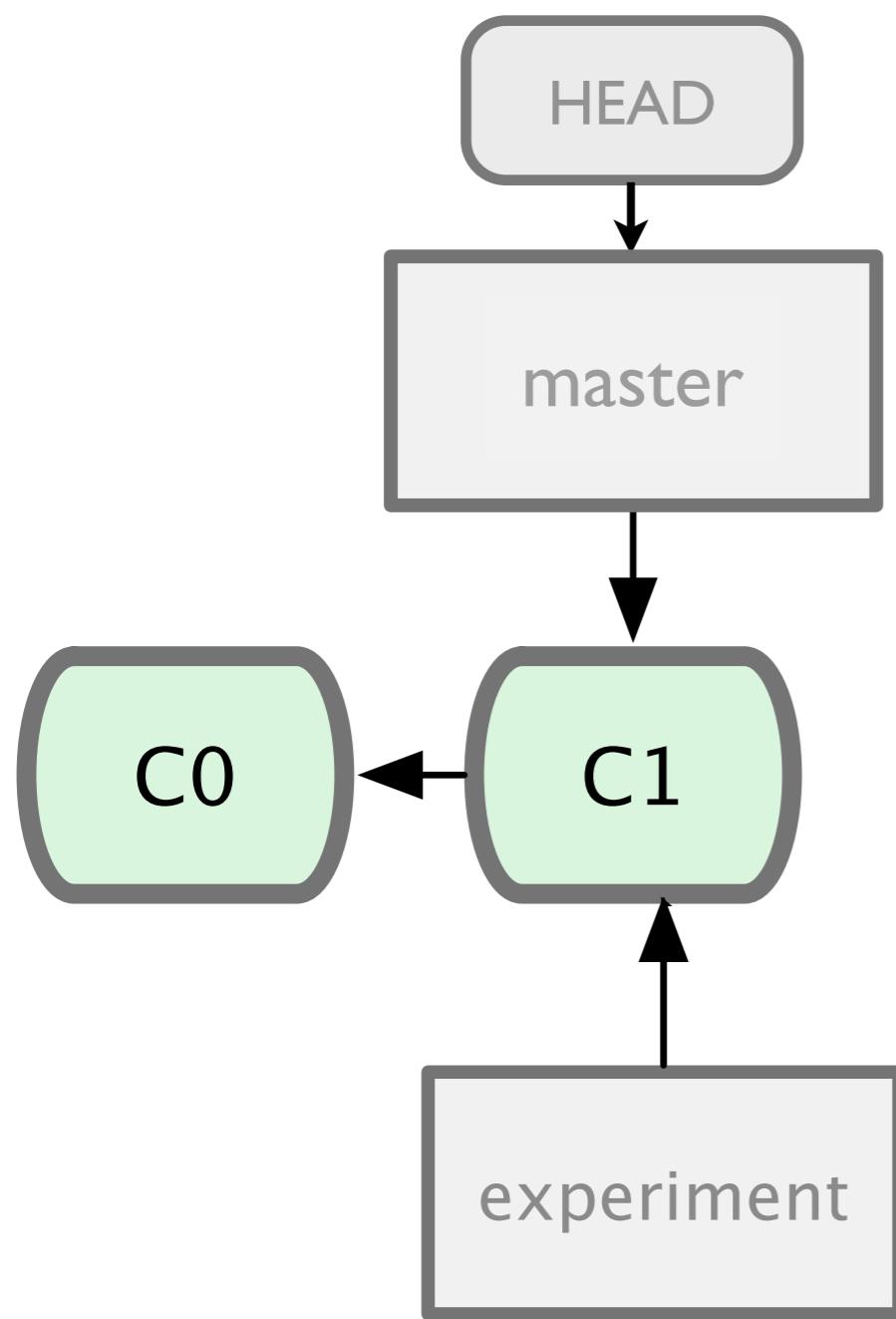


branching

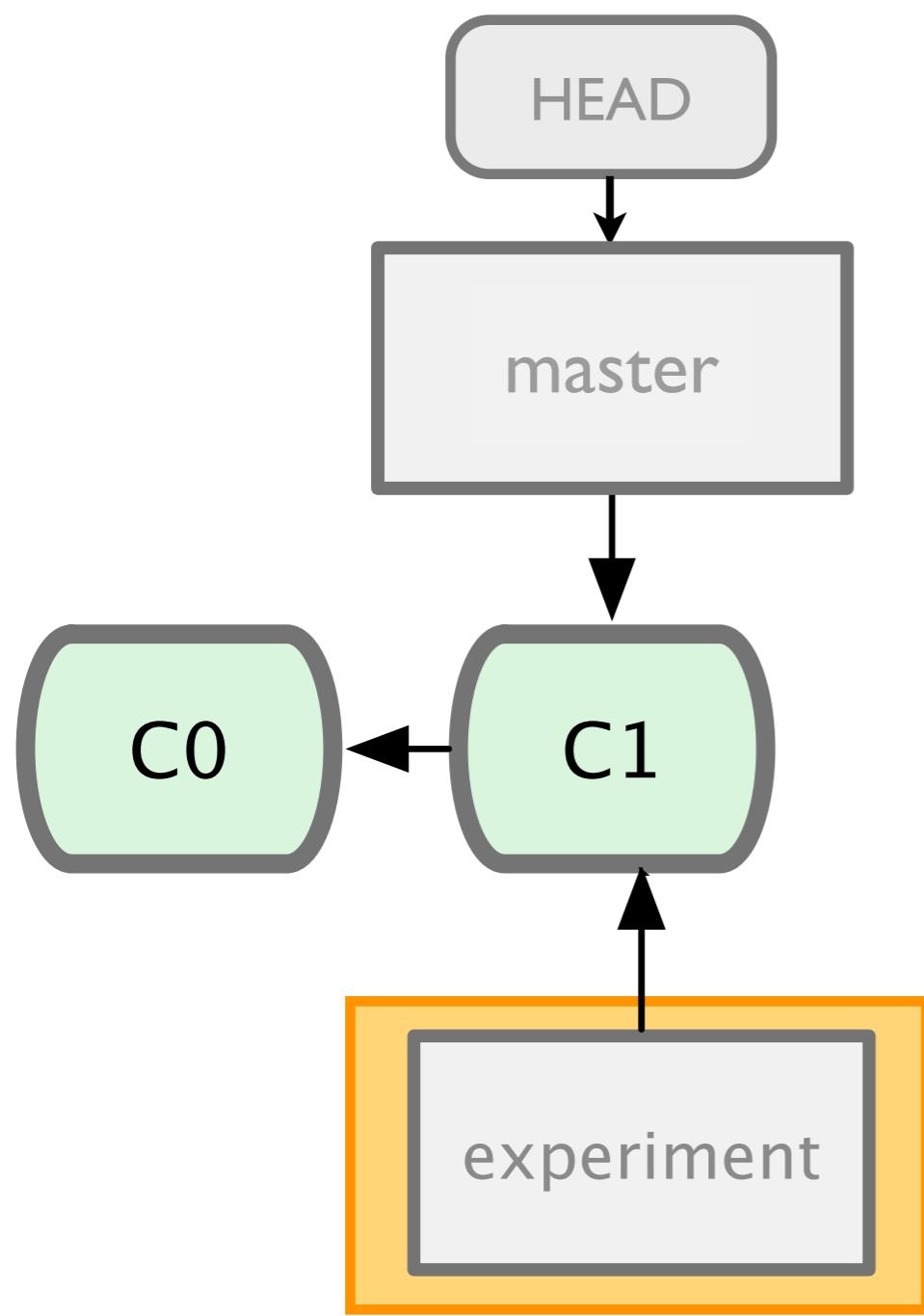
git branch

git checkout

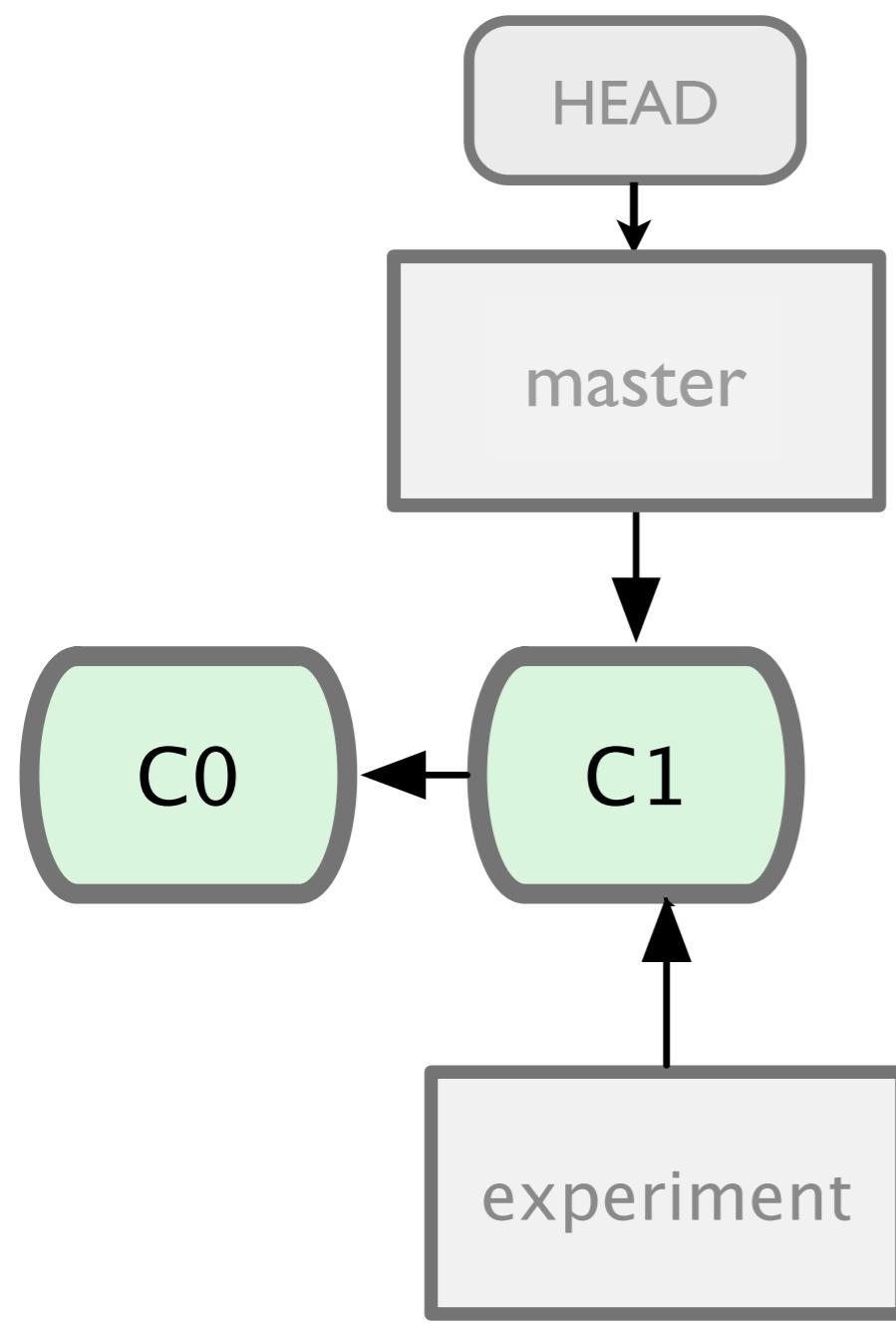




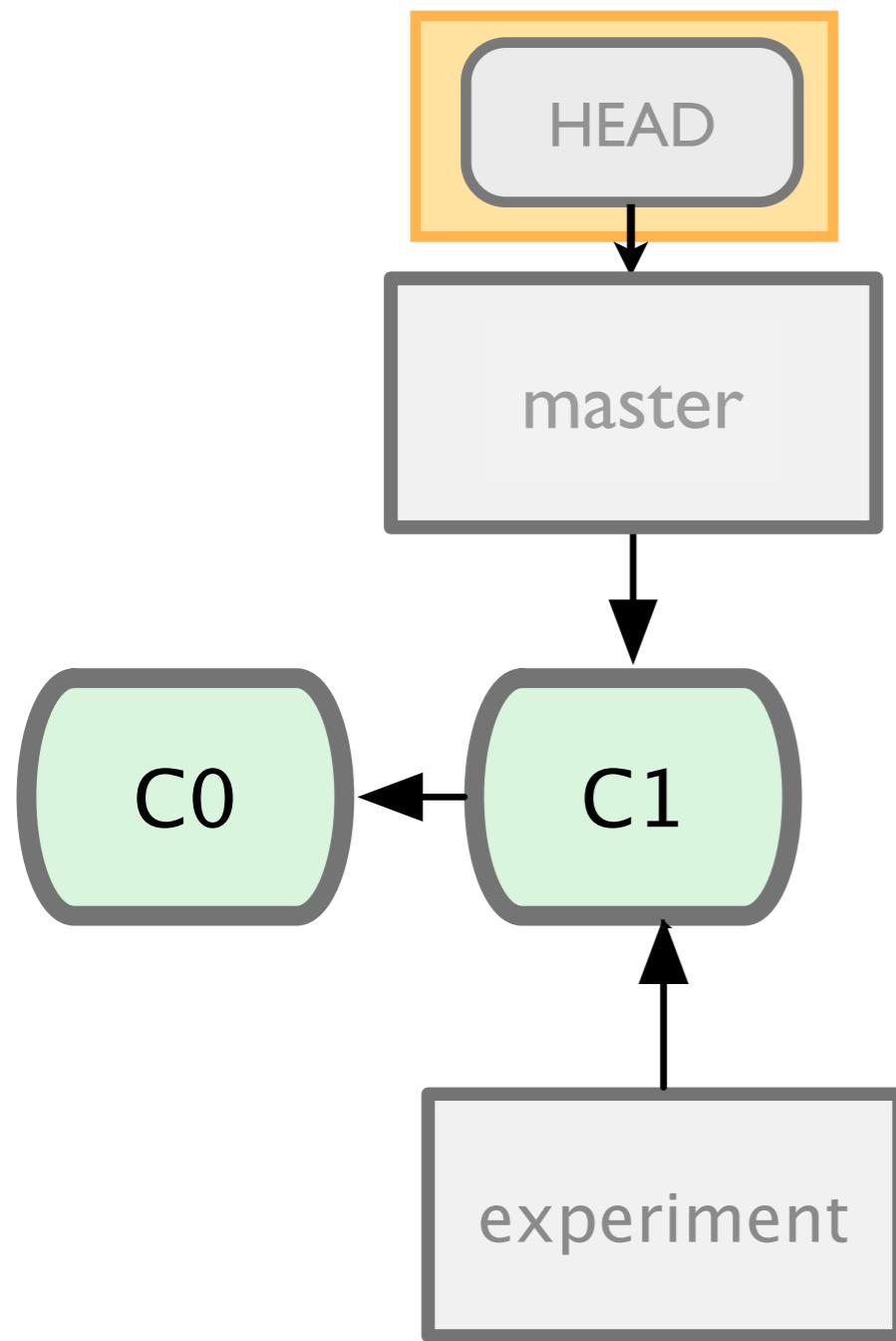
git branch experiment



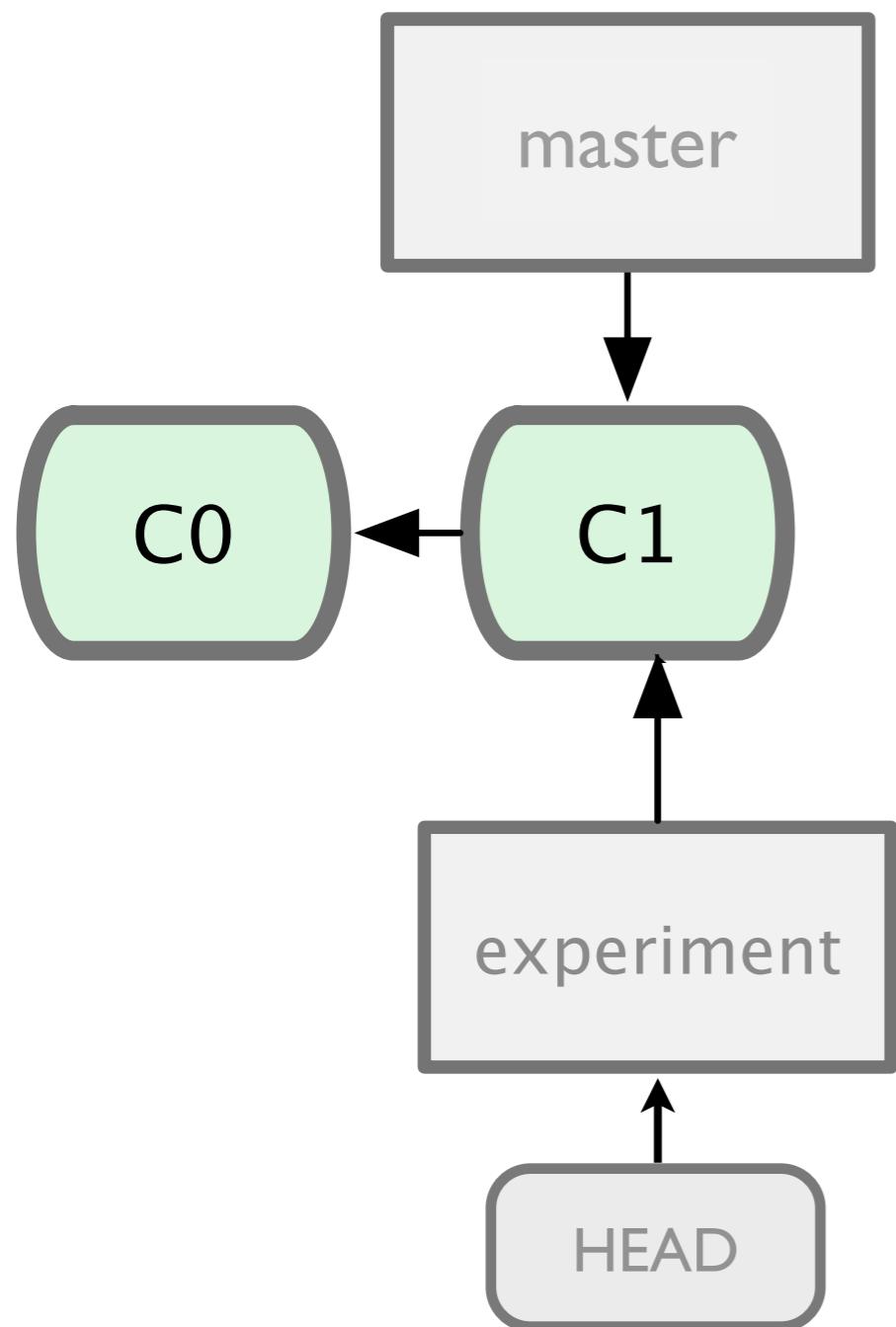
git branch experiment



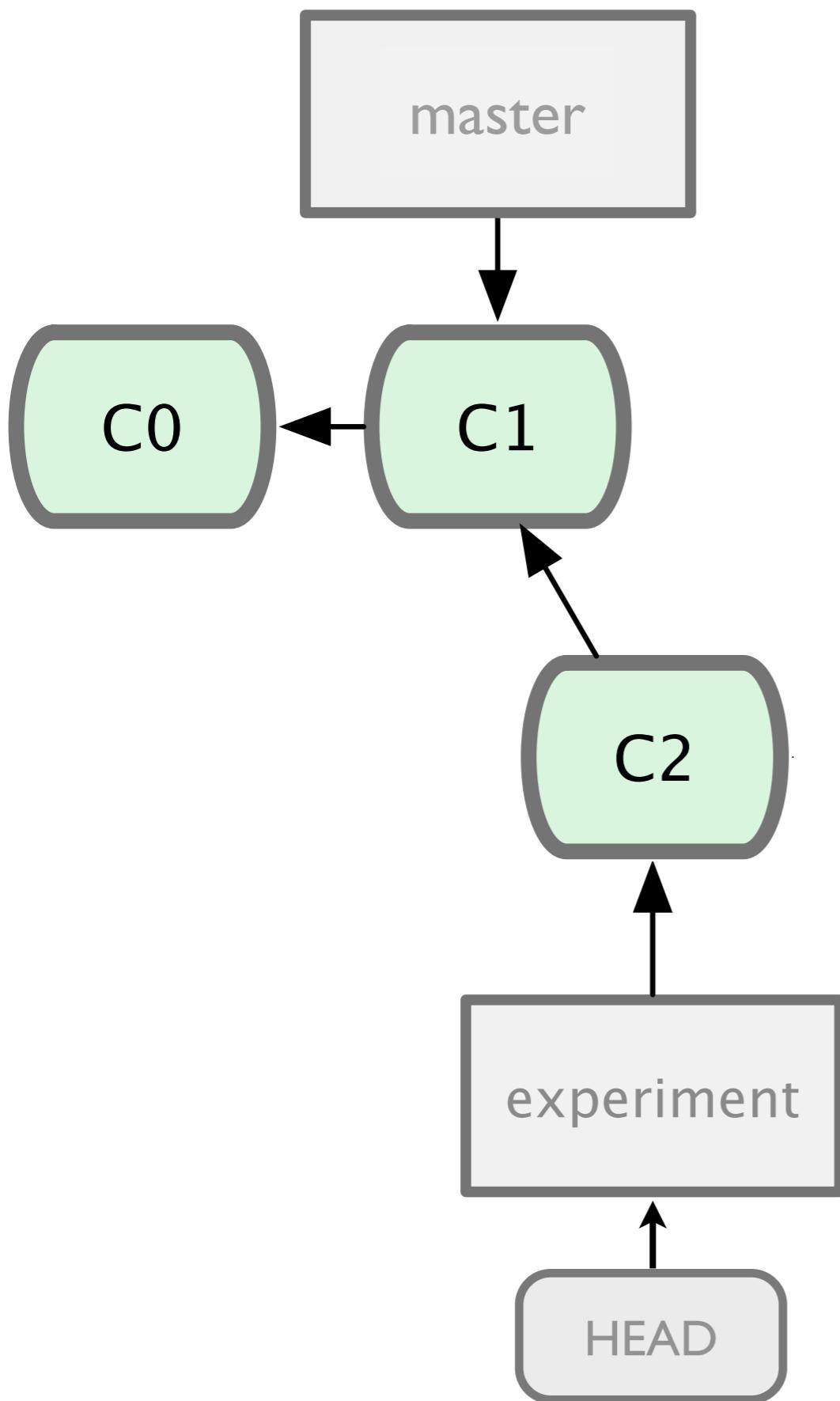
```
$ git branch  
* master  
  experiment
```



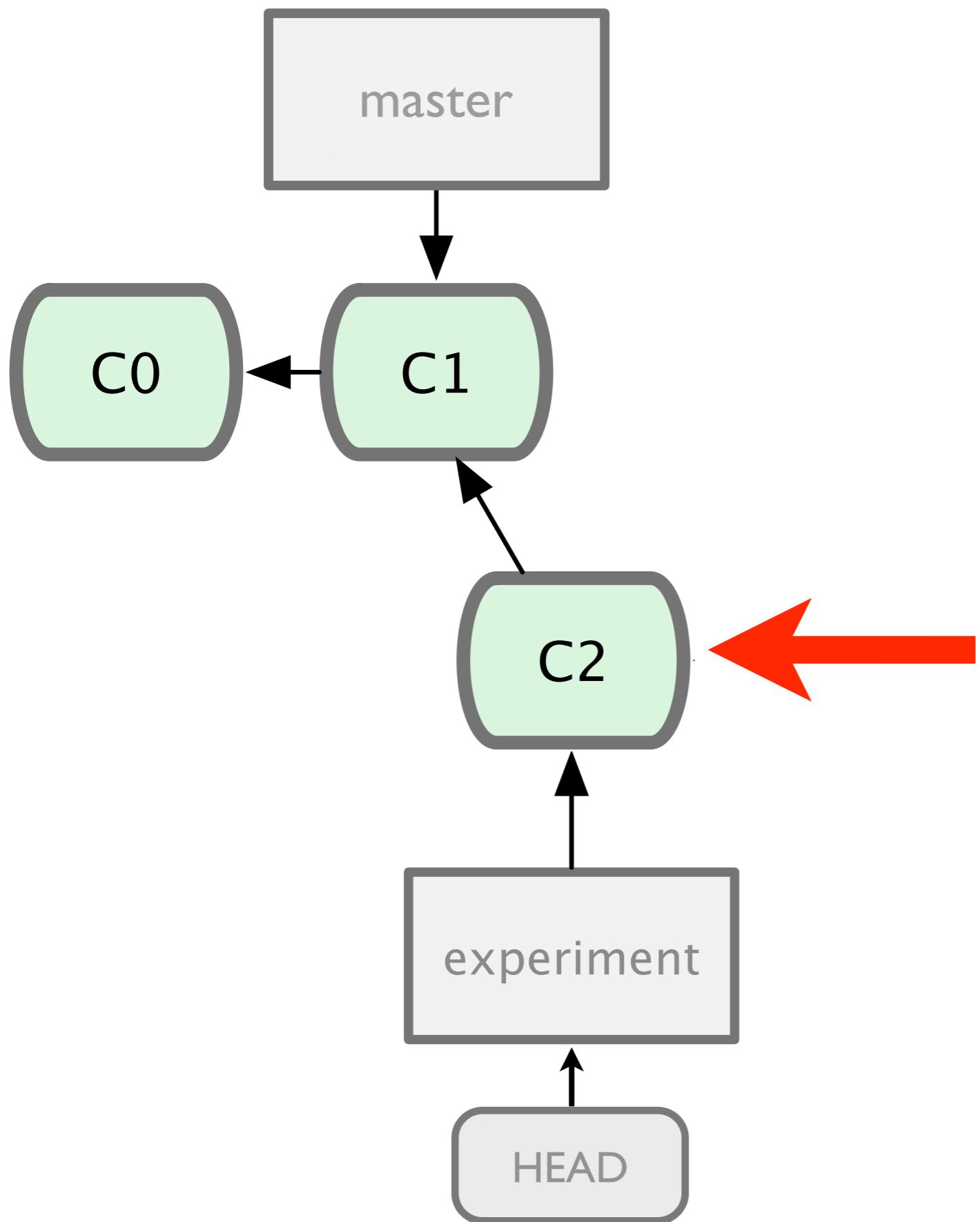
```
$ git branch  
* master  
  experiment
```



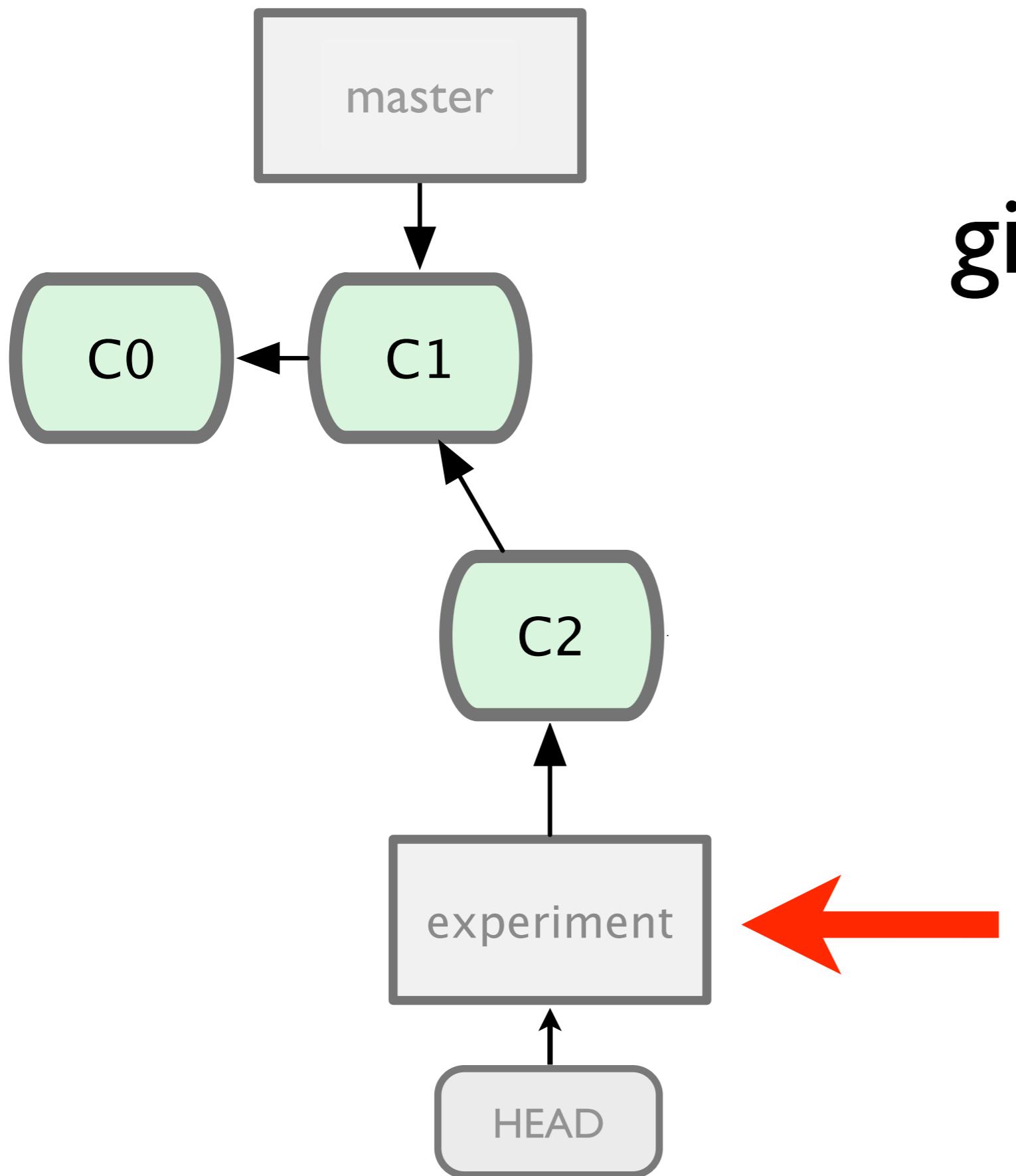
git checkout experiment



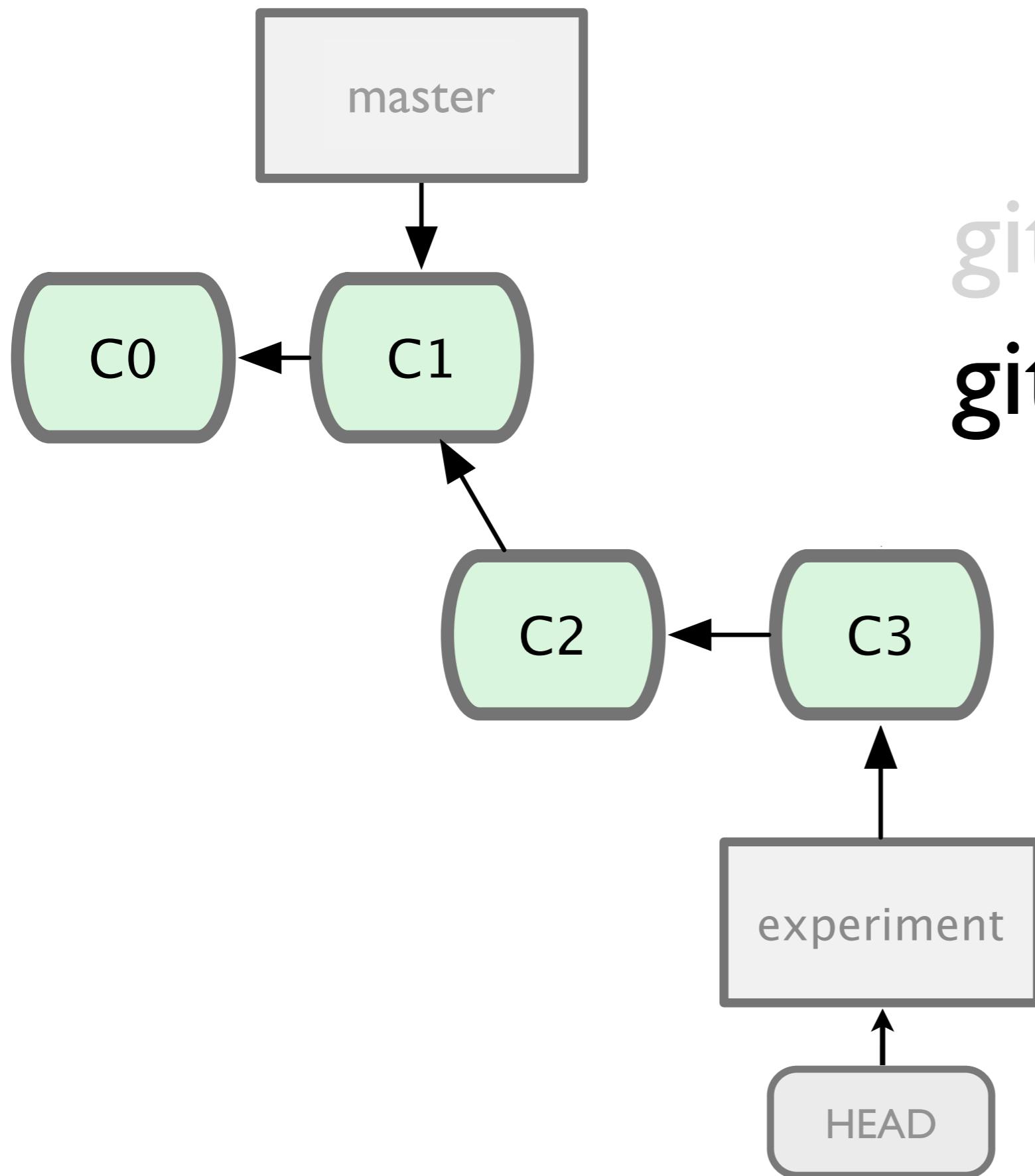
git commit



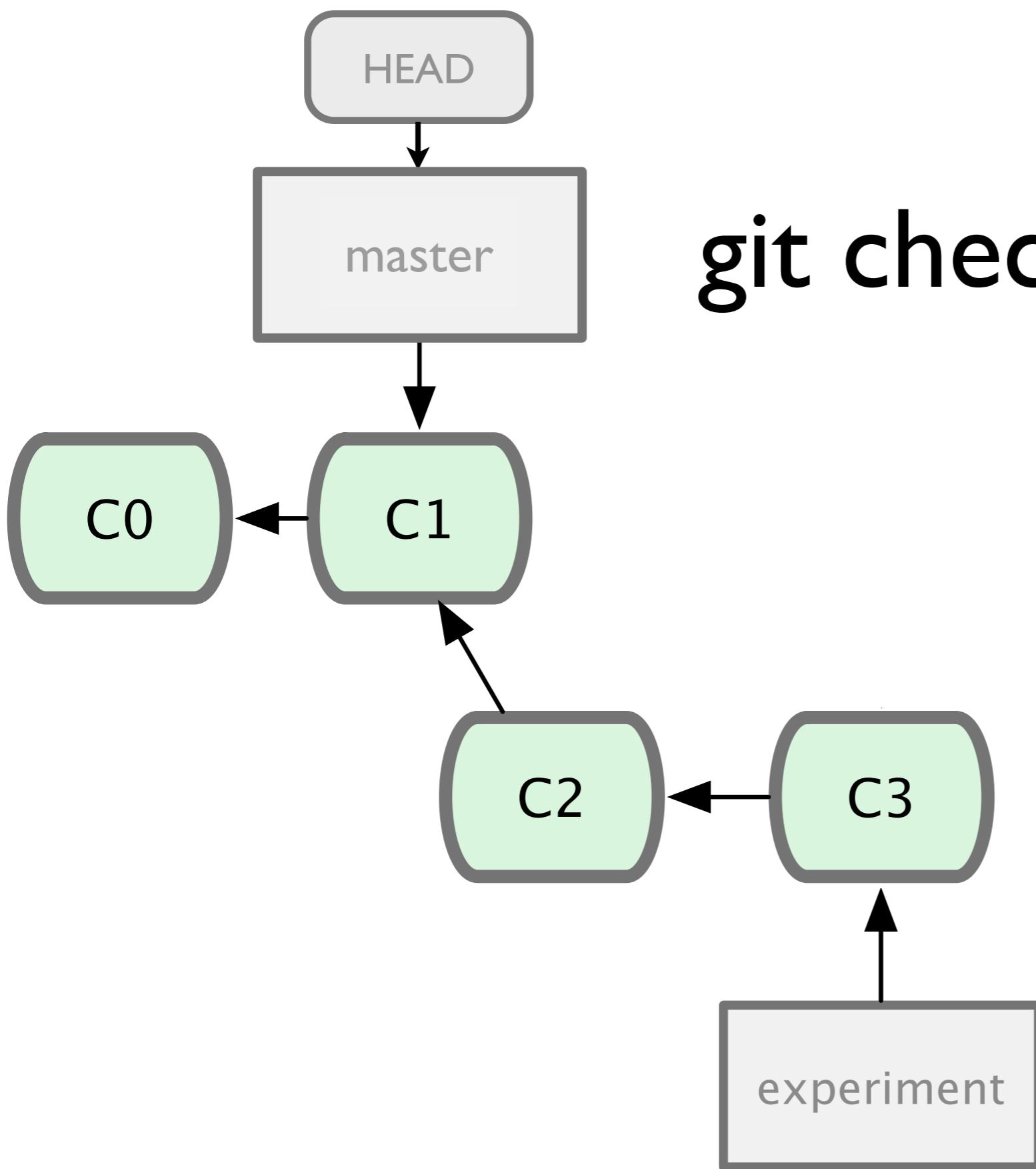
git commit



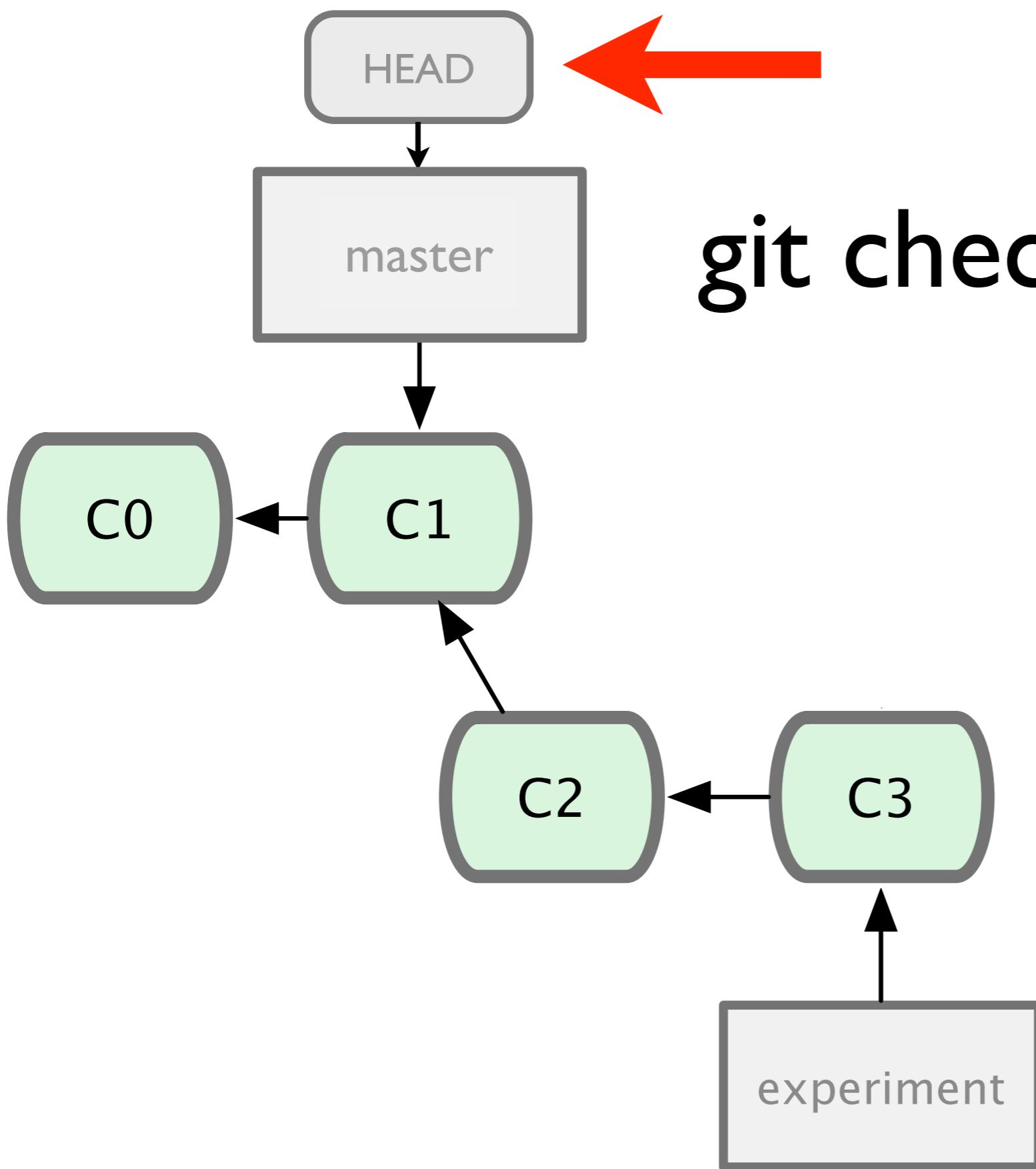
git commit



git commit
git commit

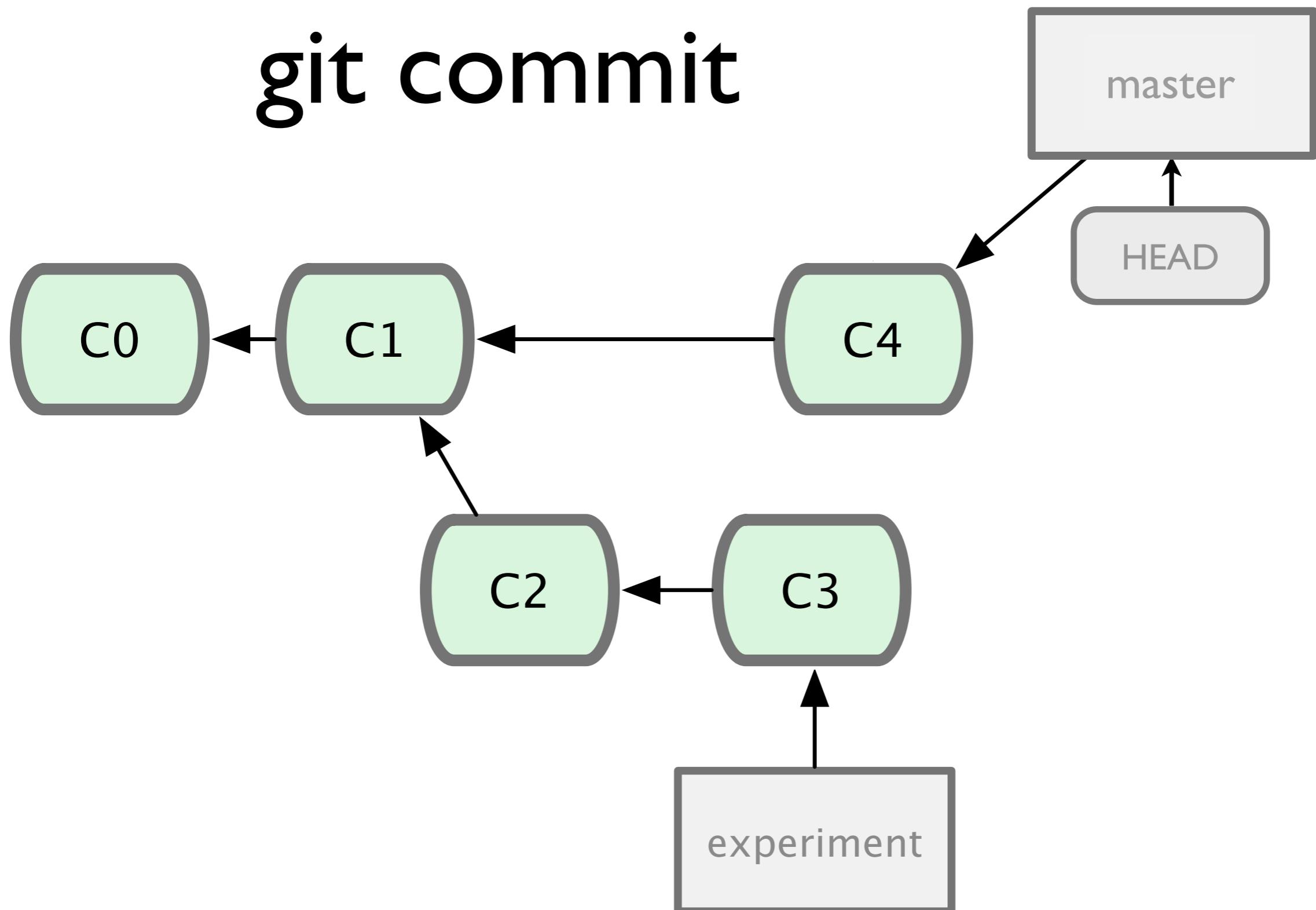


git checkout master

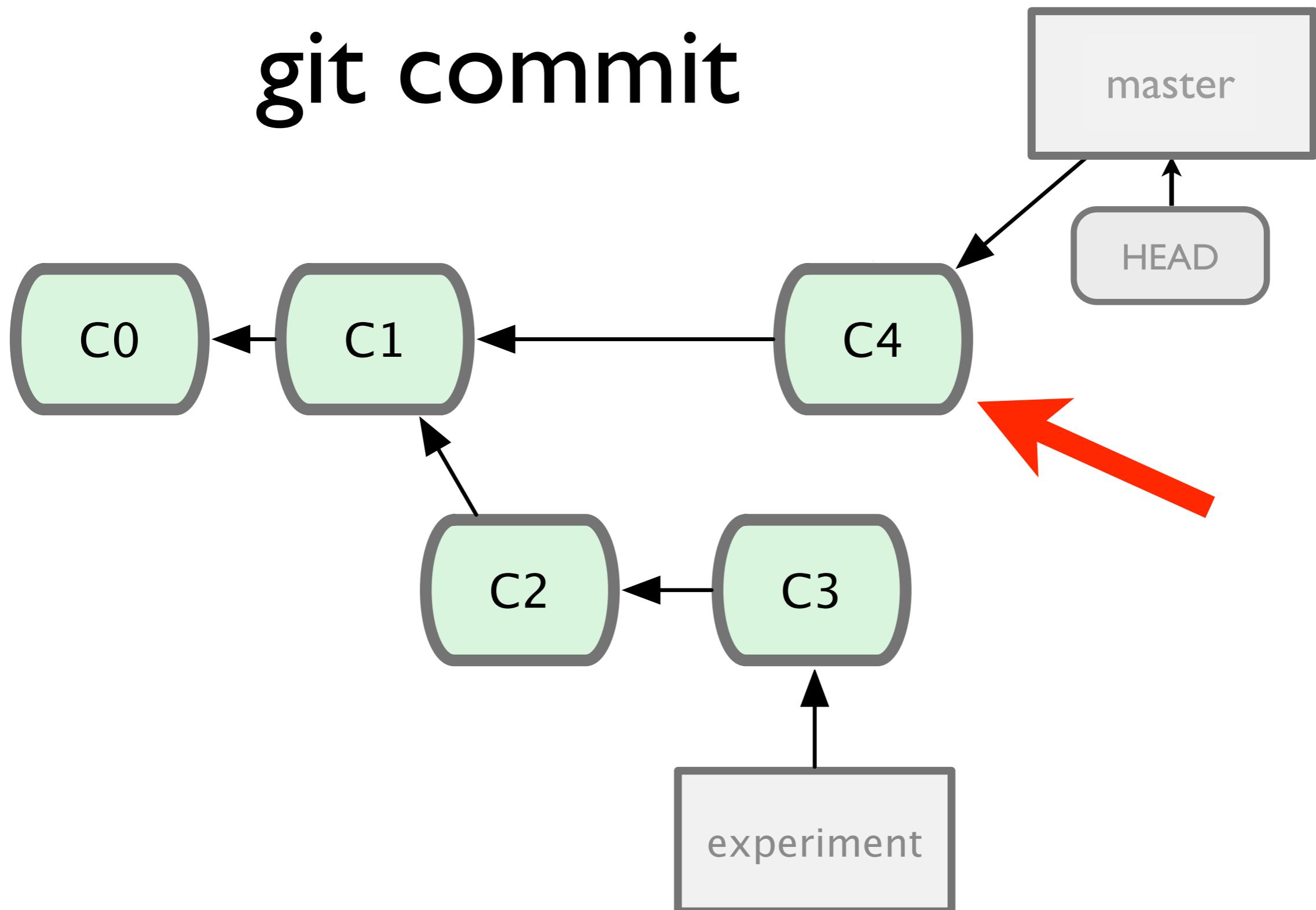


git checkout master

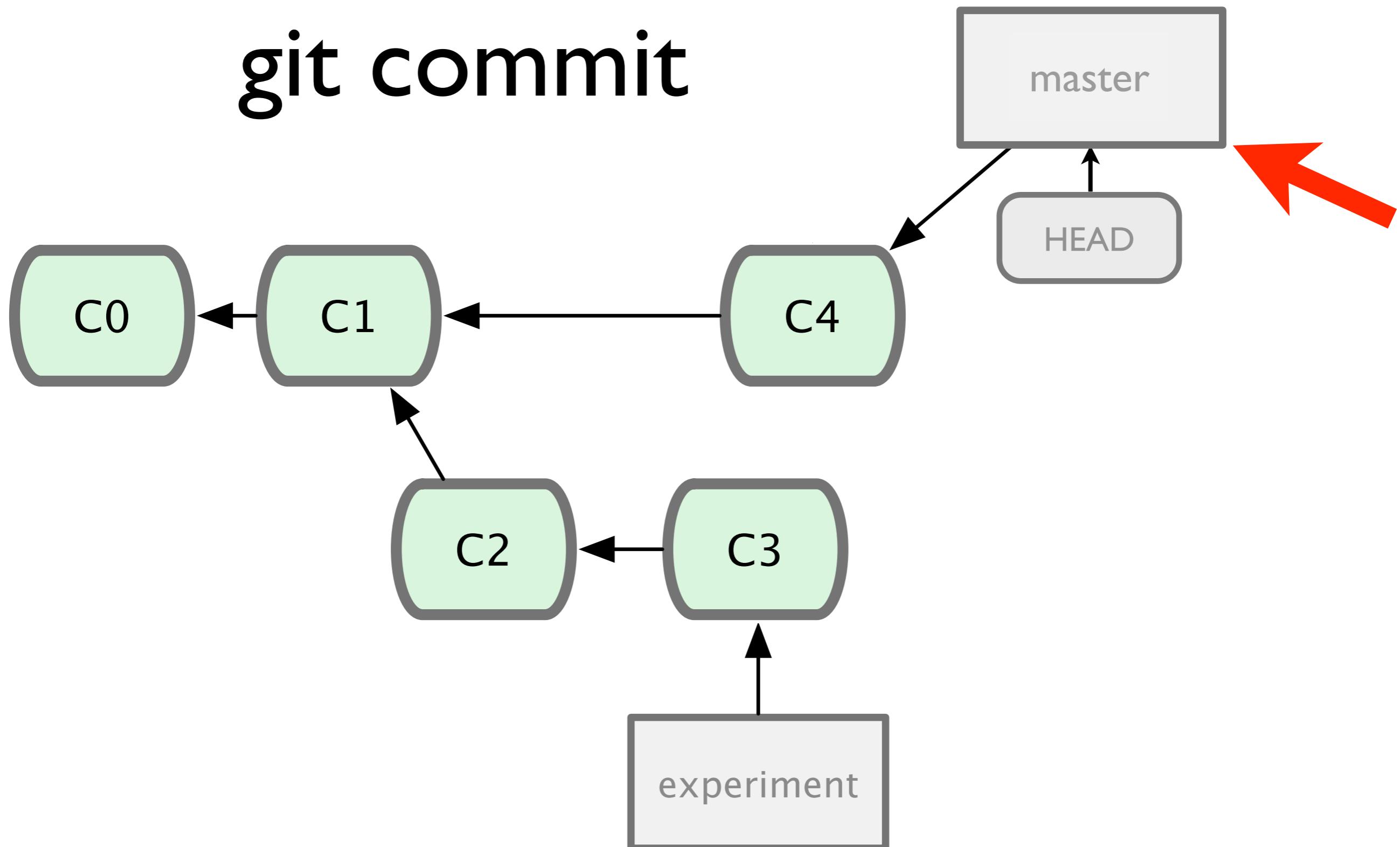
git commit

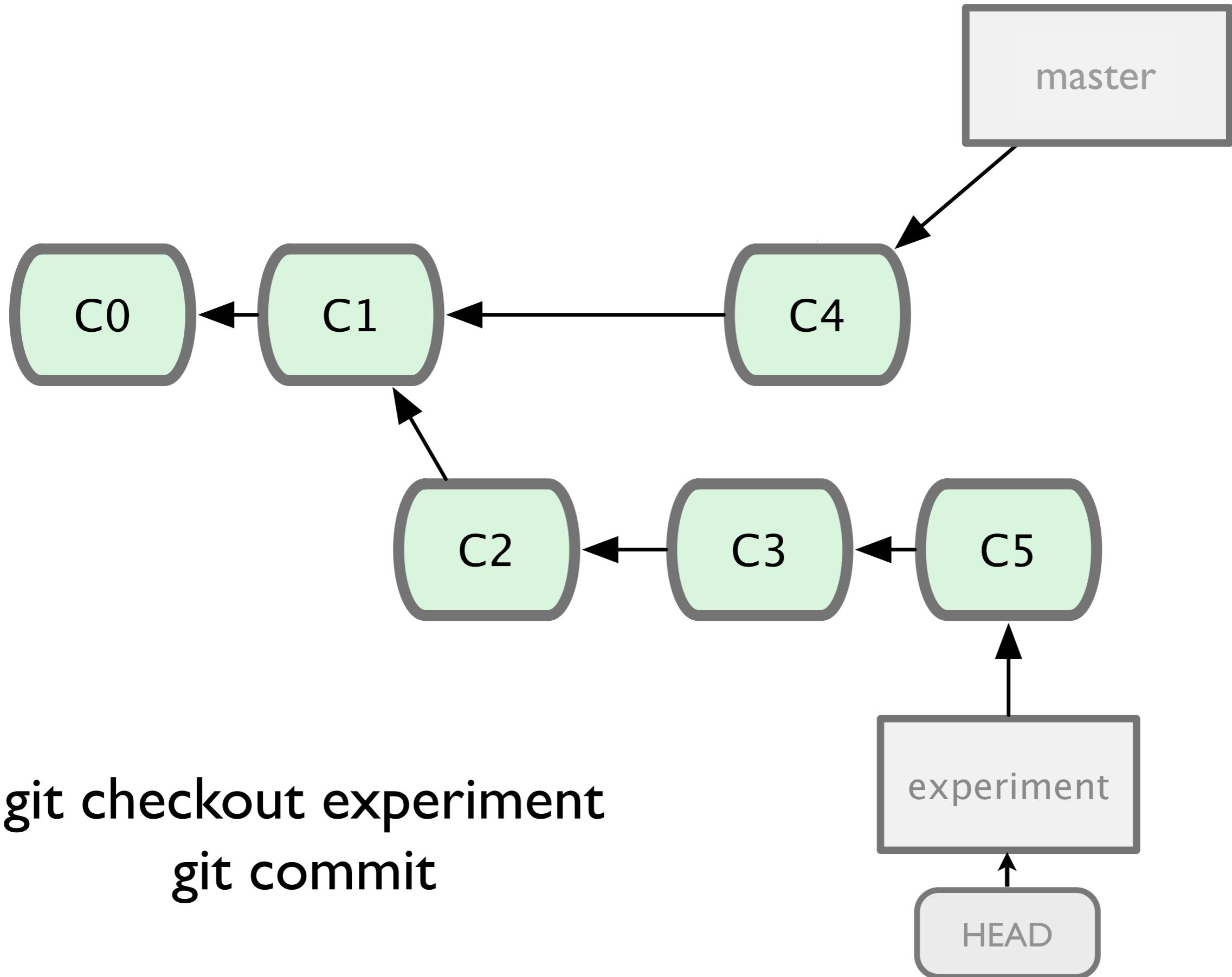


git commit

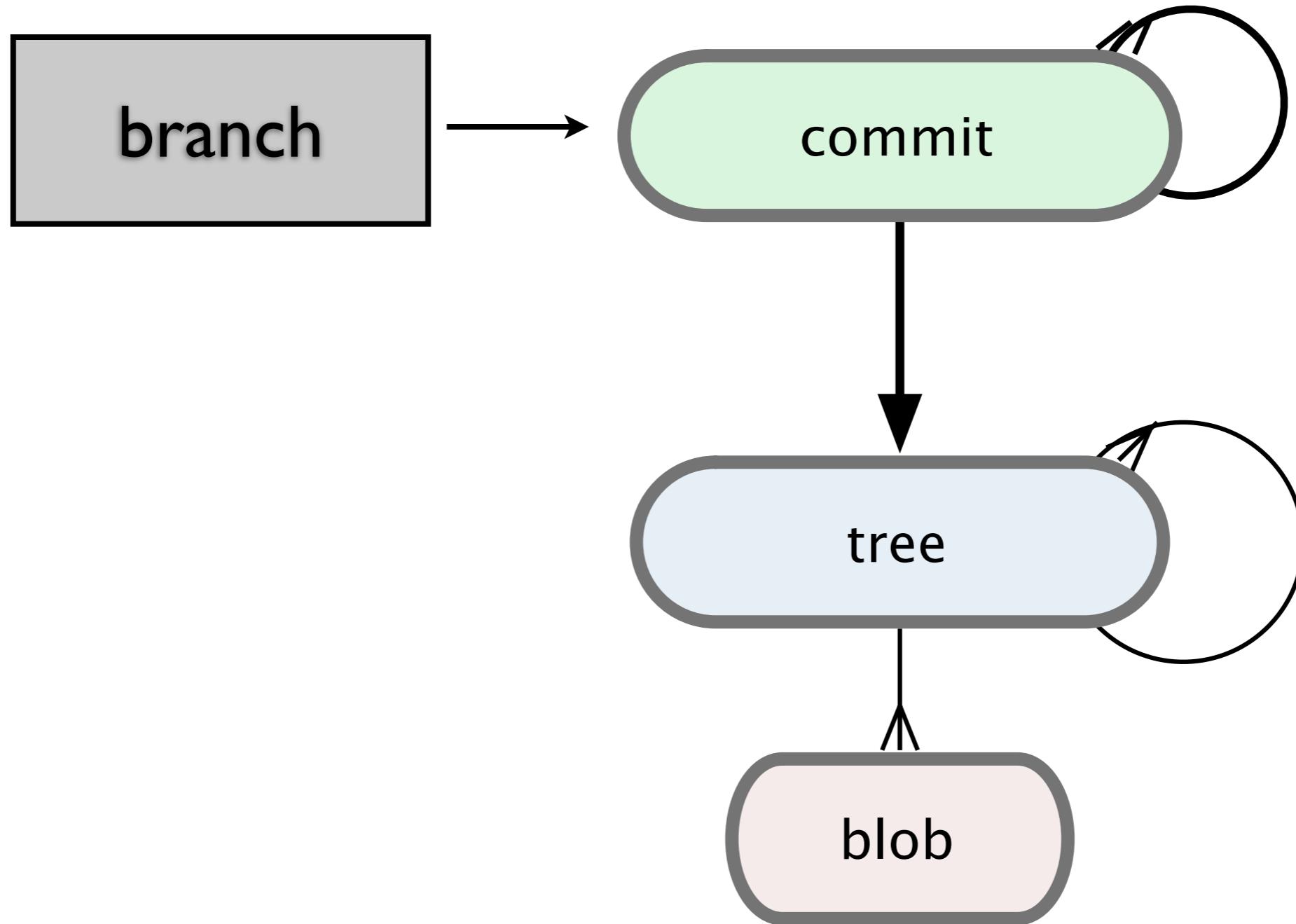


git commit



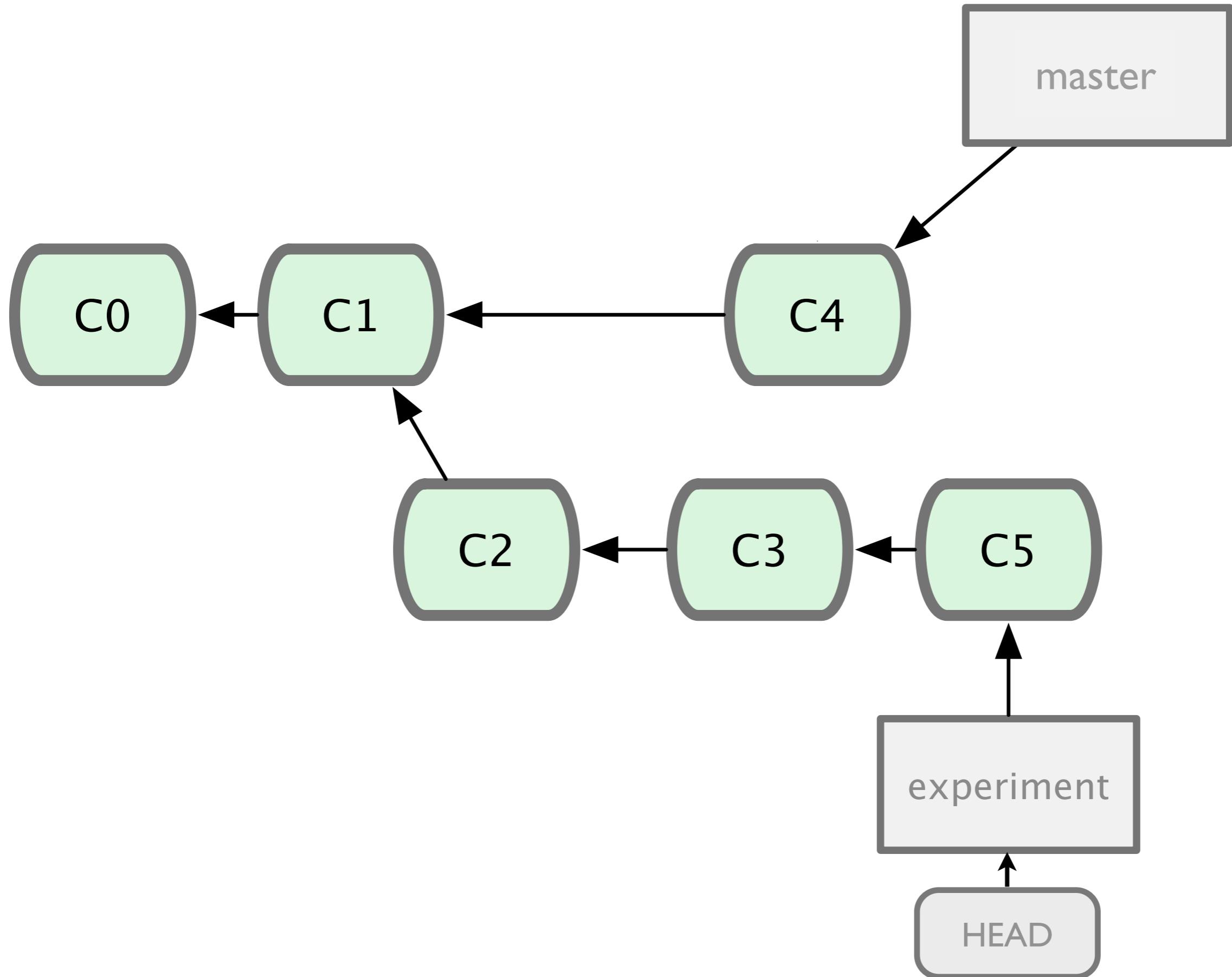


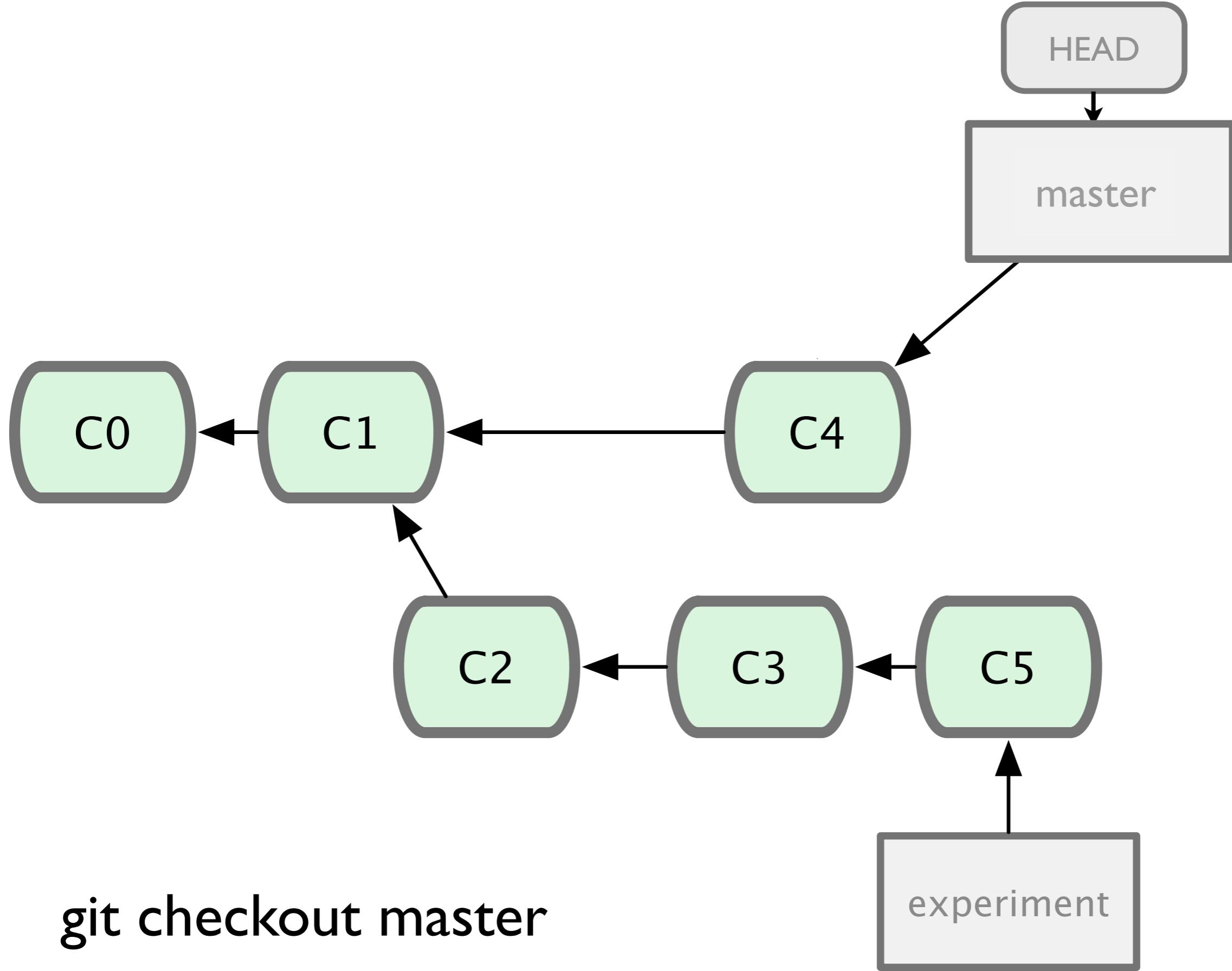
object model

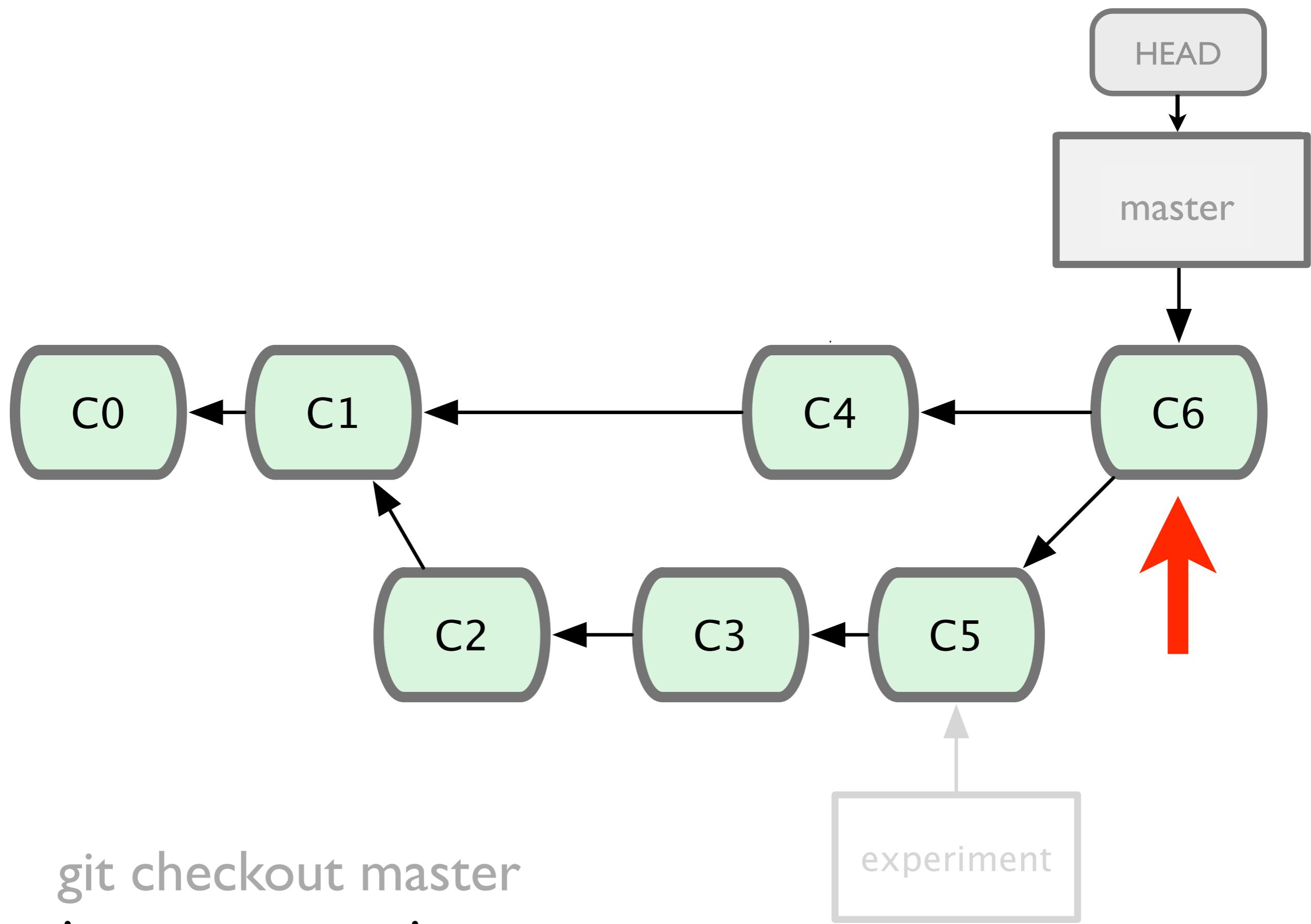


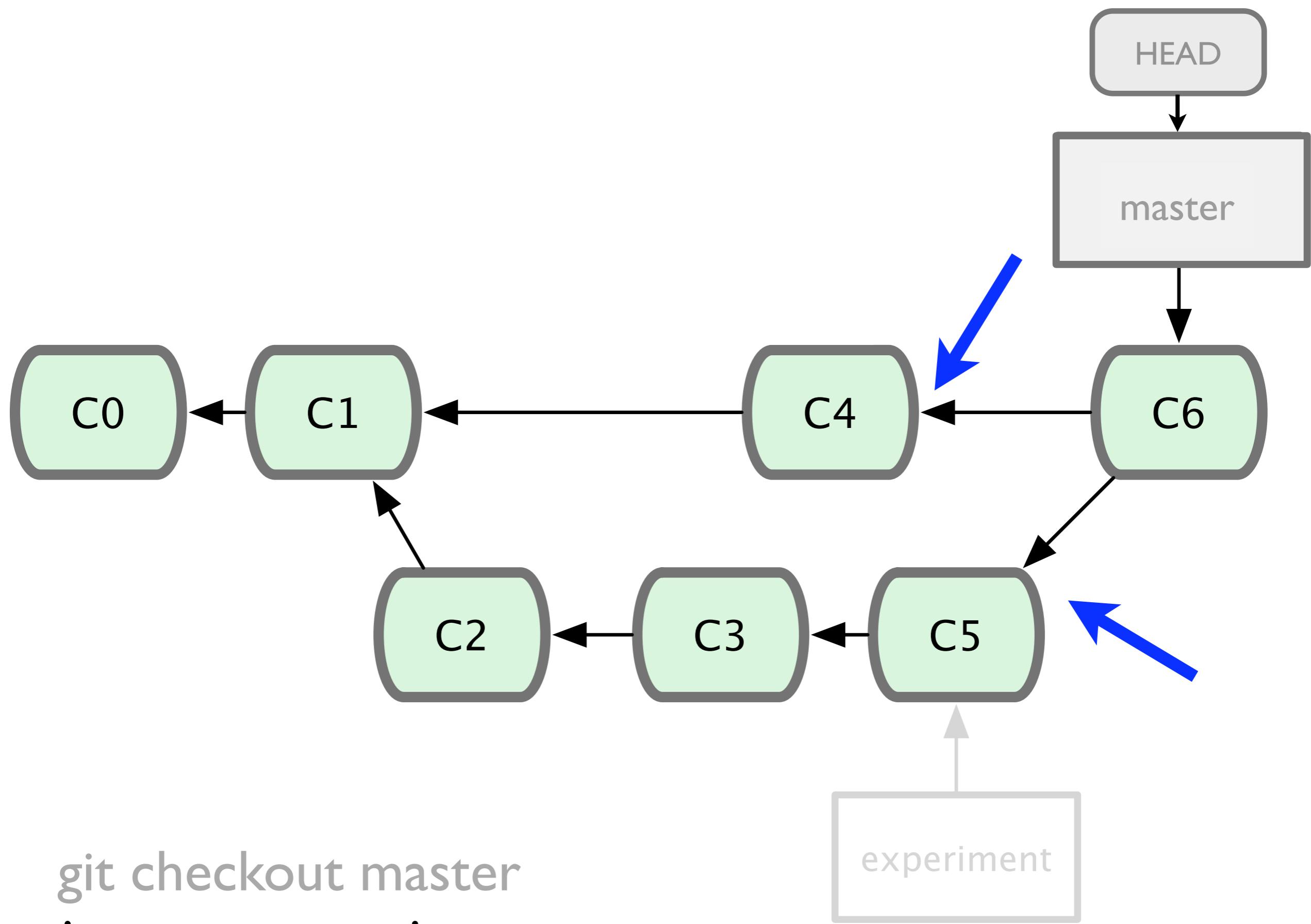
merging

git merge

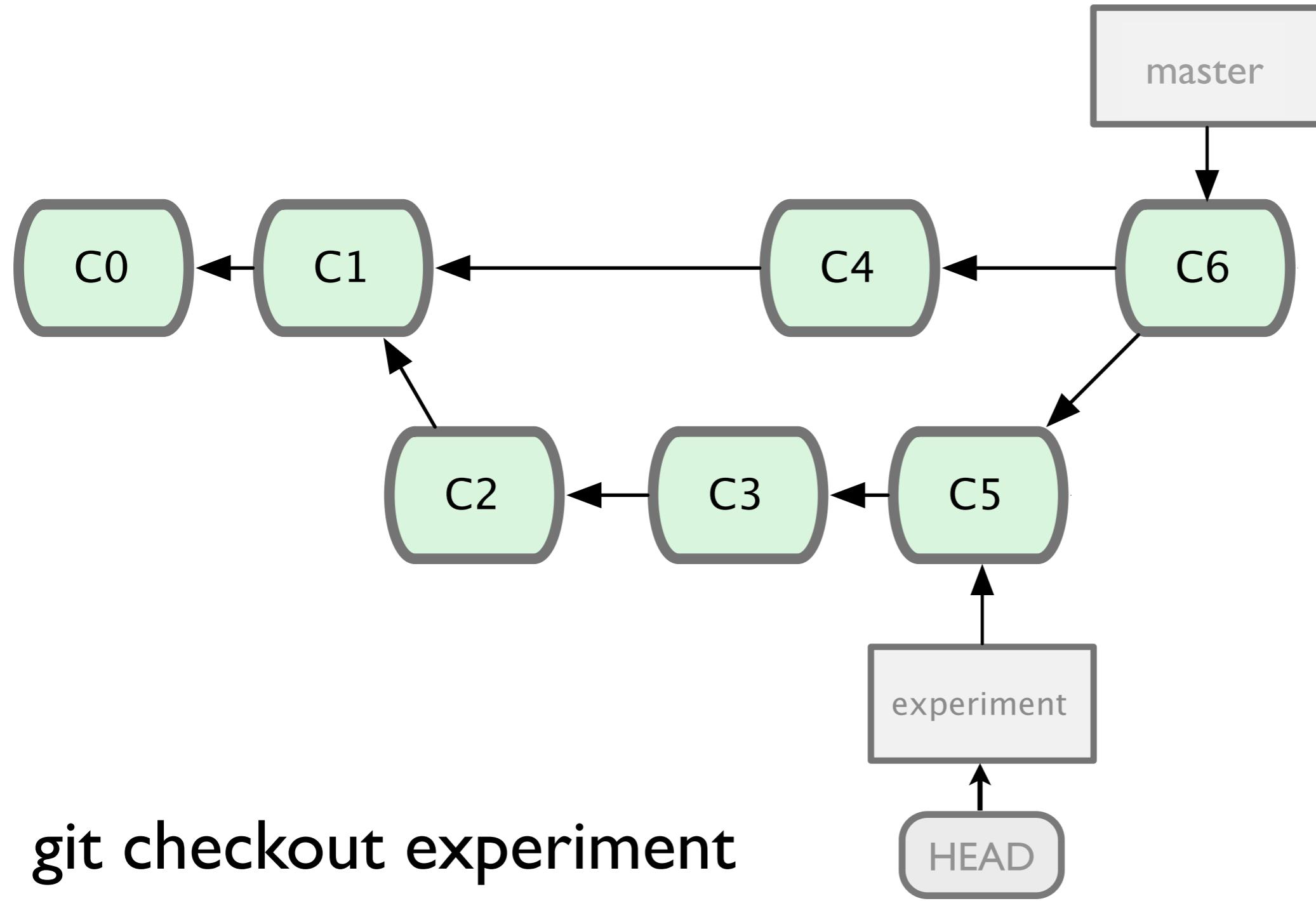


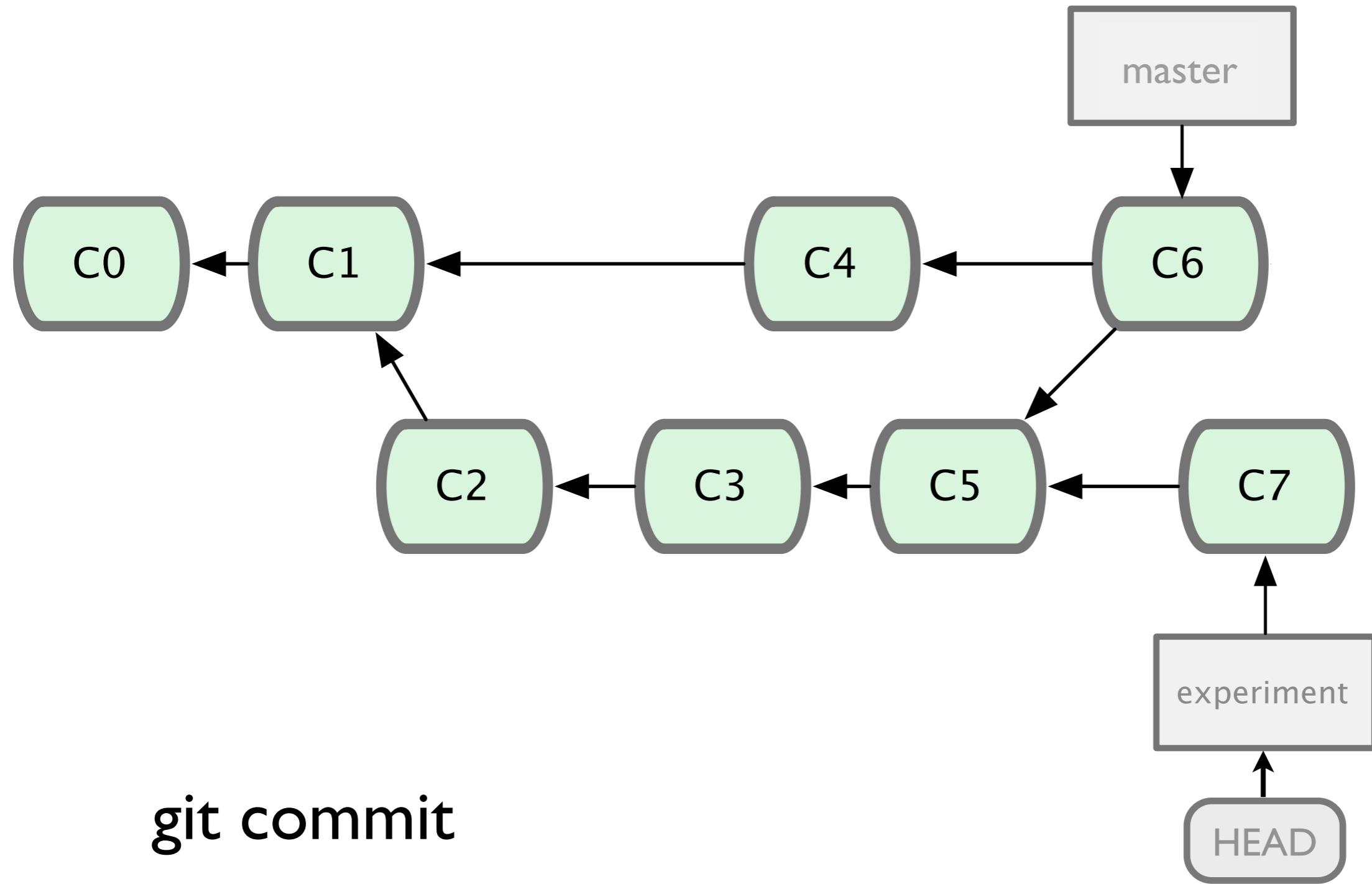


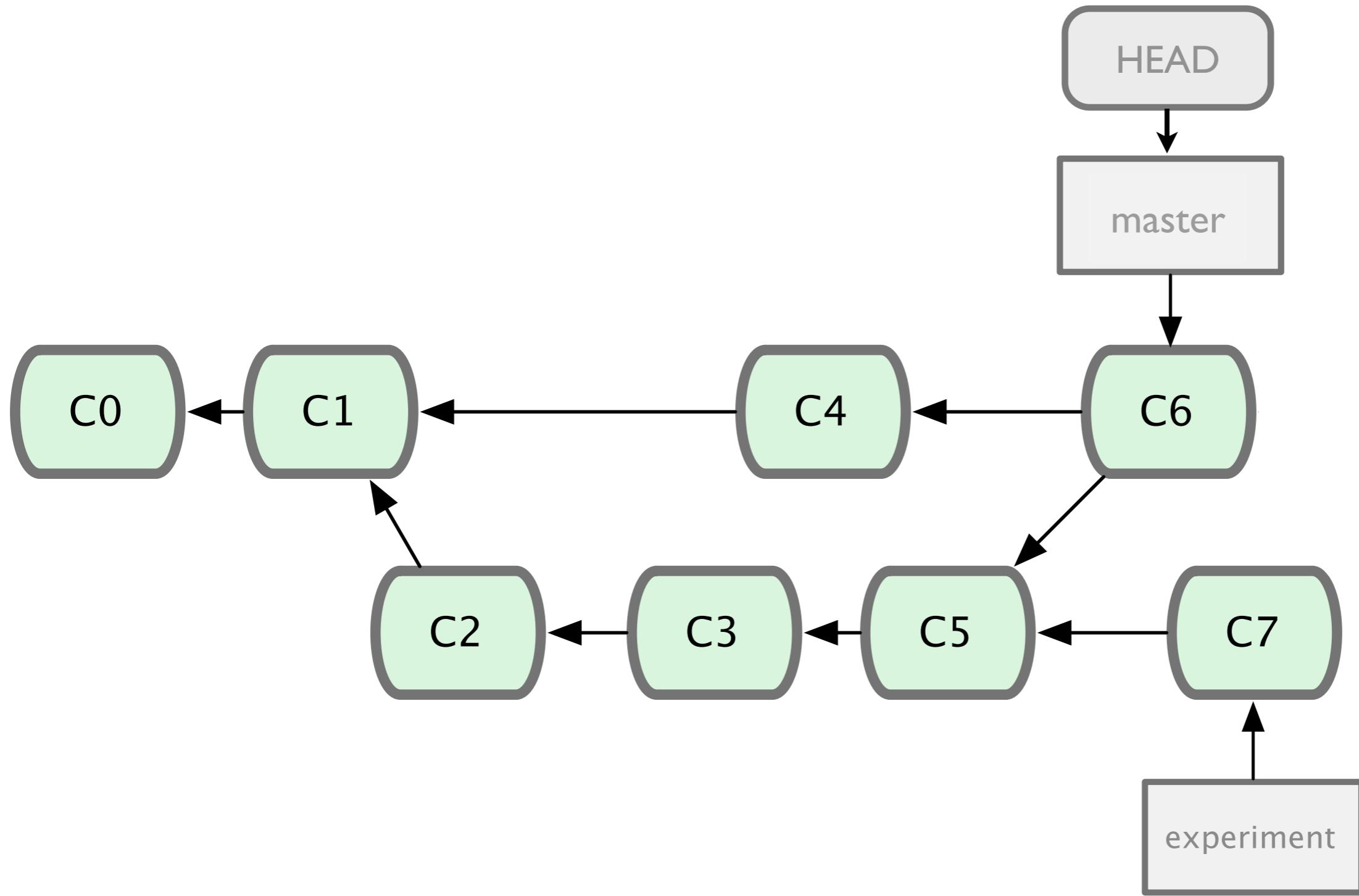




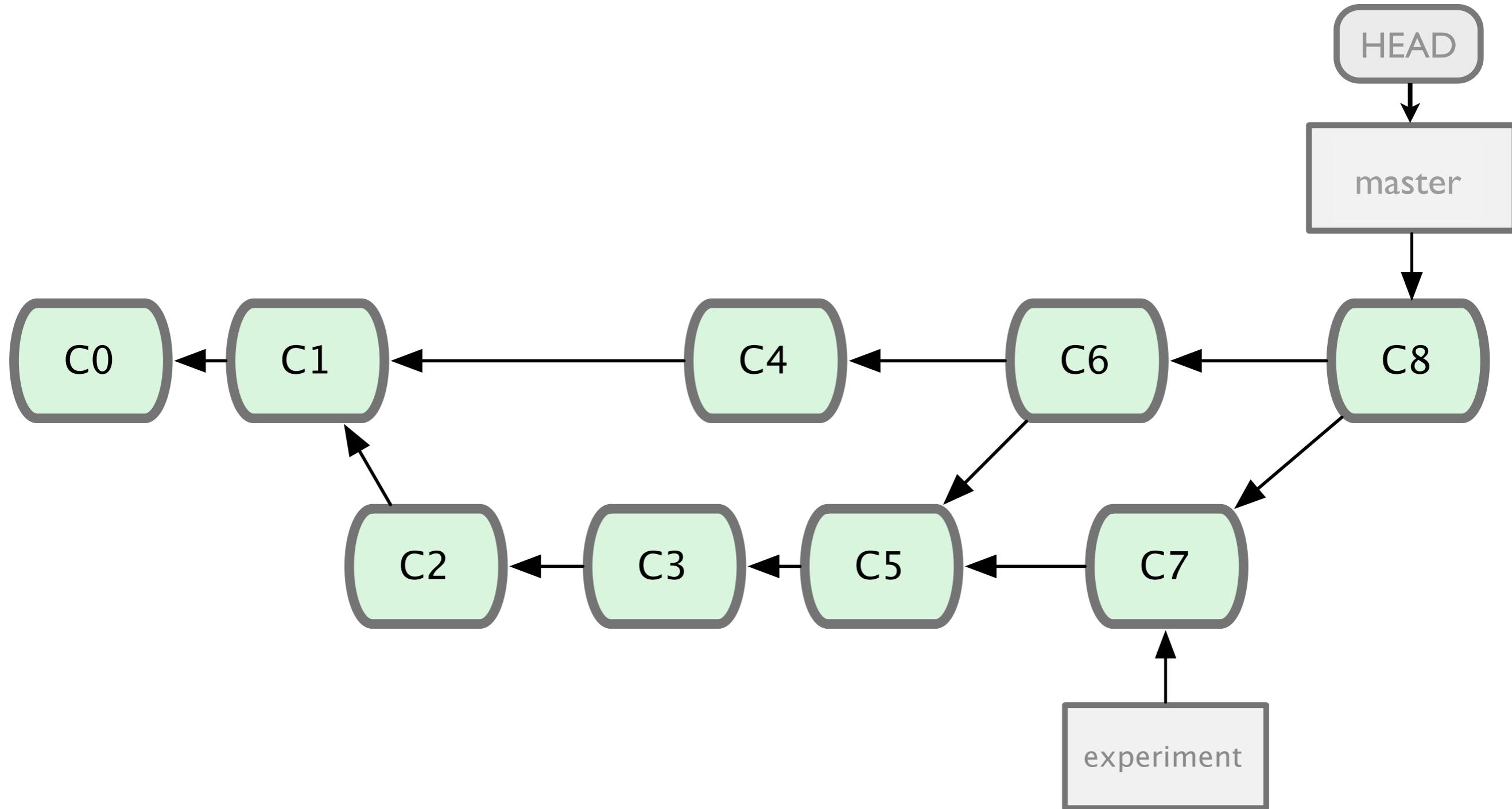
git checkout master
git merge experiment





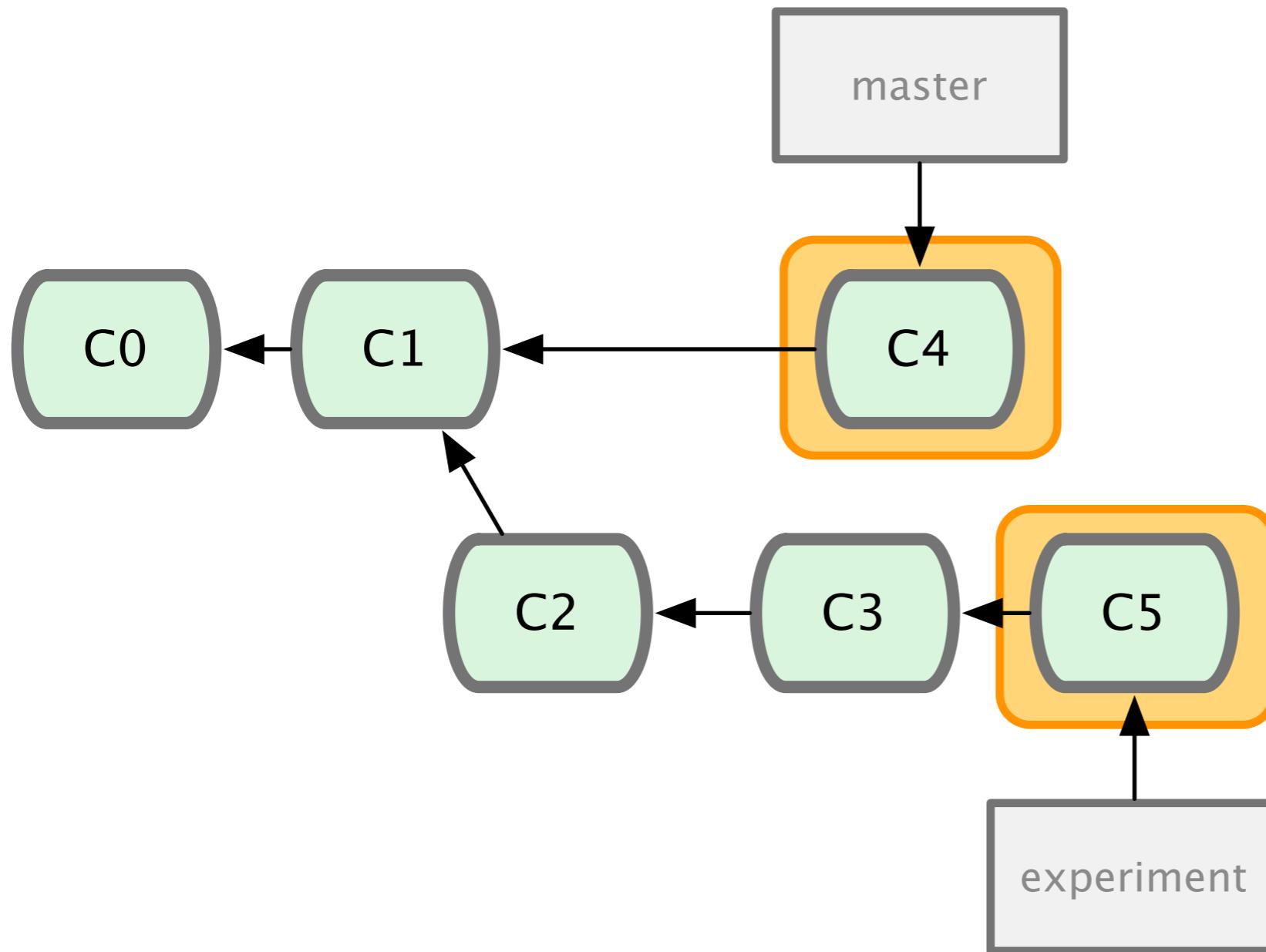


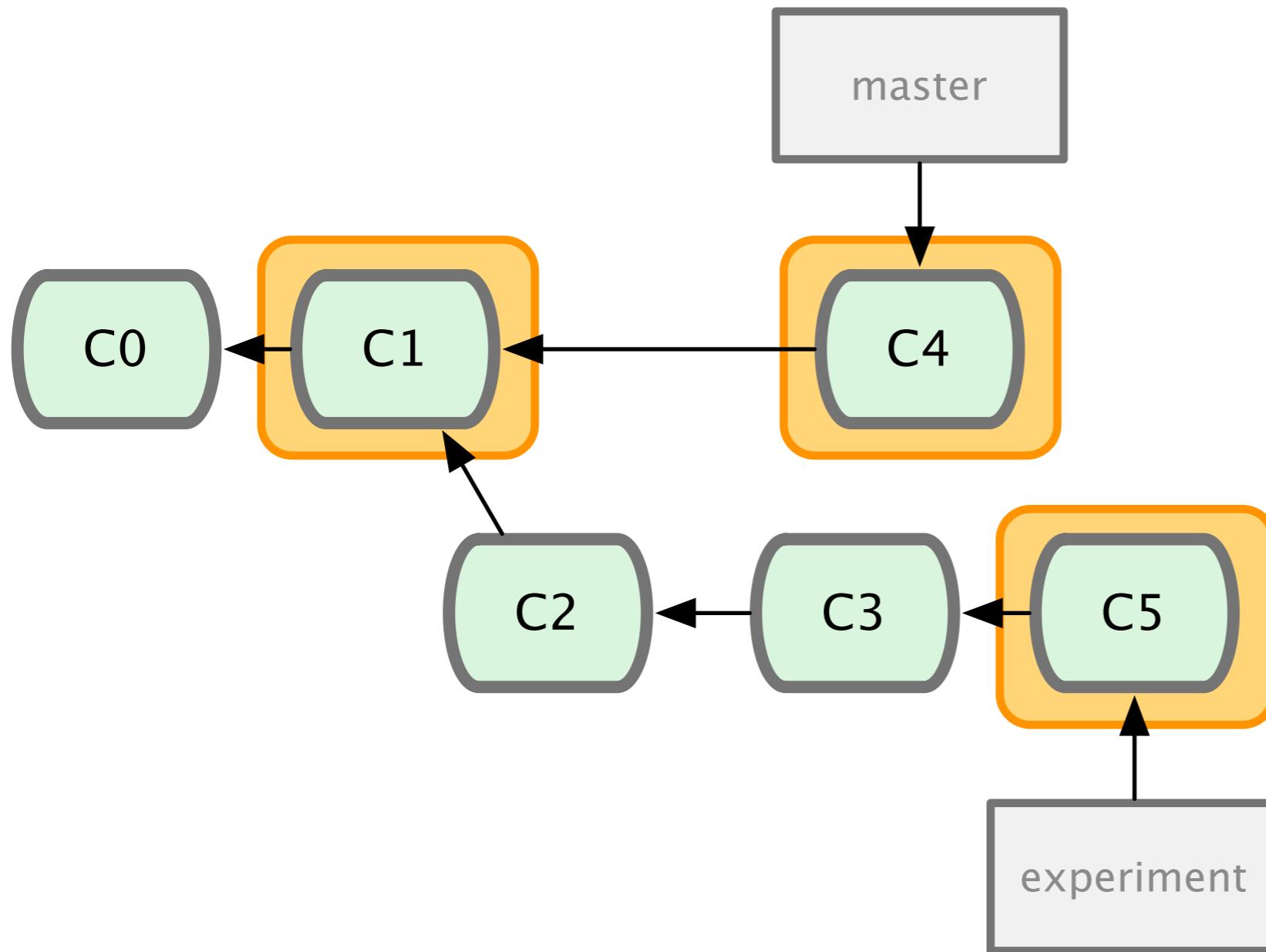
git checkout master

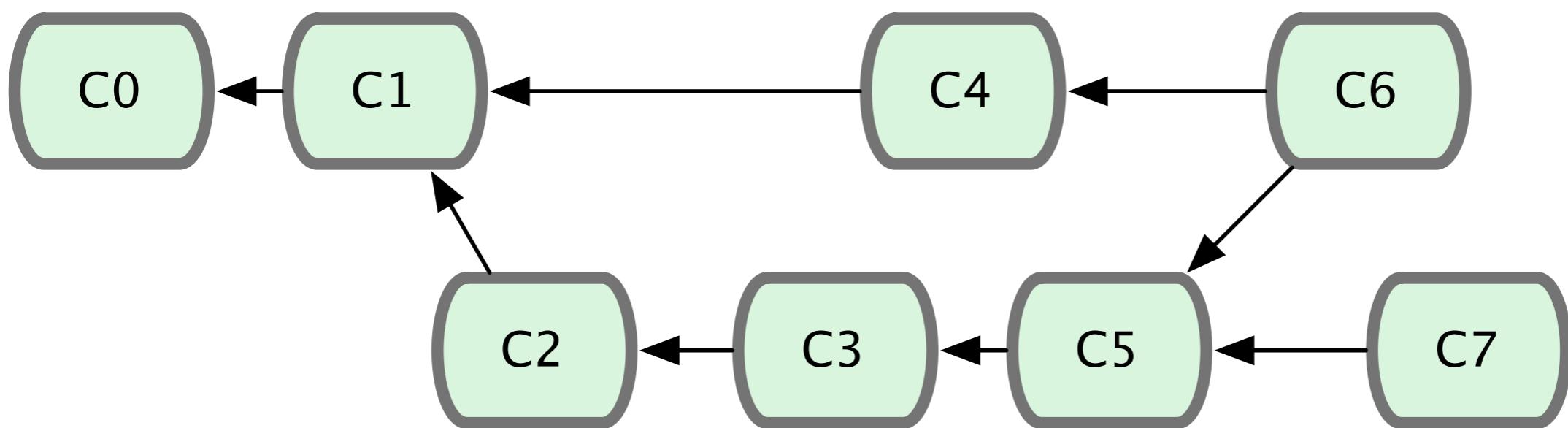


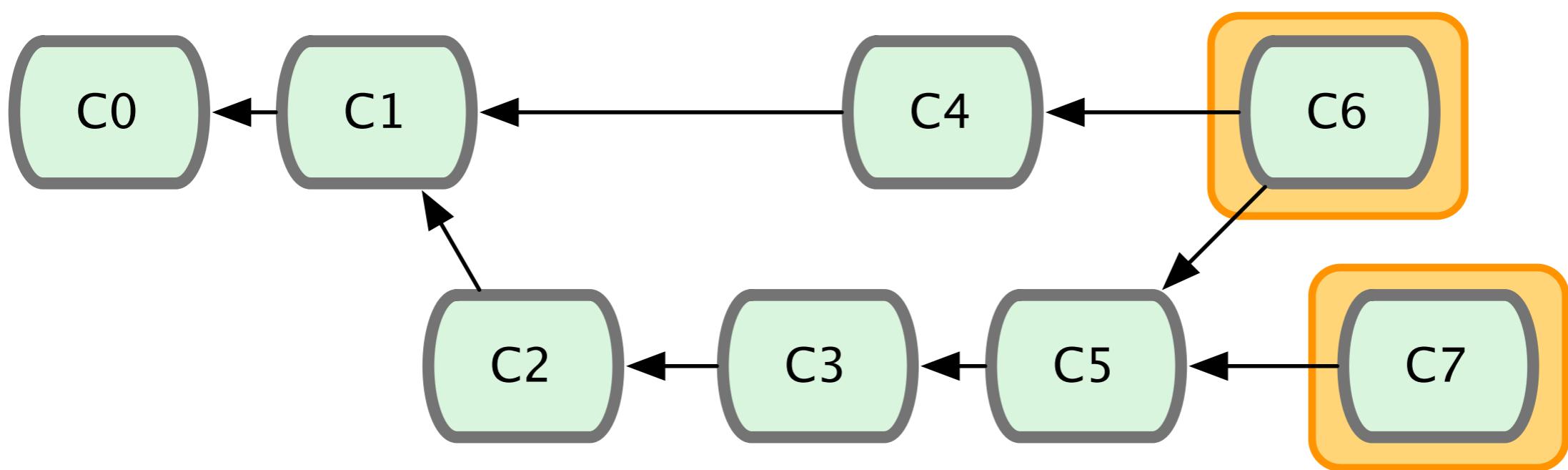
git merge experiment

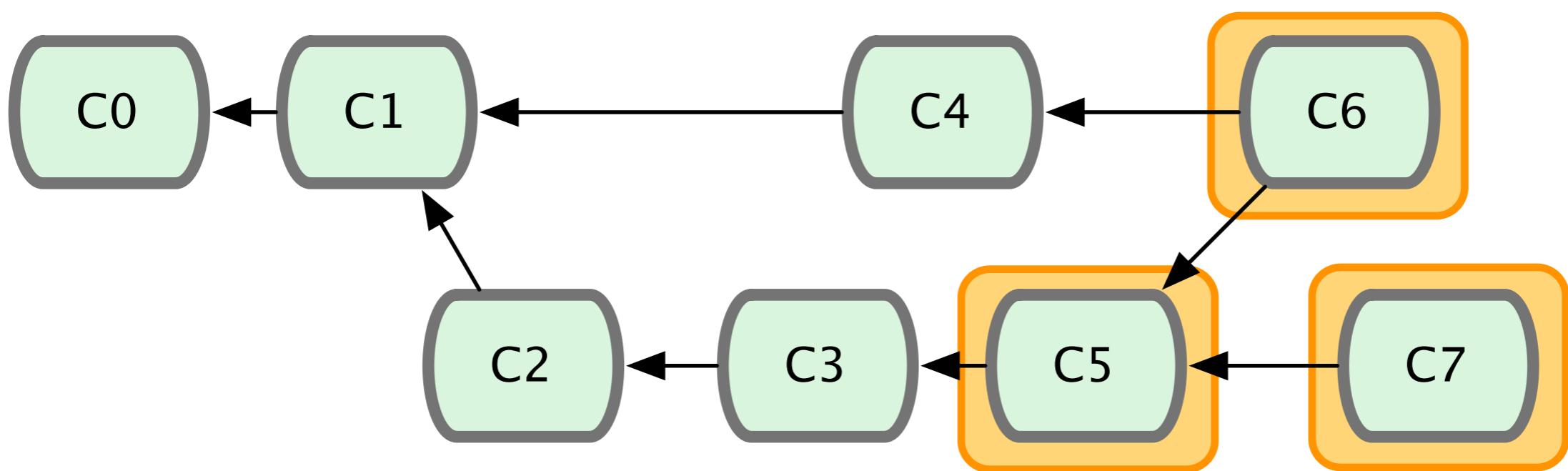
Hot
3 Way Merge
Action











Branching as a Patch Queue

you want to

you want to

work on one topic

you want to

work on one topic

revert to a base state

you want to

work on one topic

revert to a base state

work on another, unrelated topic

you want to

work on one topic

revert to a base state

work on another, unrelated topic

revert to a base state

you want to

work on one topic

revert to a base state

work on another, unrelated topic

revert to a base state

continue working on the first topic

you want to

work on one topic

revert to a base state

work on another, unrelated topic

revert to a base state

continue working on the first topic

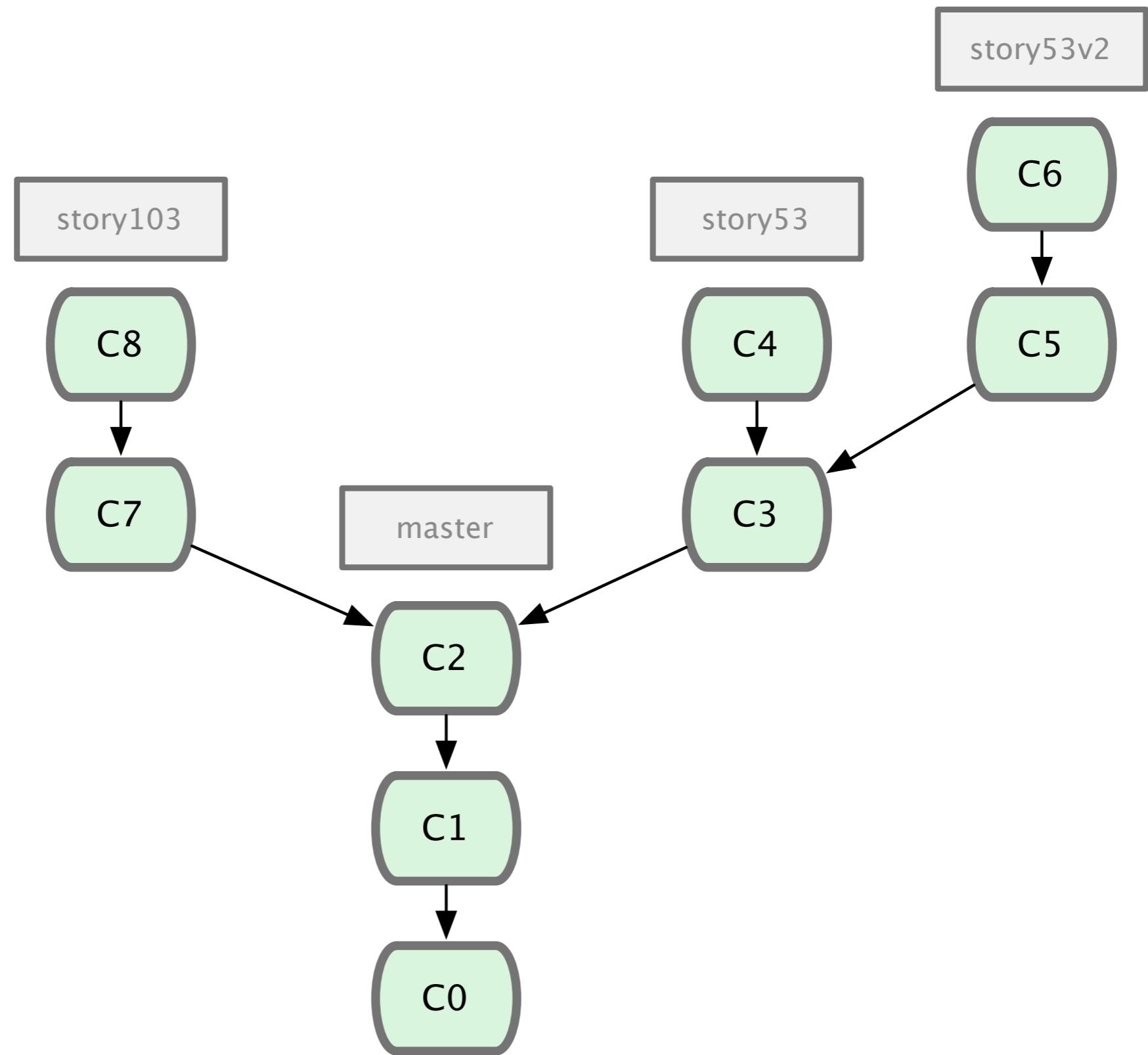
export either as a series of patches to the
base state for a maintainer

quilt

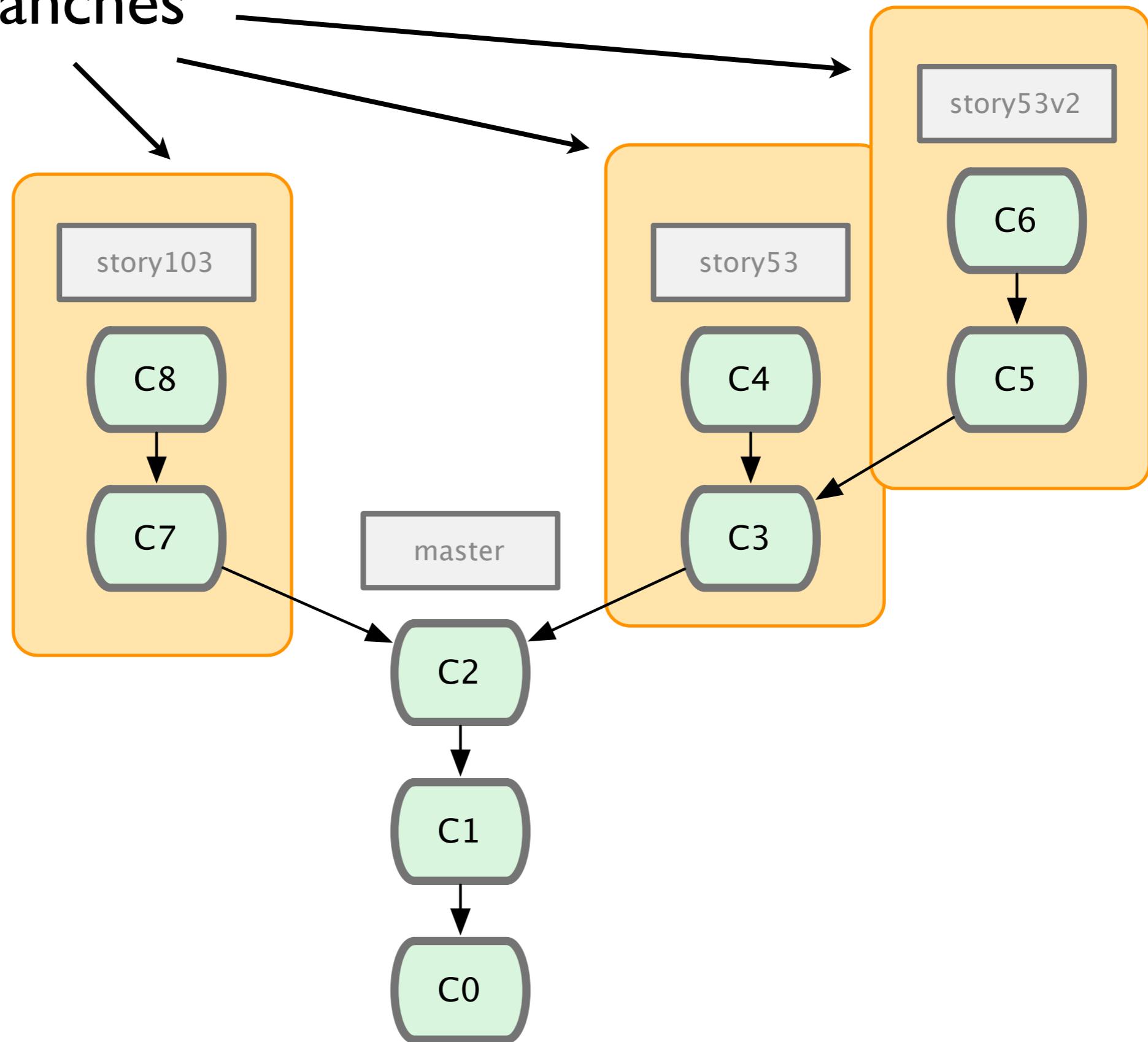
mercurial's mq extension

qapplied	print the patches already applied
qclone	clone main and patch repository at same time
qcommit	commit changes in the queue repository
qdelete	remove patches from queue
qdiff	diff of the current patch and subsequent modifications
qfinish	move applied patches into repository history
qfold	fold the named patches into the current patch
qgoto	push or pop patches until named patch is at top of stack
qguard	set or print guards for a patch
qheader	Print the header of the topmost or specified patch
qimport	import a patch
qinit	init a new queue repository
qnew	create a new patch
qnext	print the name of the next patch
qpop	pop the current patch off the stack
qprev	print the name of the previous patch
qpush	push the next patch onto the stack
qrefresh	update the current patch
qrename	rename a patch
qrestore	restore the queue state saved by a rev
qsave	save current queue state
qselect	set or print guarded patches to push
qseries	print the entire series file
qtop	print the name of the current patch
qunapplied	print the patches not yet applied
strip	strip a revision and all its descendants from the repository

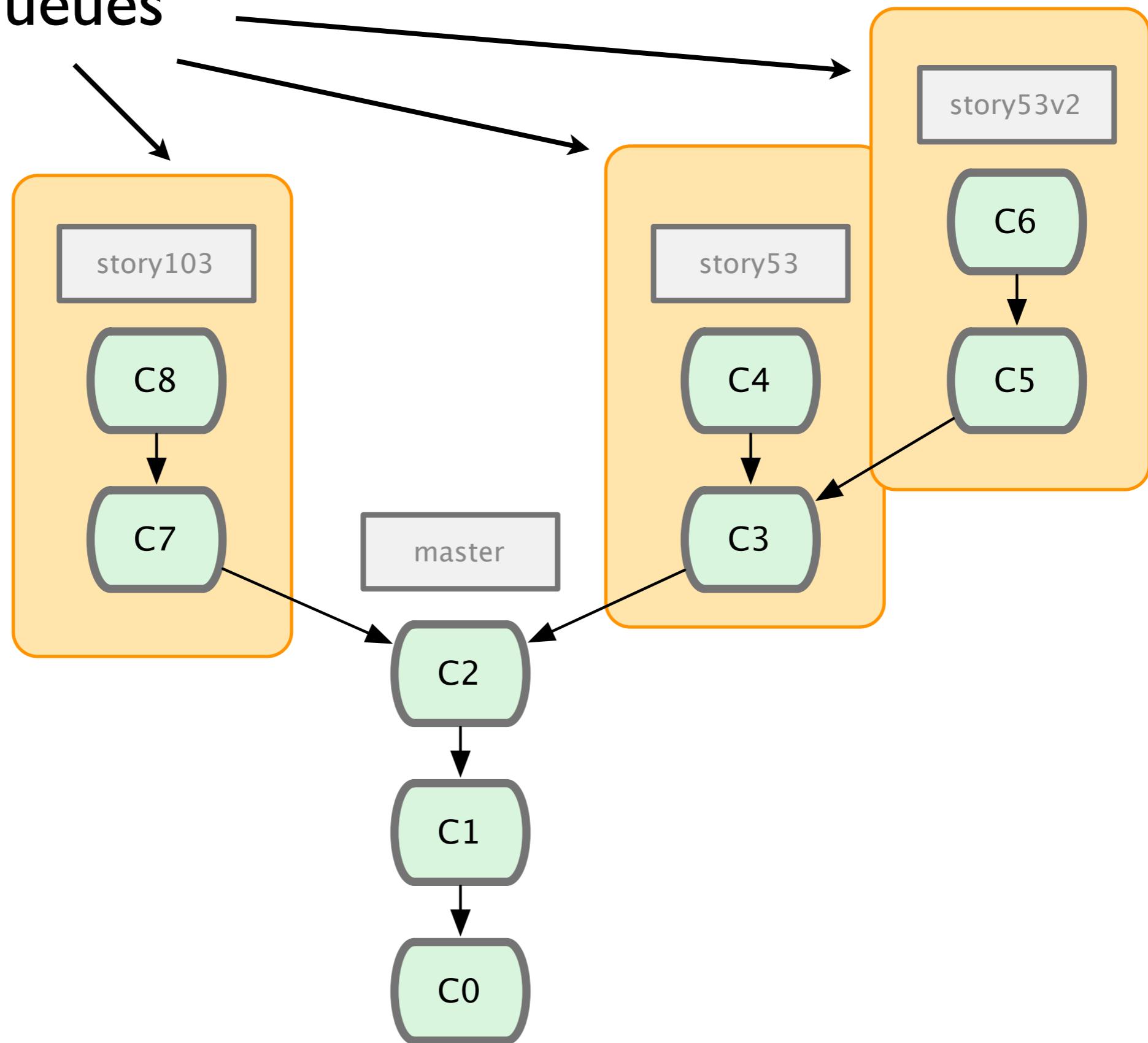
topic branches



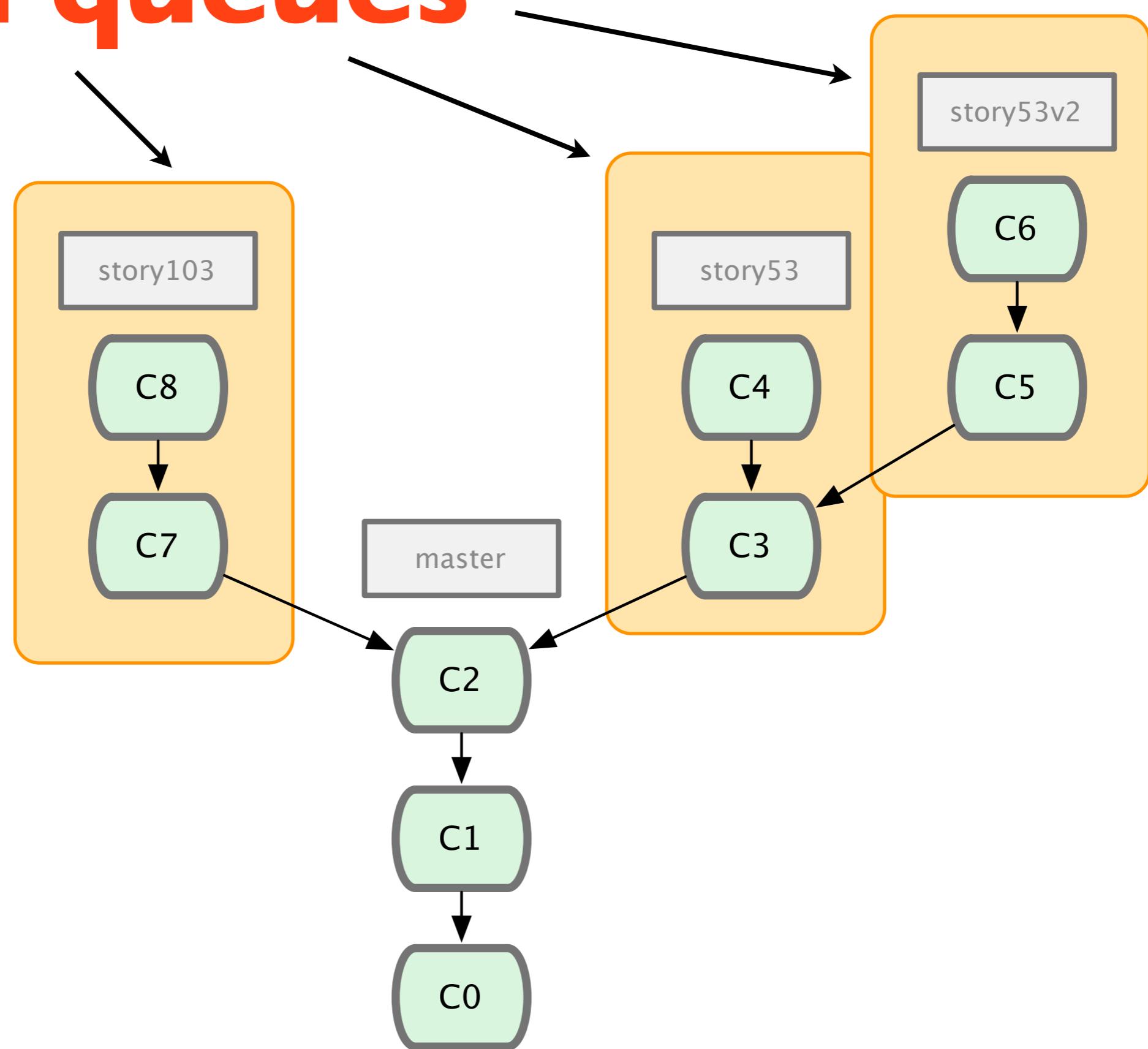
topic branches



patch queues



patch queues



qapplied	print the patches already applied
qclone	clone main and patch repository at same time
qcommit	commit changes in the queue repository
qdelete	remove patches from queue
qdiff	diff of the current patch and subsequent modifications
qfinish	move applied patches into repository history
qfold	fold the named patches into the current patch
qgoto	push or pop patches until named patch is at top of stack
qguard	set or print guards for a patch
qheader	Print the header of the topmost or specified patch
qimport	import a patch
qinit	init a new queue repository
qnew	create a new patch
qnext	print the name of the next patch
qpop	pop the current patch off the stack
qprev	print the name of the previous patch
qpush	push the next patch onto the stack
qrefresh	update the current patch
qrename	rename a patch
qrestore	restore the queue state saved by a rev
qsave	save current queue state
qselect	set or print guarded patches to push
qseries	print the entire series file
qtop	print the name of the current patch
qunapplied	print the patches not yet applied
strip	strip a revision and all its descendants from the repository

```
git branch <branch-name>
```

```
git checkout <branch-name>
```

```
git merge <branch-name>
```

```
git branch <patch-queue-name>
```

```
git checkout <patch-queue-name>
```

```
git merge <patch-queue-name>
```

patch queueing related
Git tools

git rebase

git format-patch

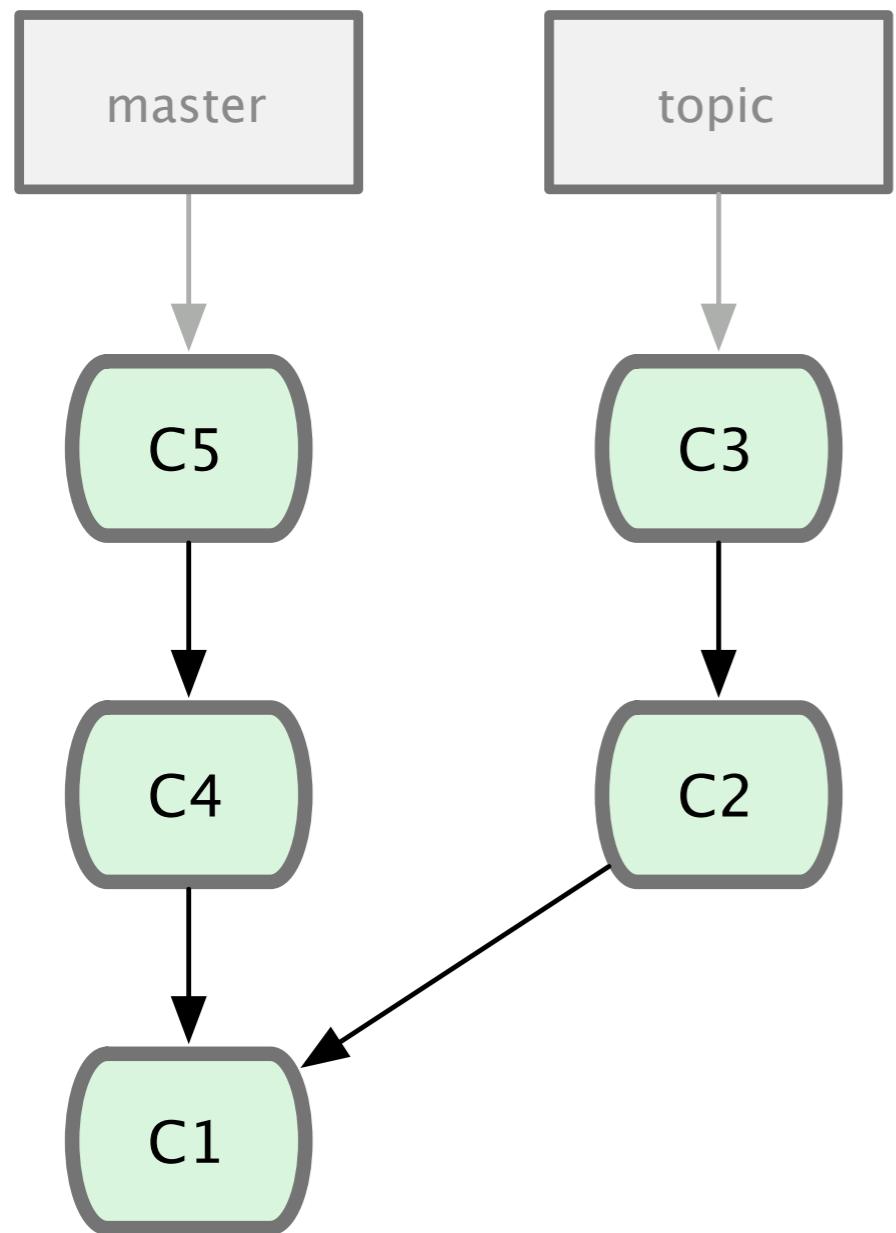
git am

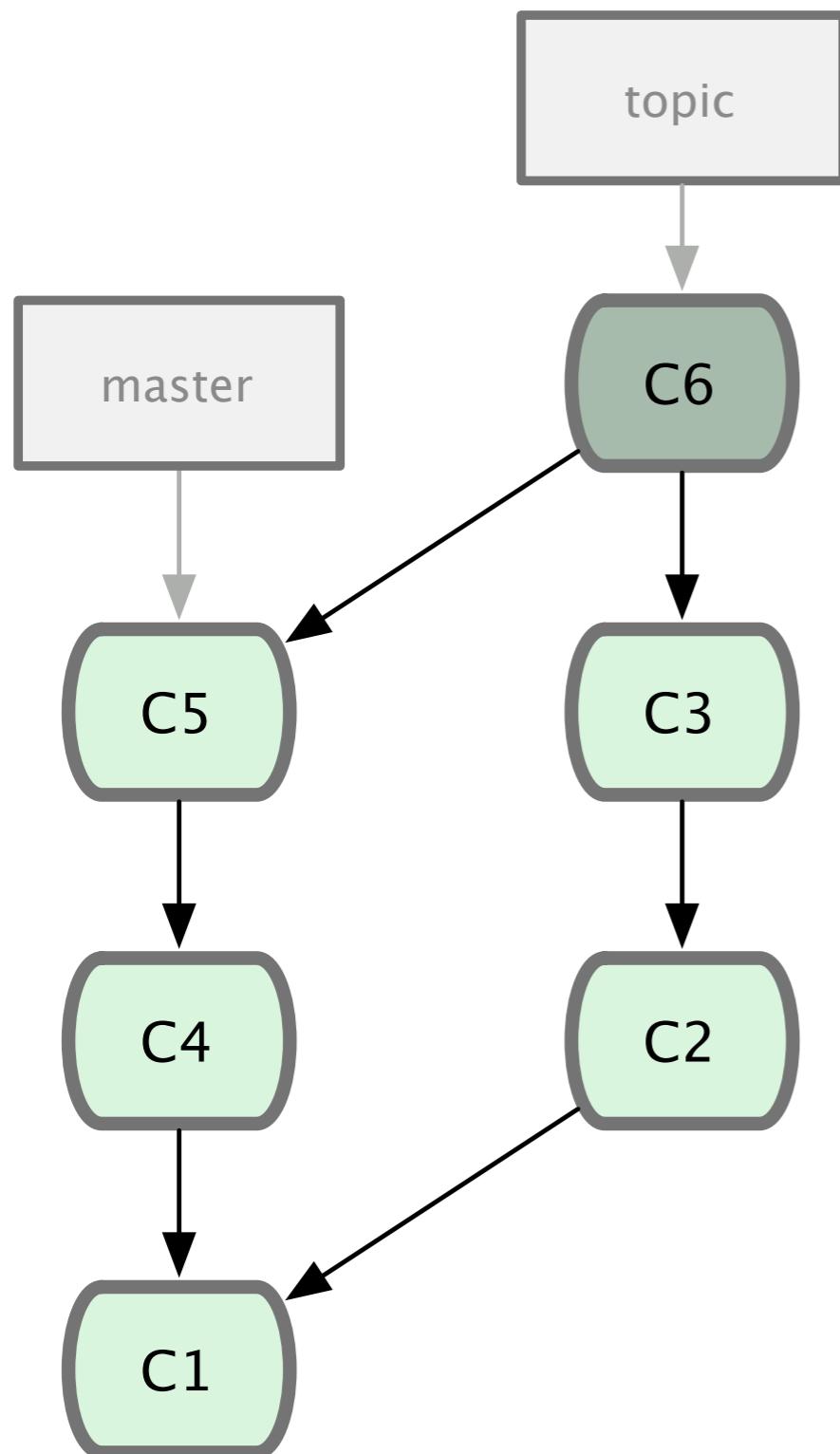
git rebase

git format-patch

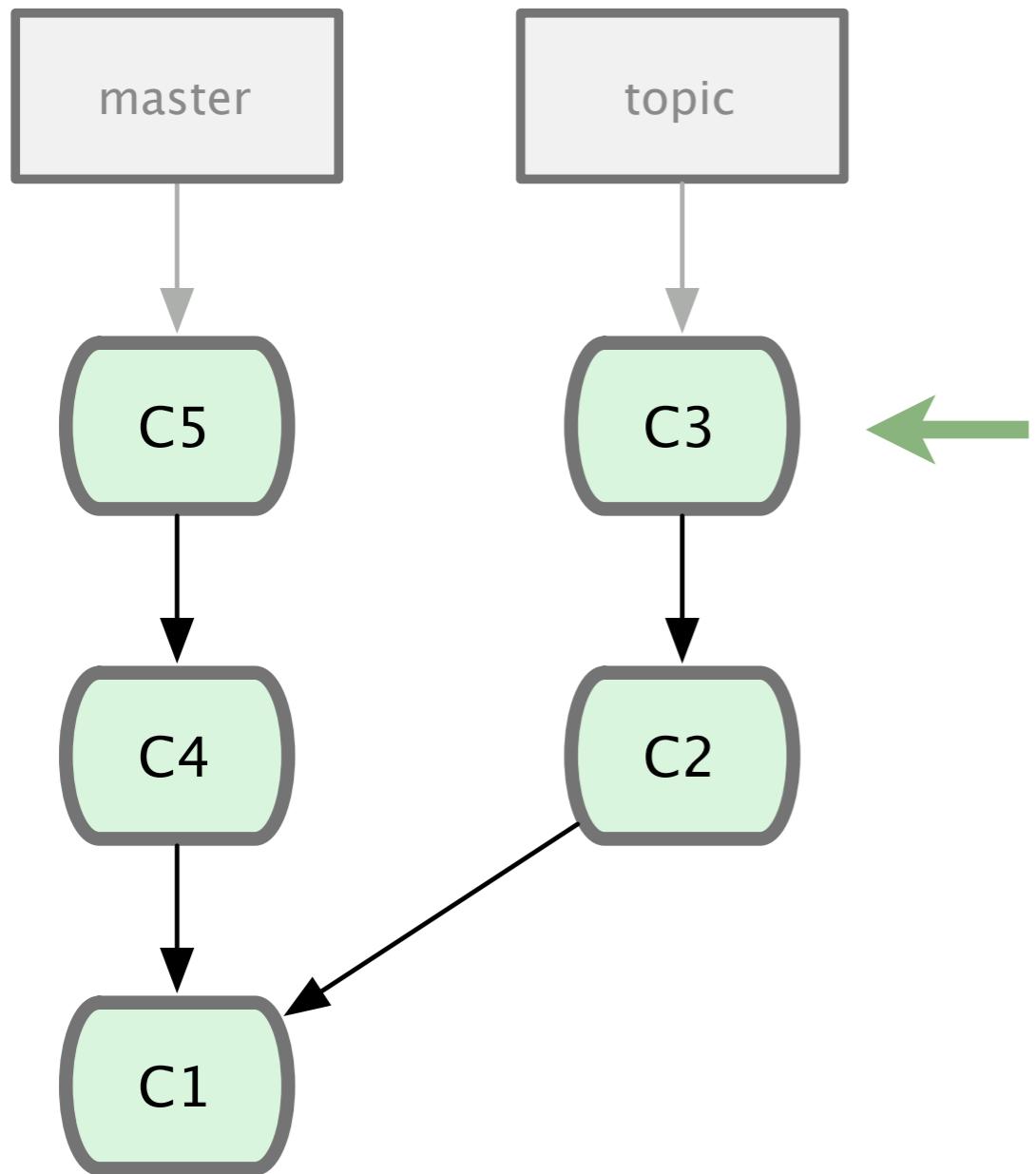
git am

Rebasing

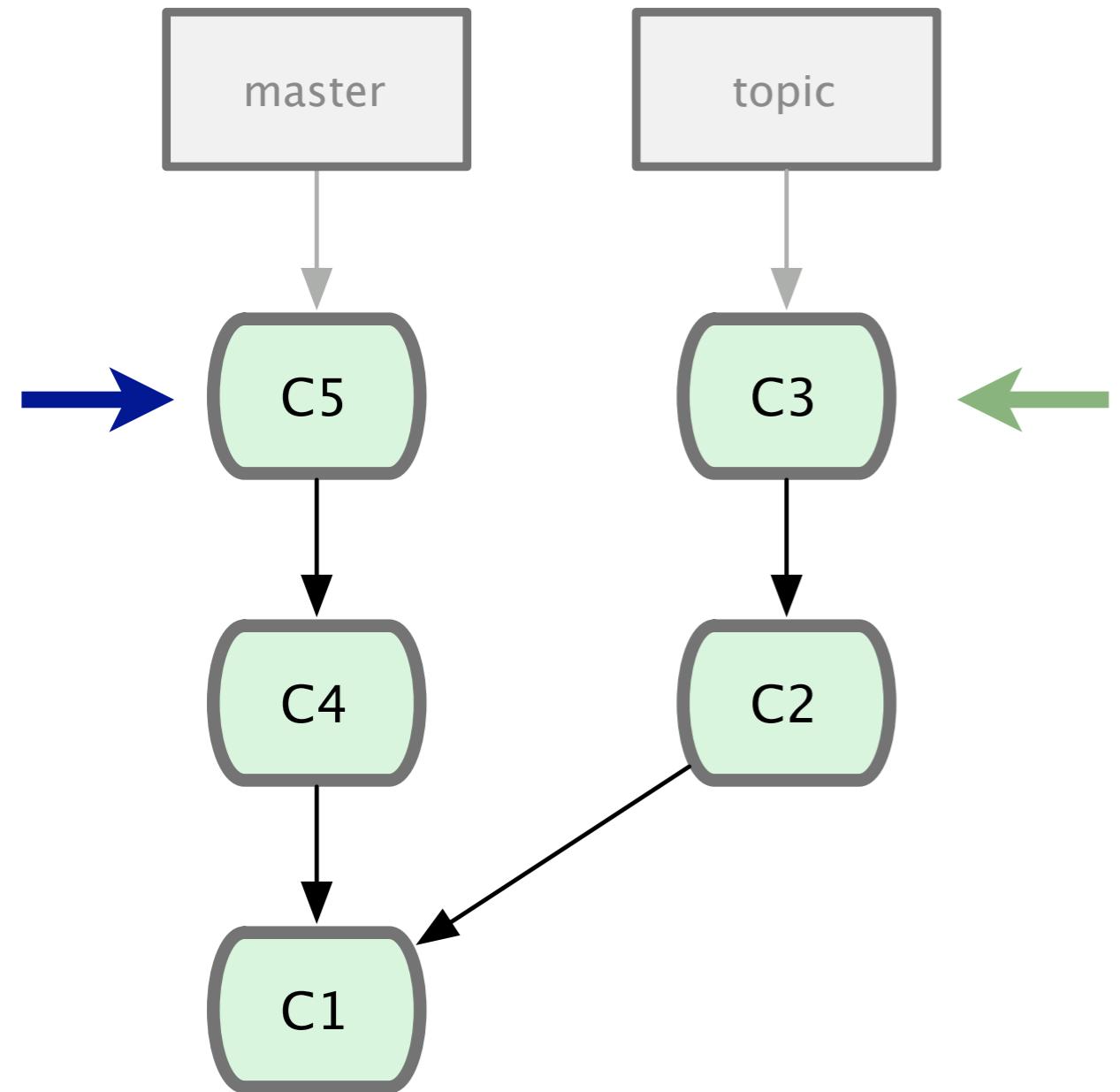




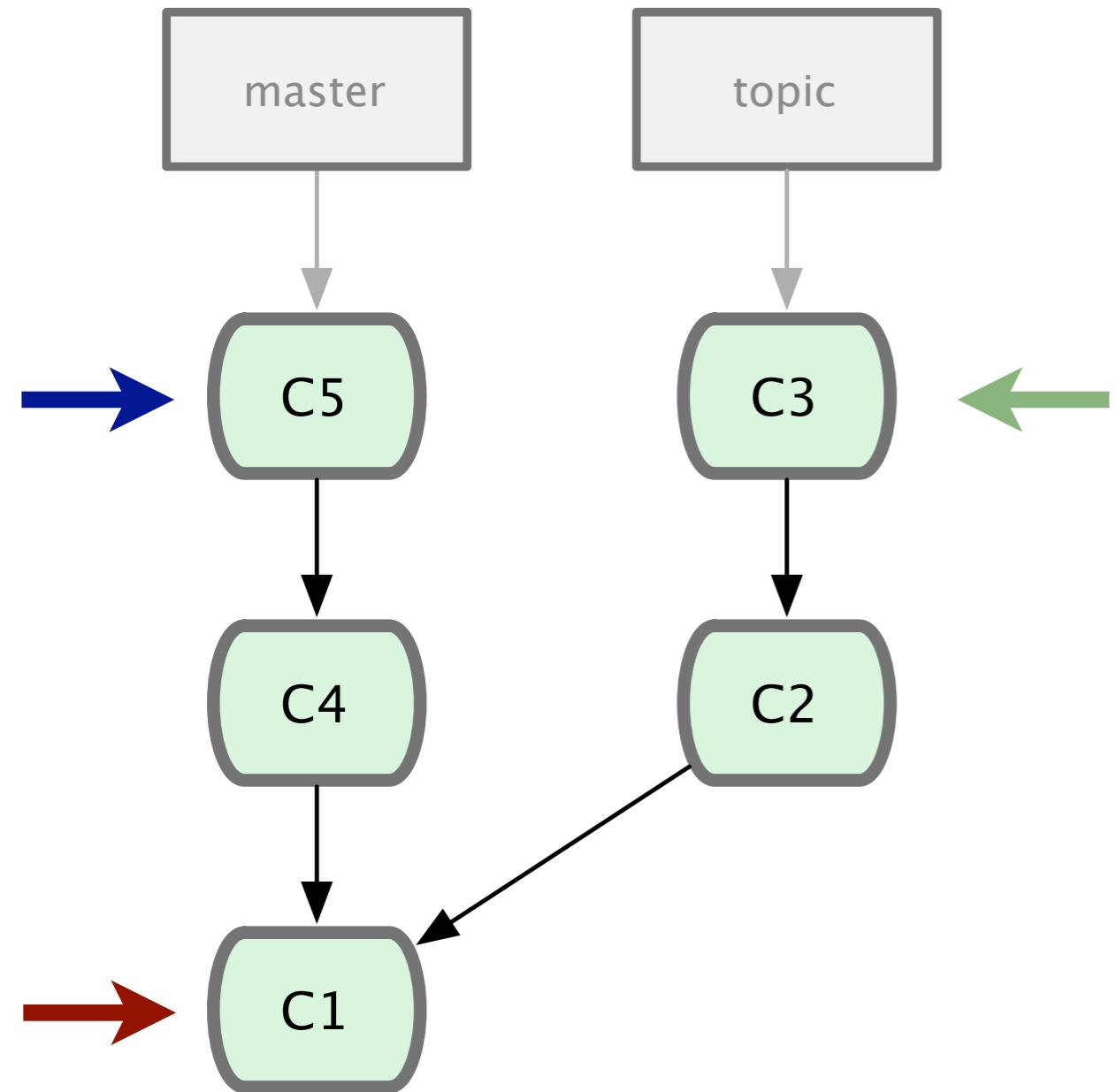
git merge master



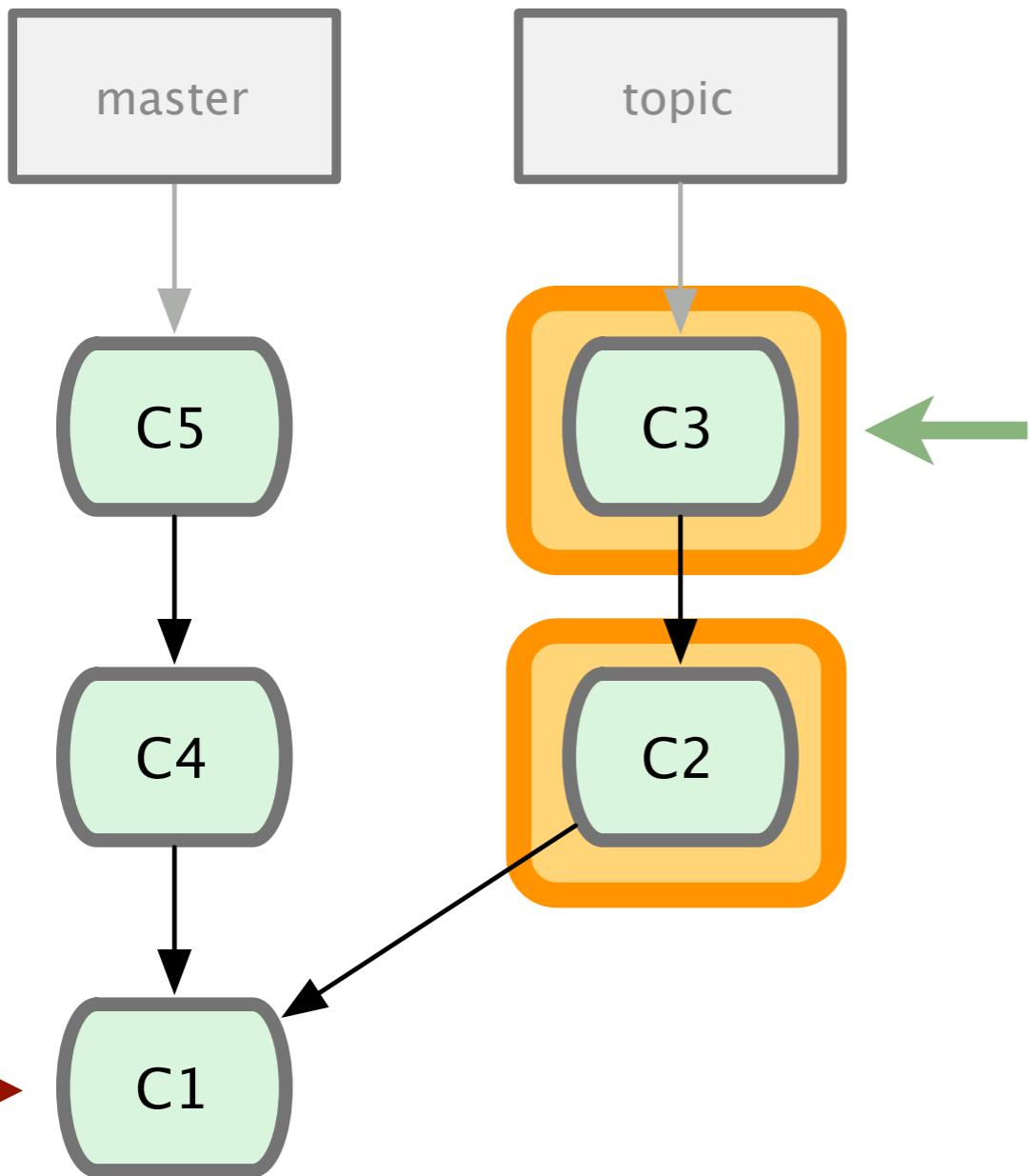
git rebase master



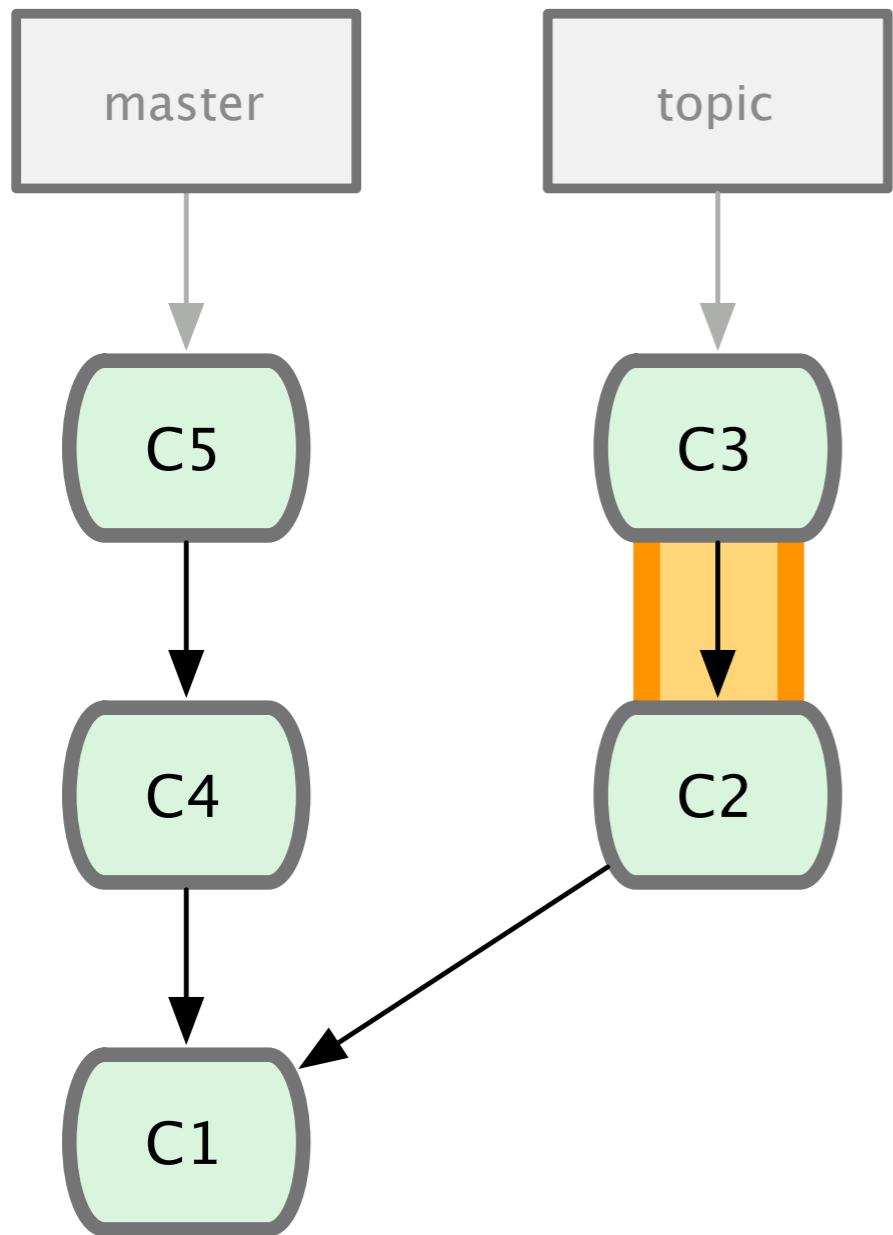
git rebase master



git rebase master

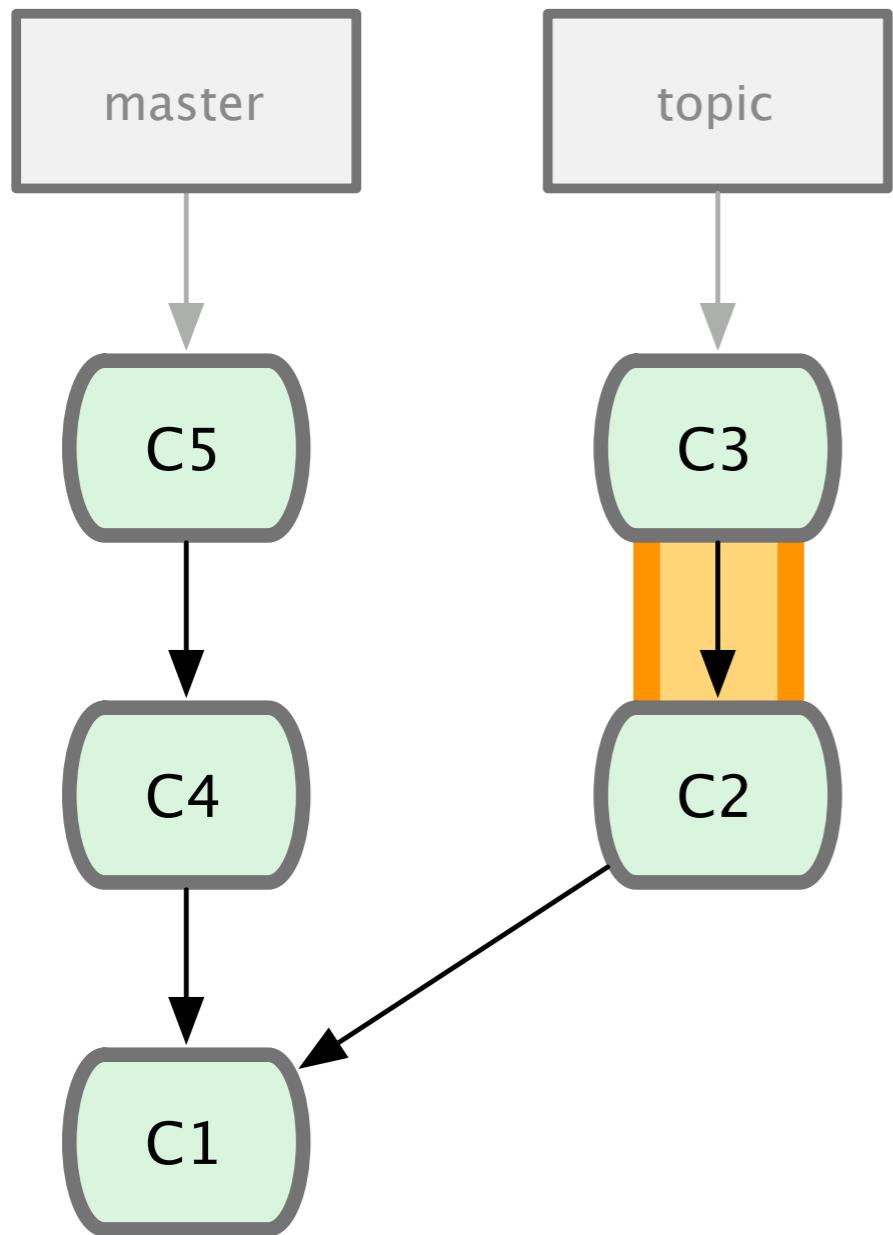


git rebase master



`git diff c2 c3 > 2-3.patch`

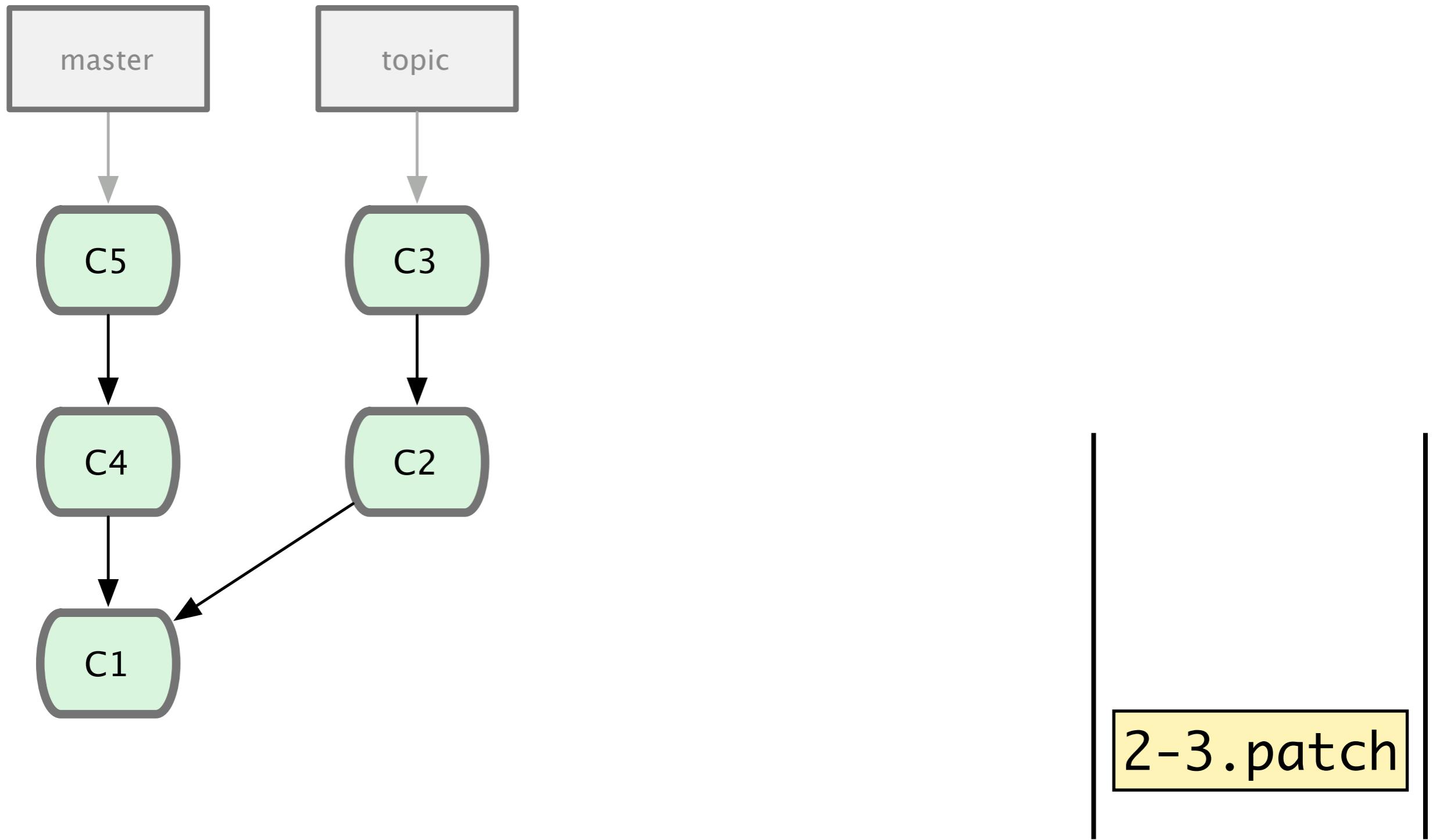
`git rebase master`

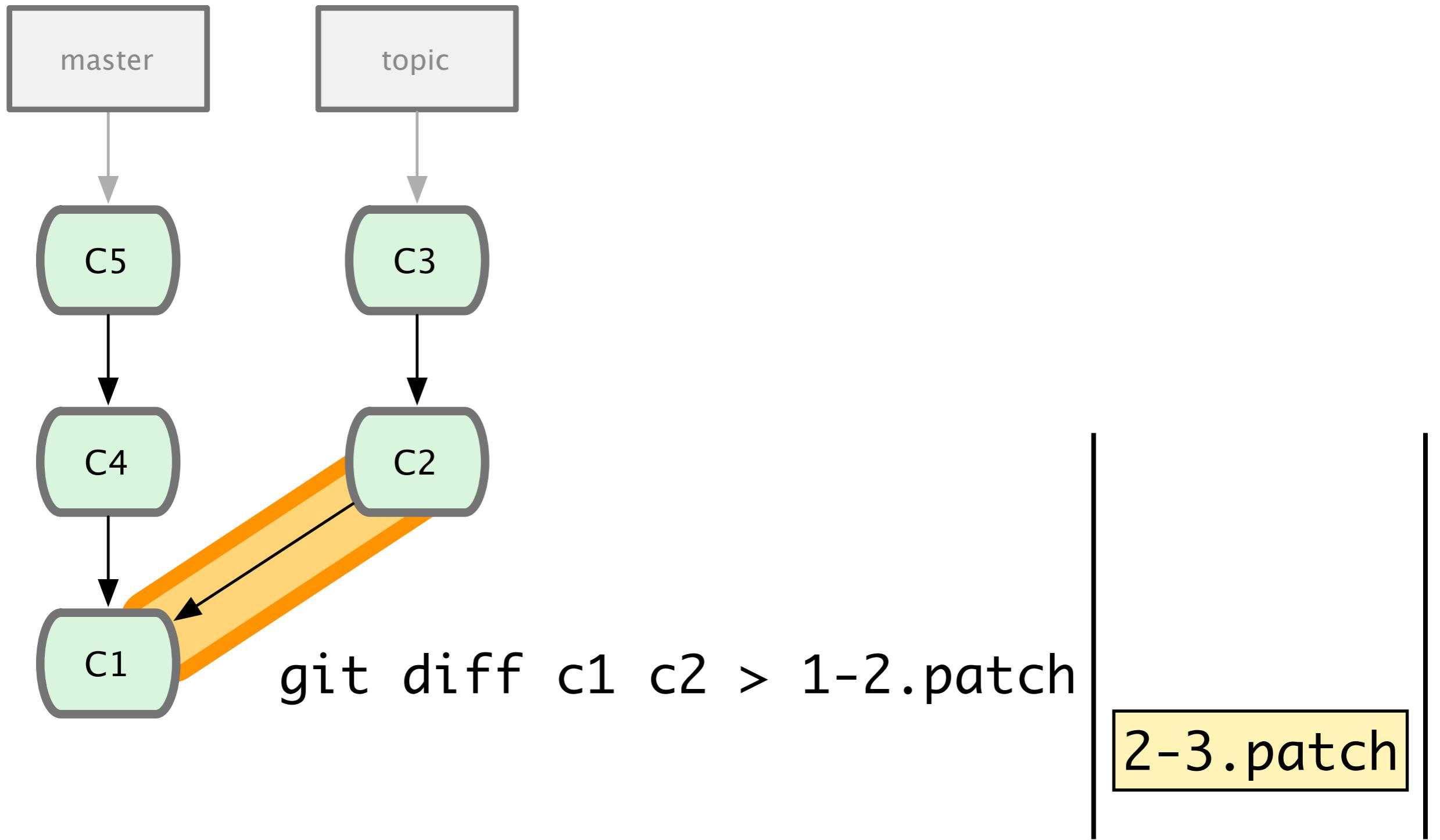


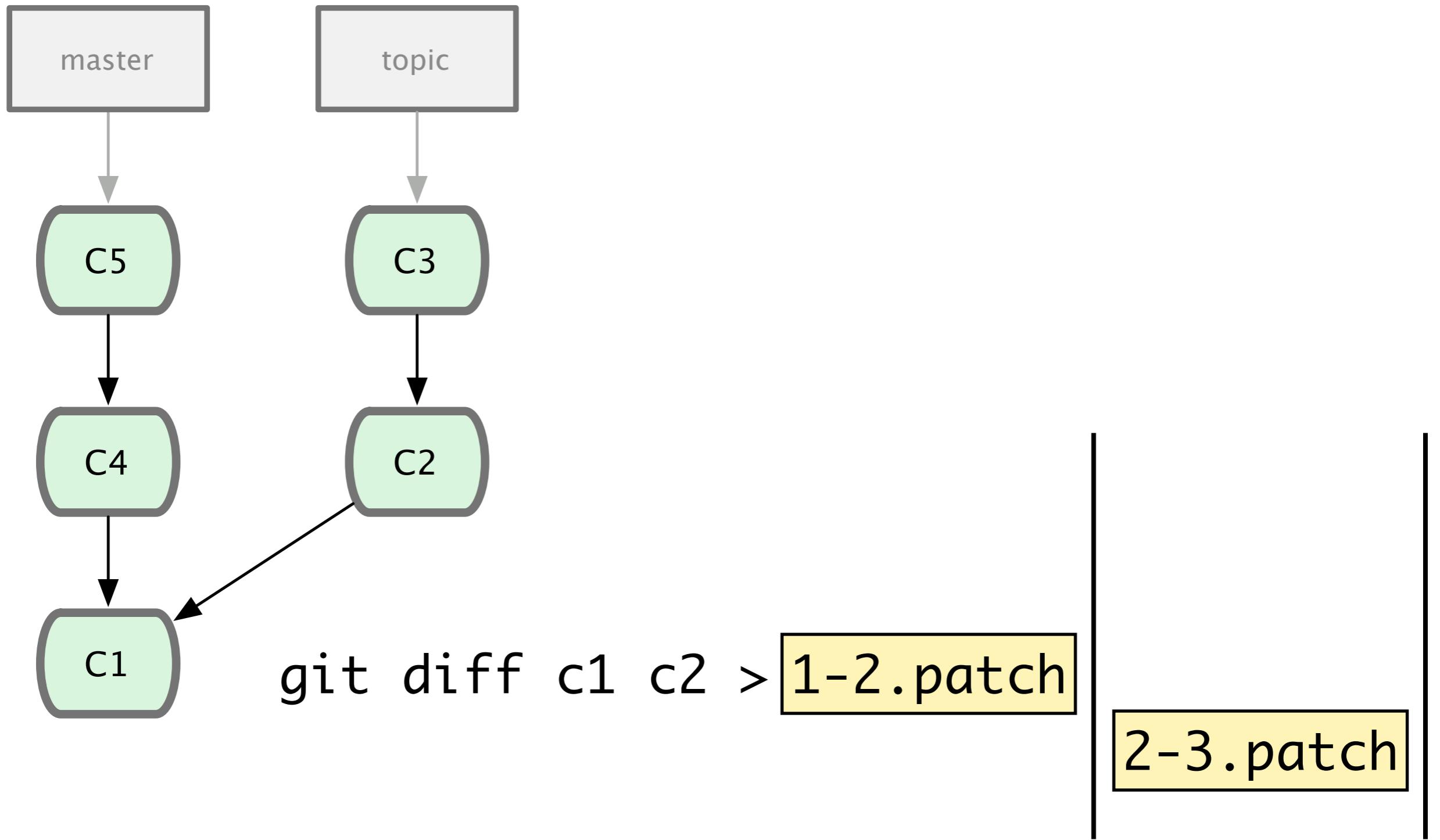
git diff c2 c3 > 2-3.patch

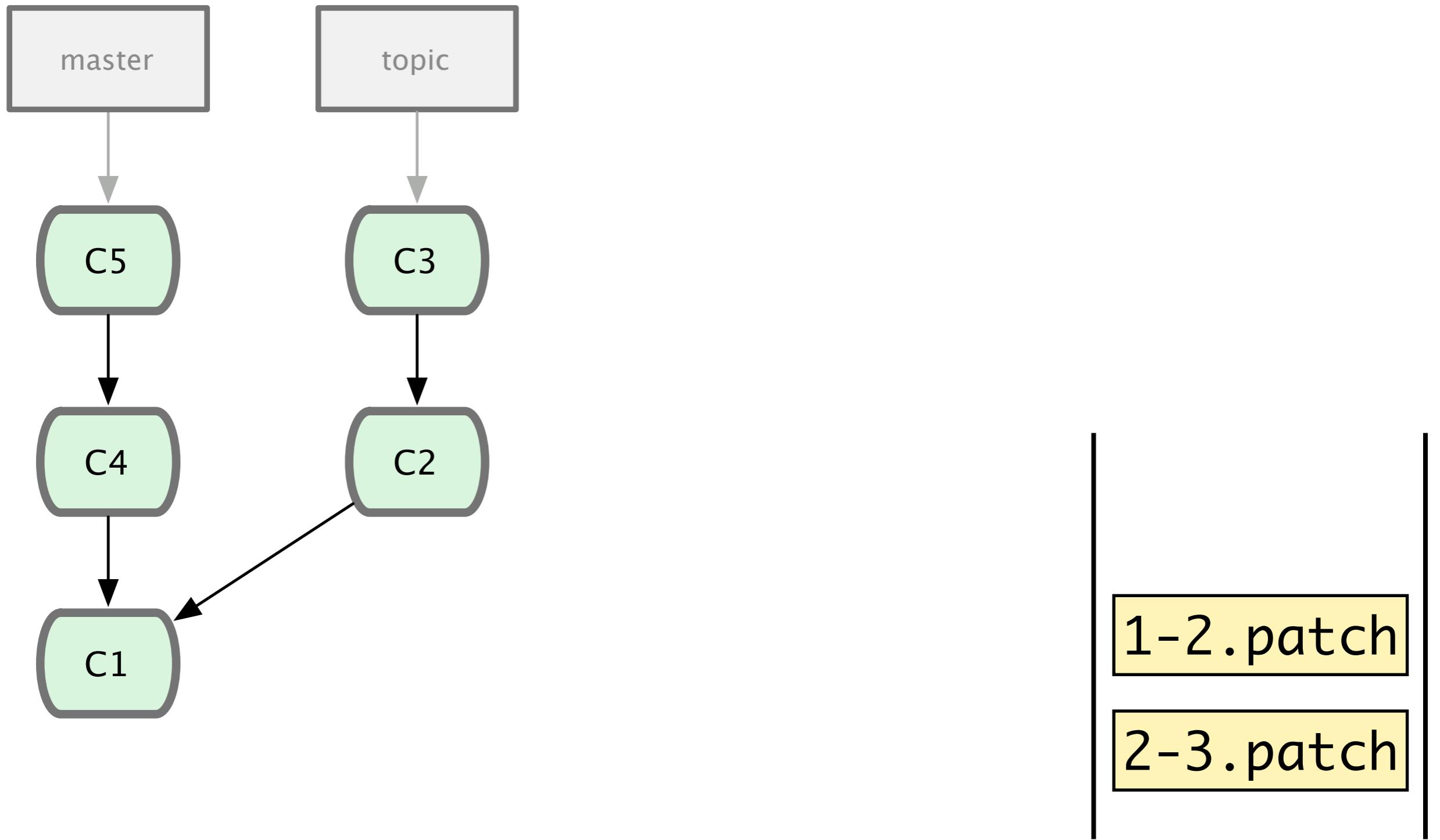
```
diff --git a/test b/test
index 2eadcec..bd8c6c9 100644
--- a/test
+++ b/test
@@ -1,2 +1,3 @@
version one
version four
+version five
```

git rebase master

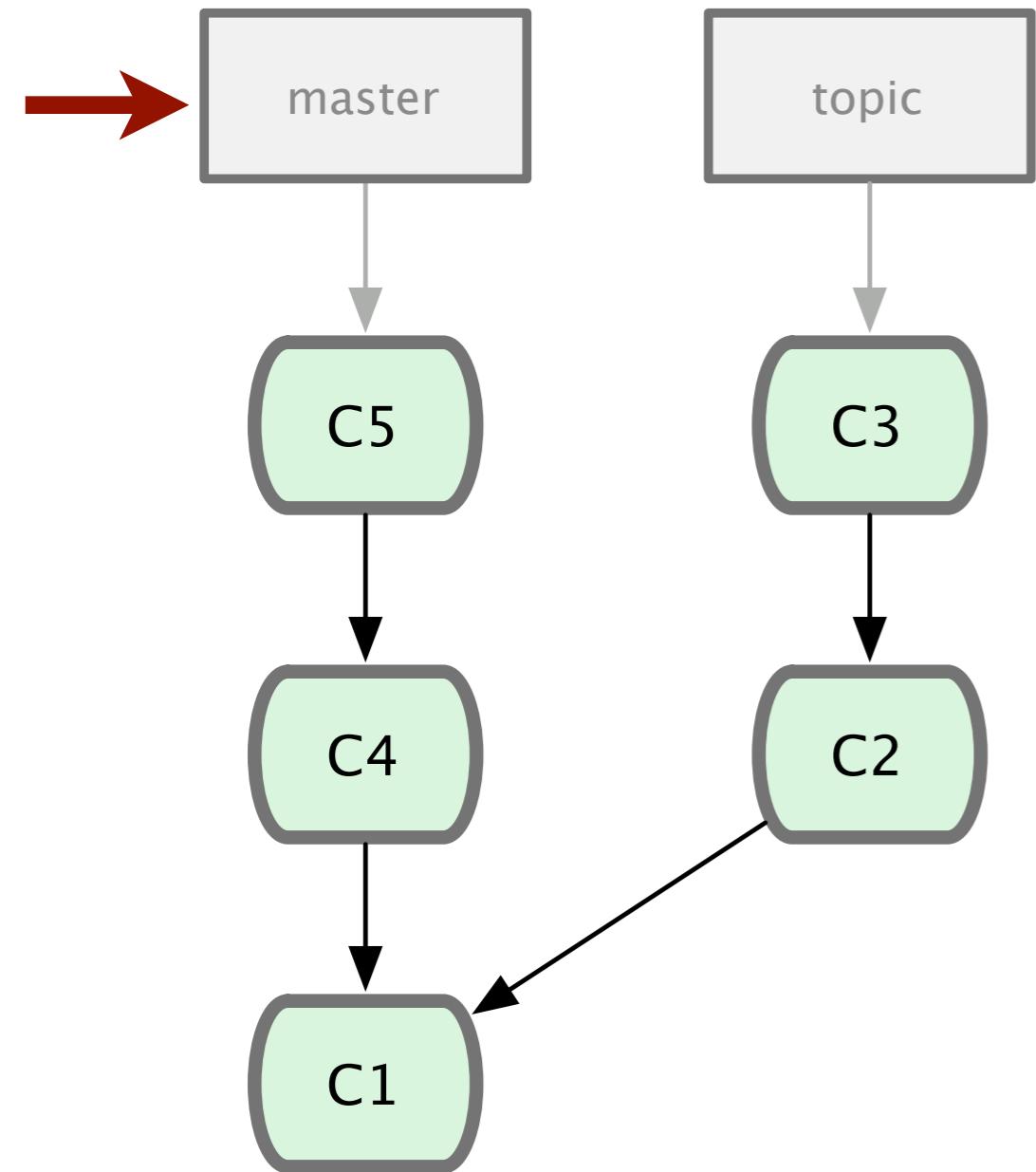








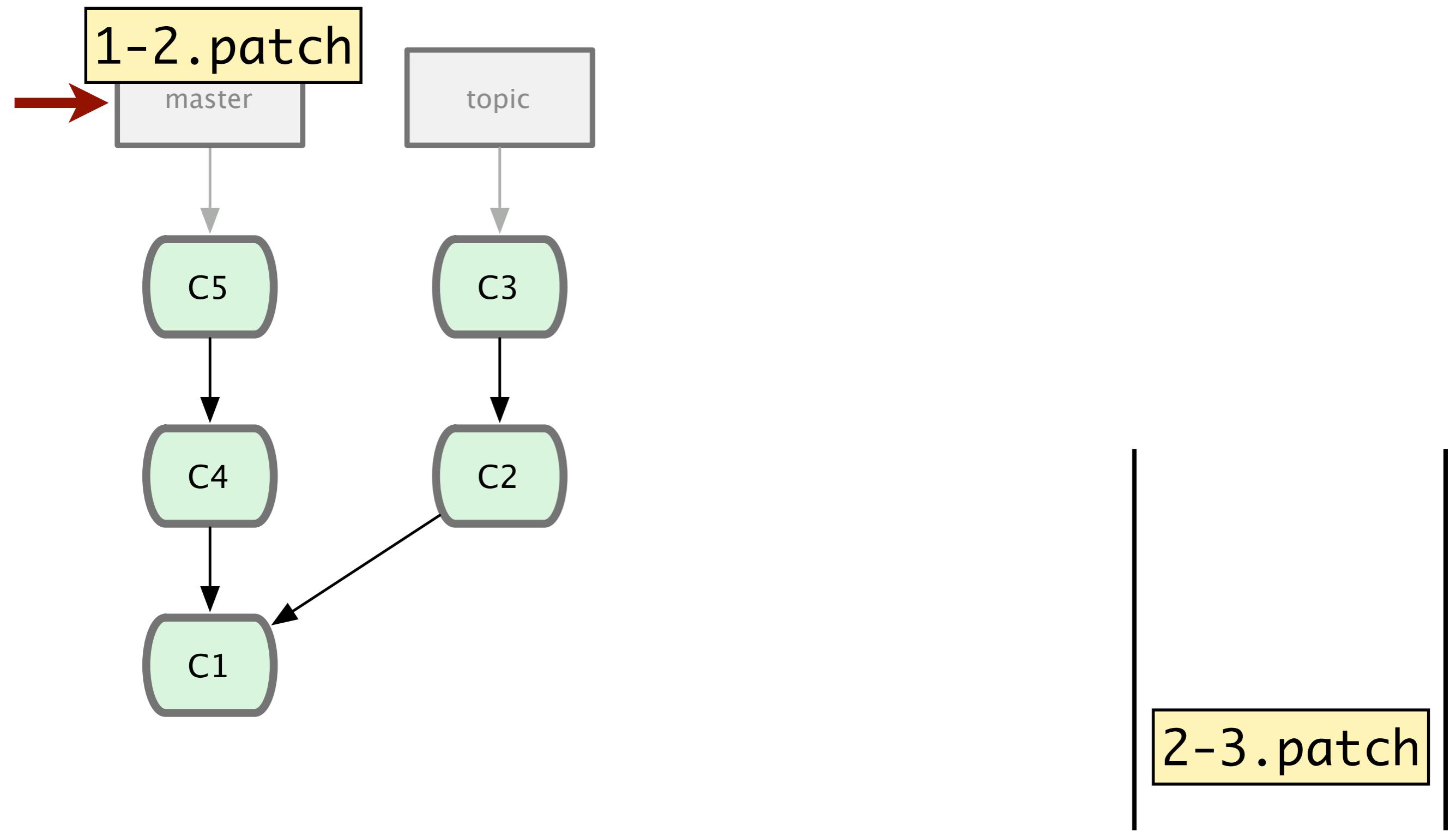
git rebase master



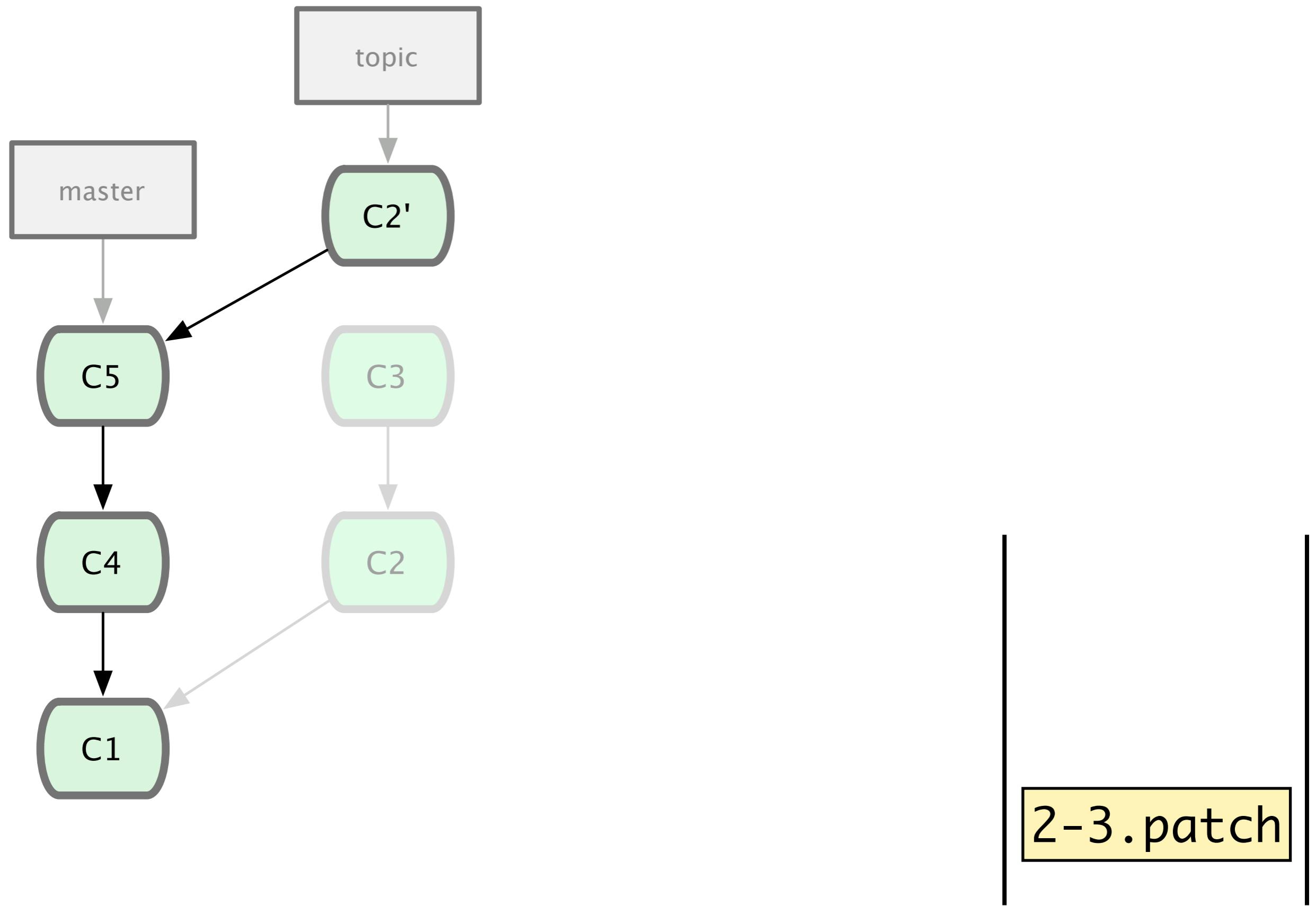
1-2.patch

2-3.patch

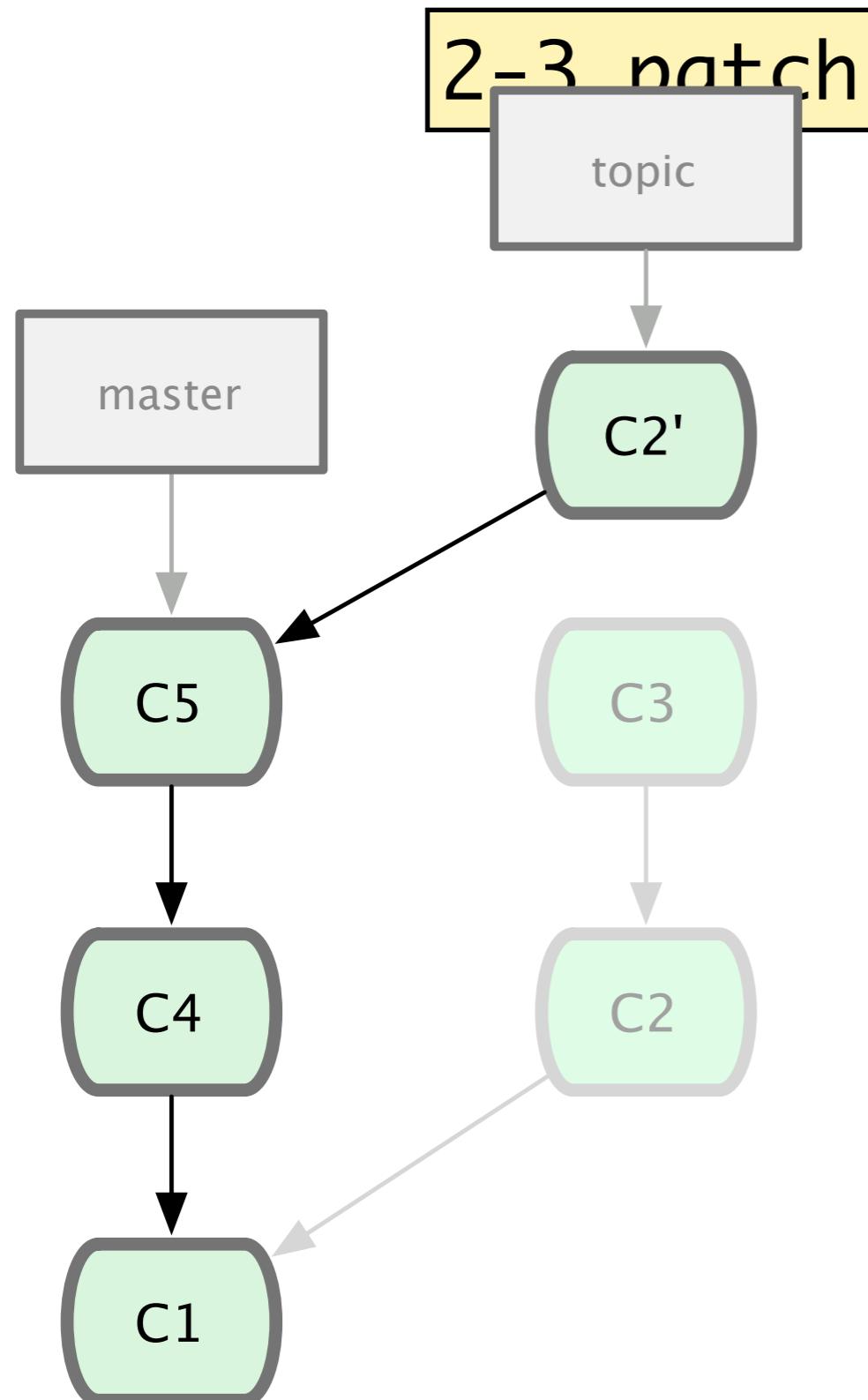
git rebase master

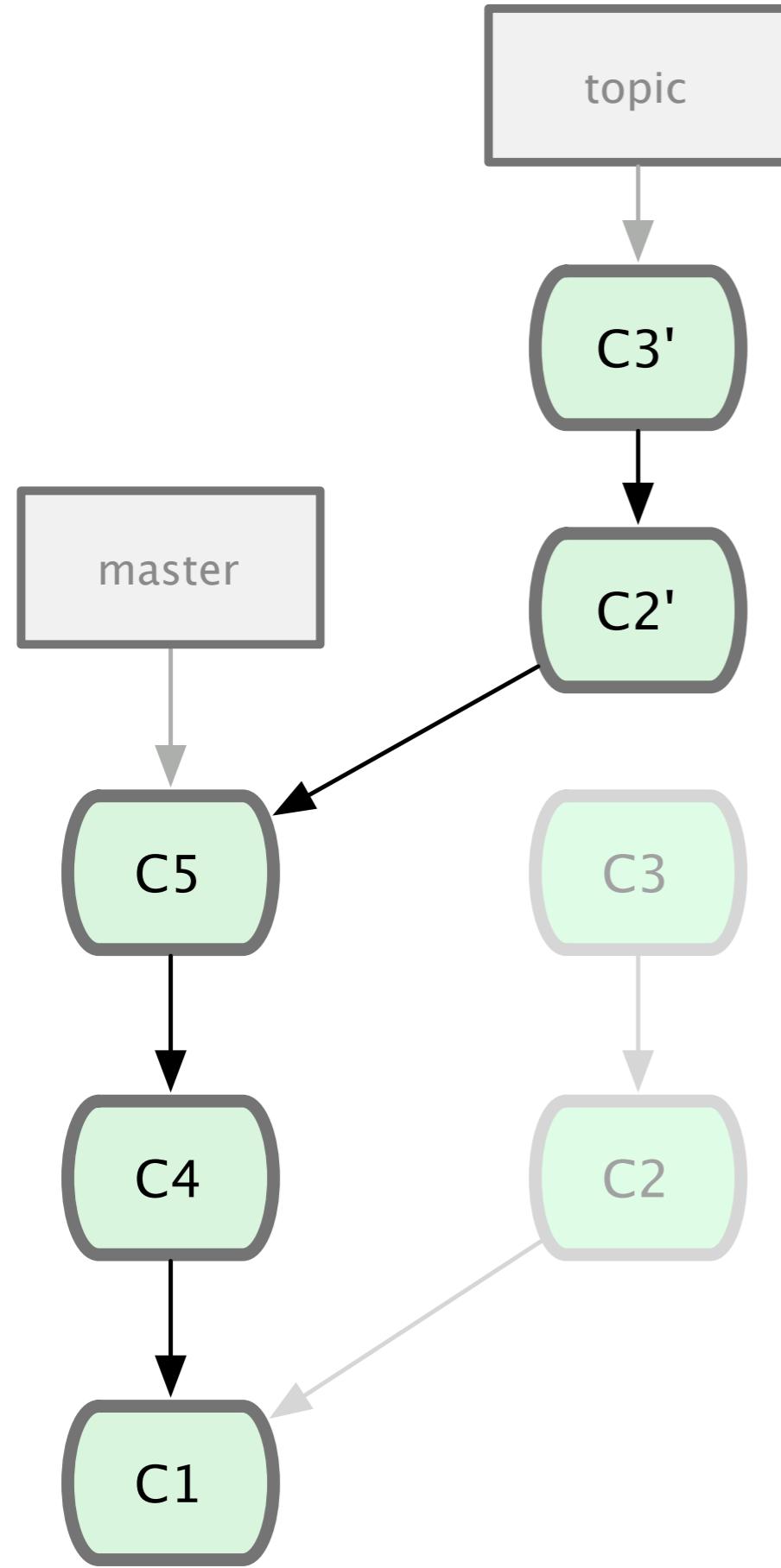


git rebase master

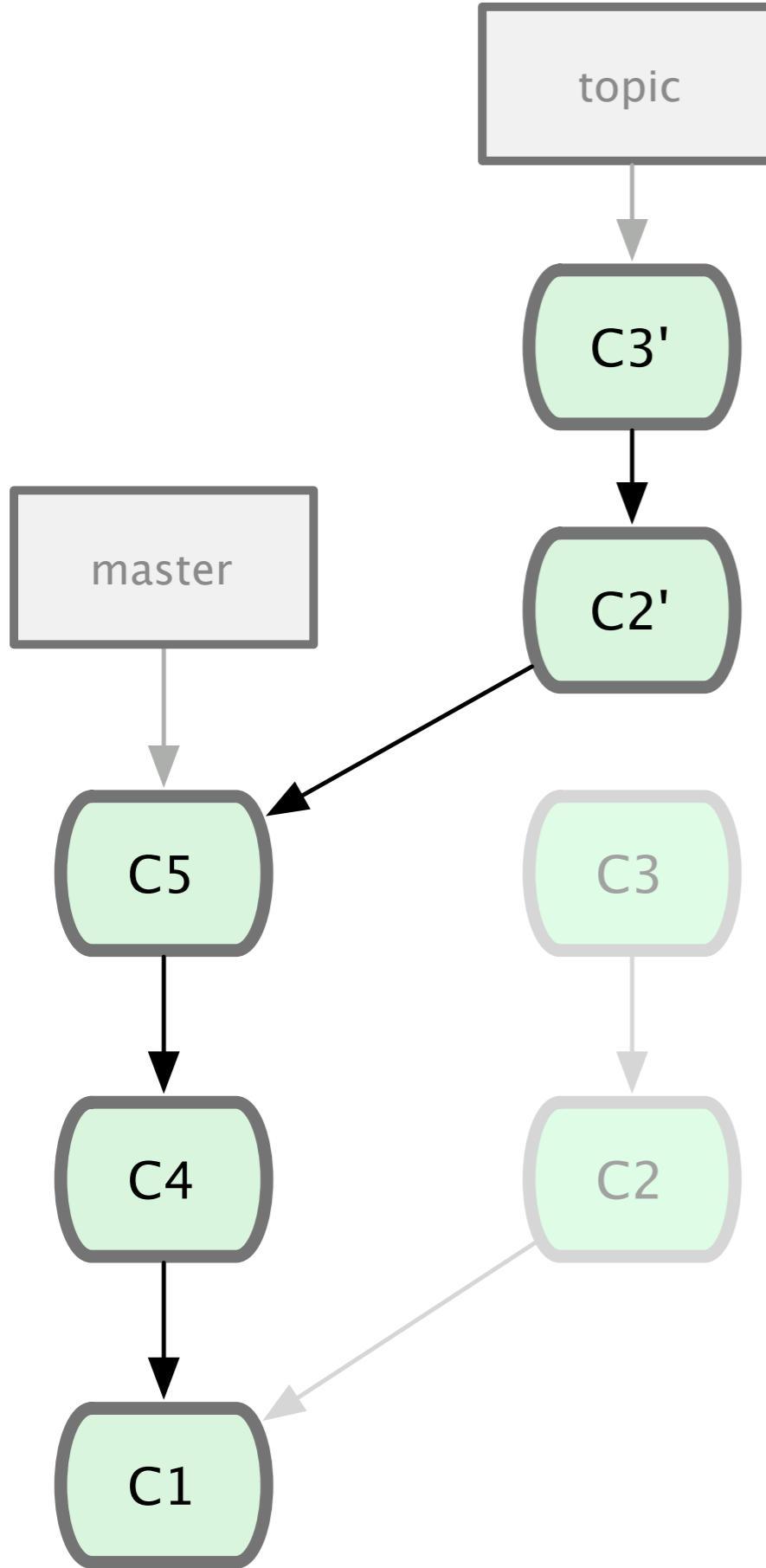


git rebase master

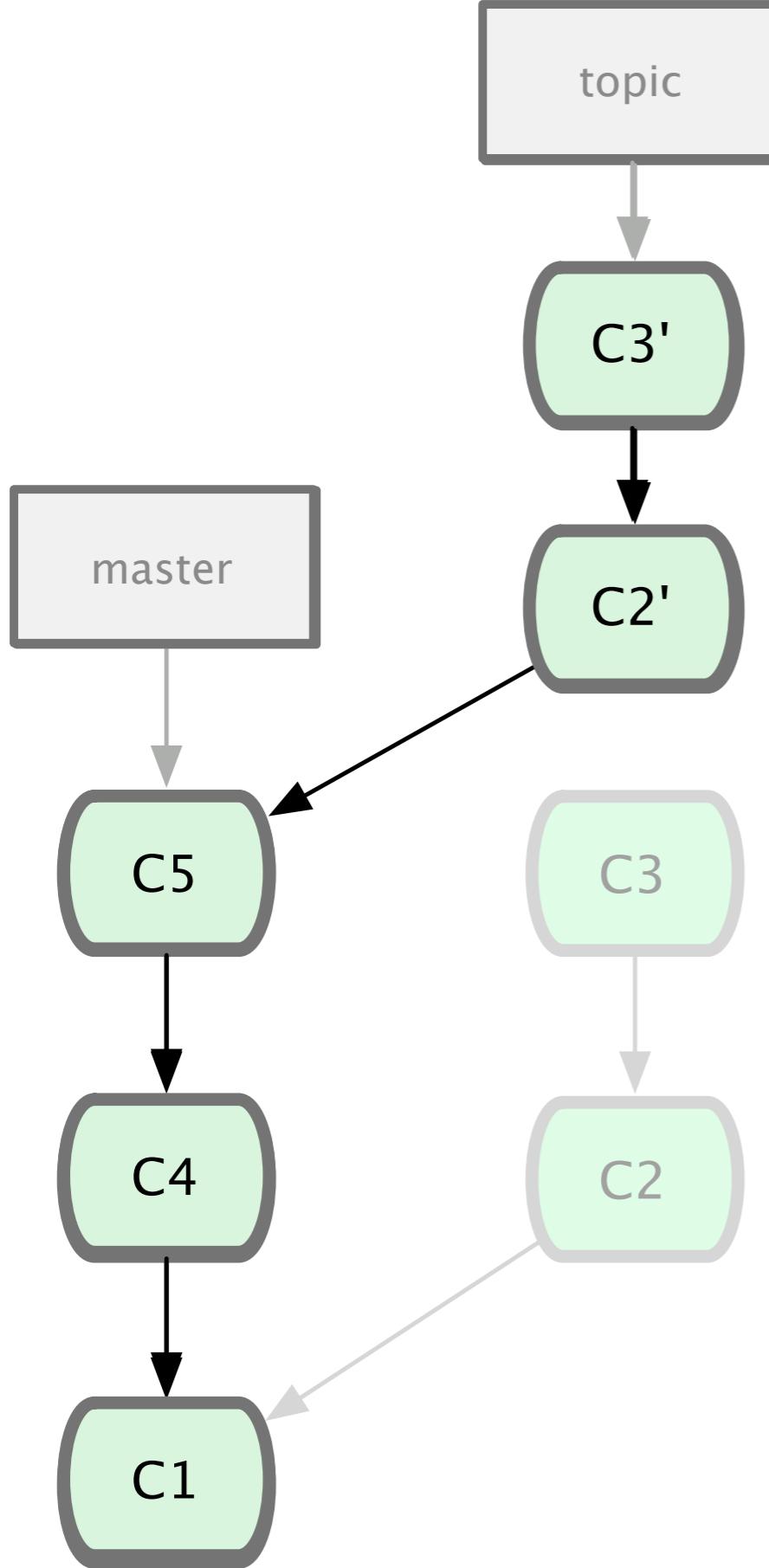




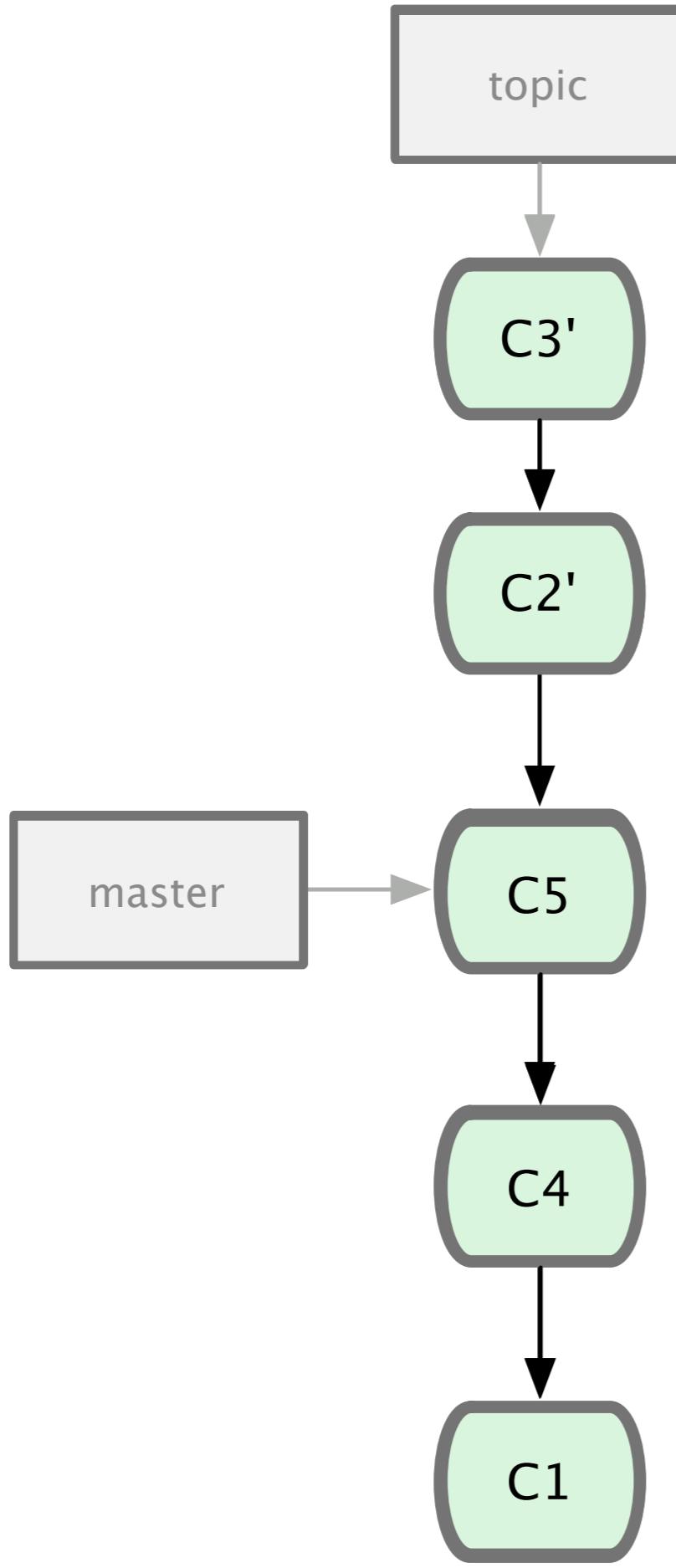
git rebase master



git rebase master



git rebase master

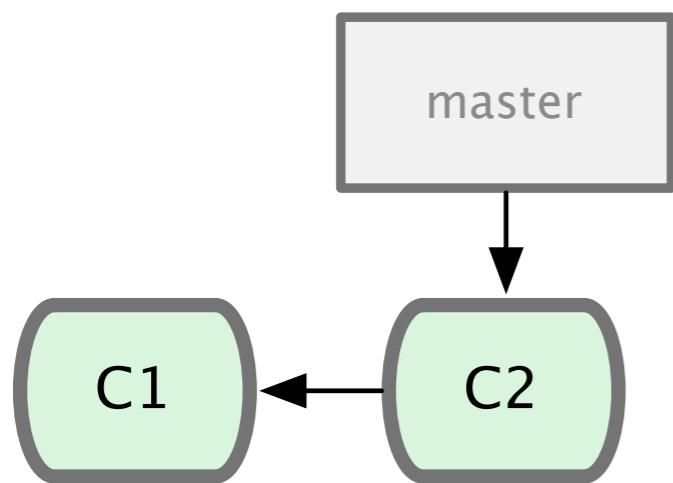


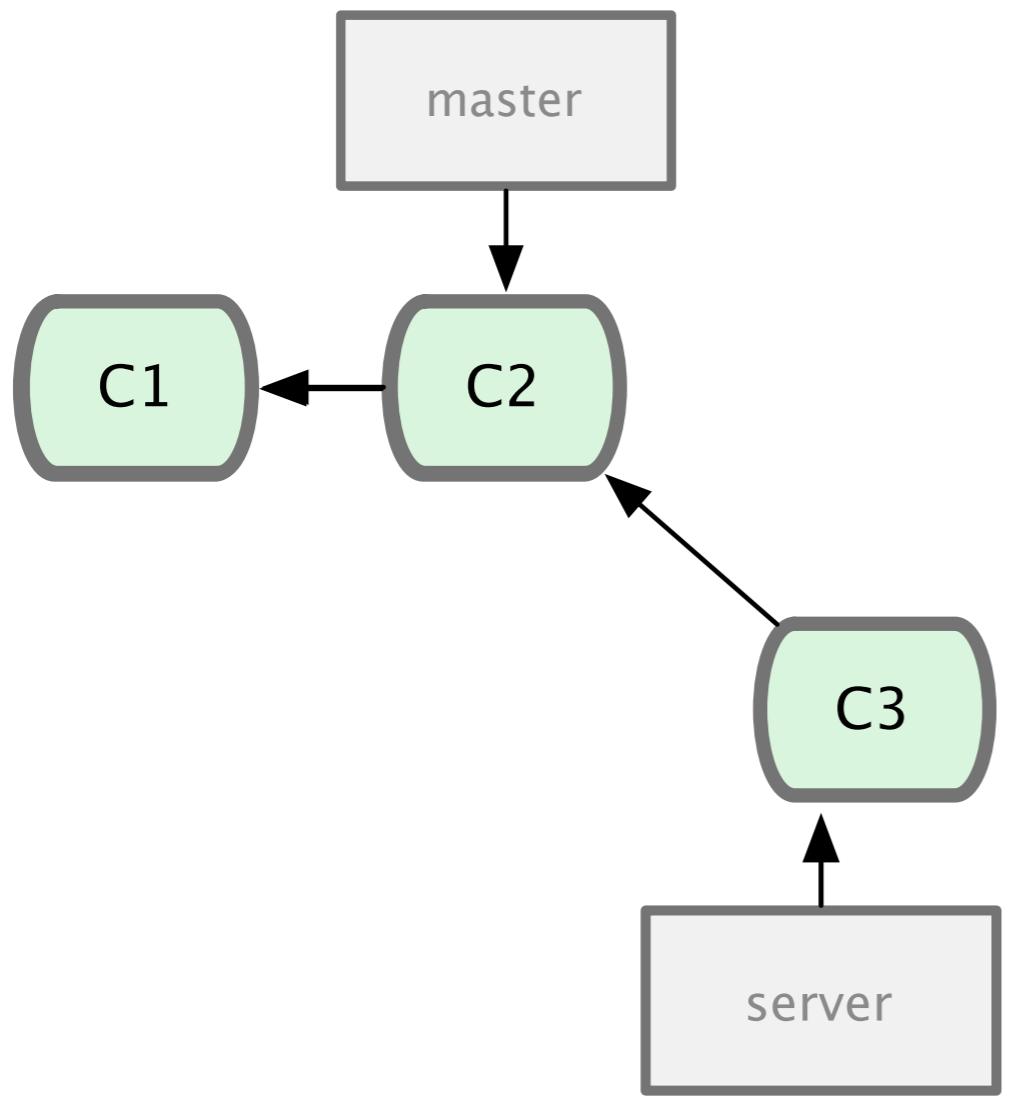
git rebase master

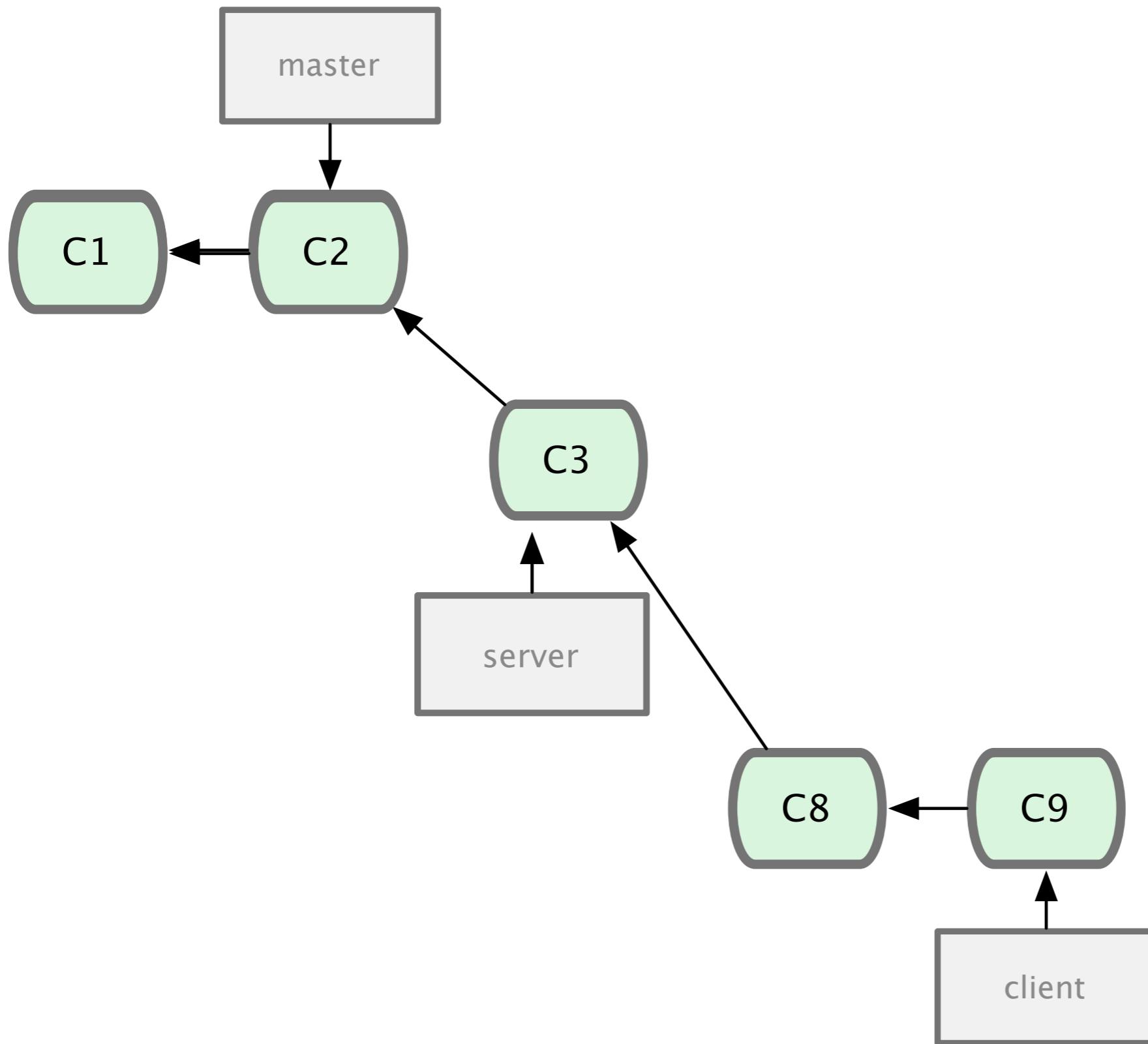
Fun with Rebasing

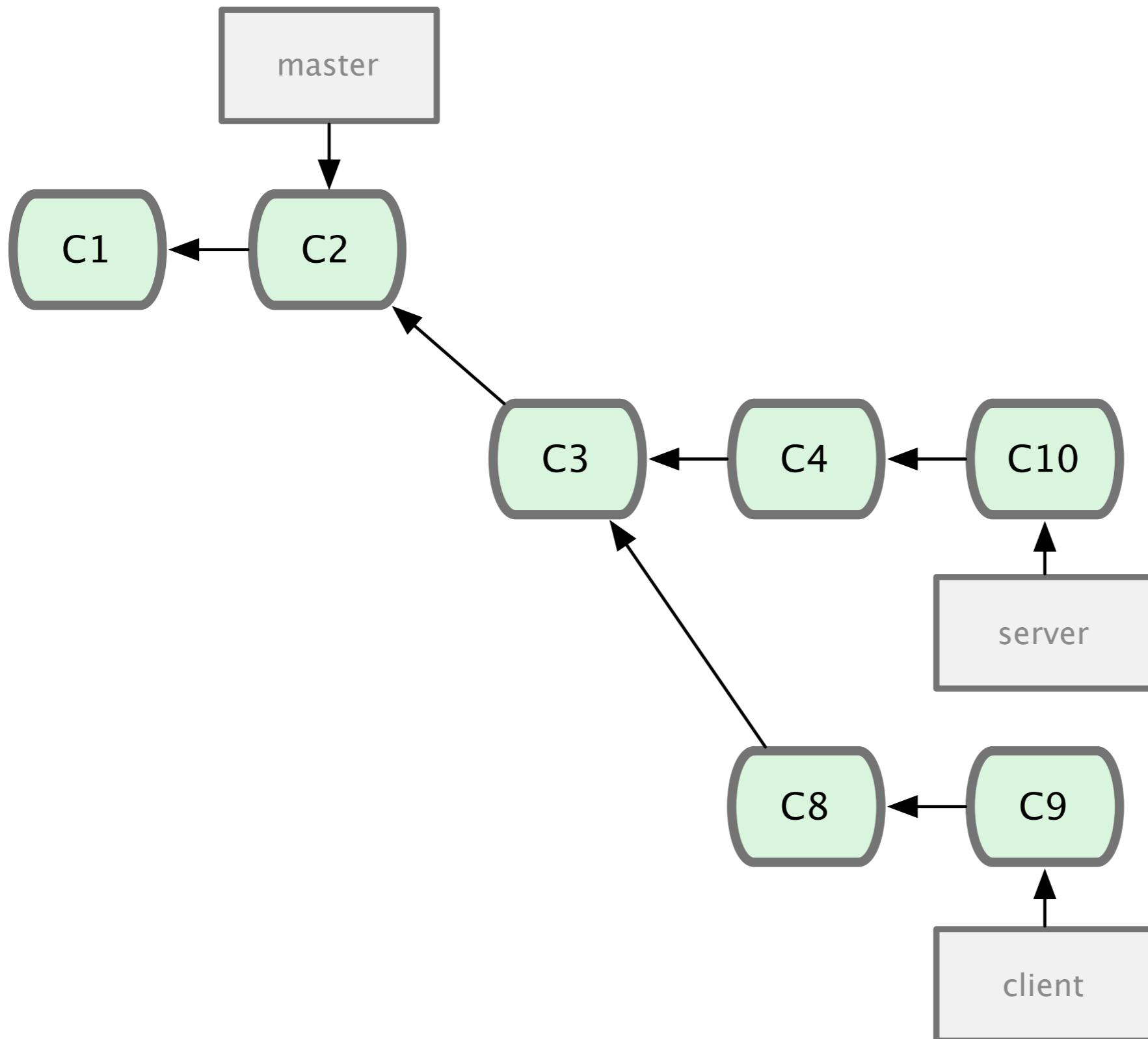
rebase --onto

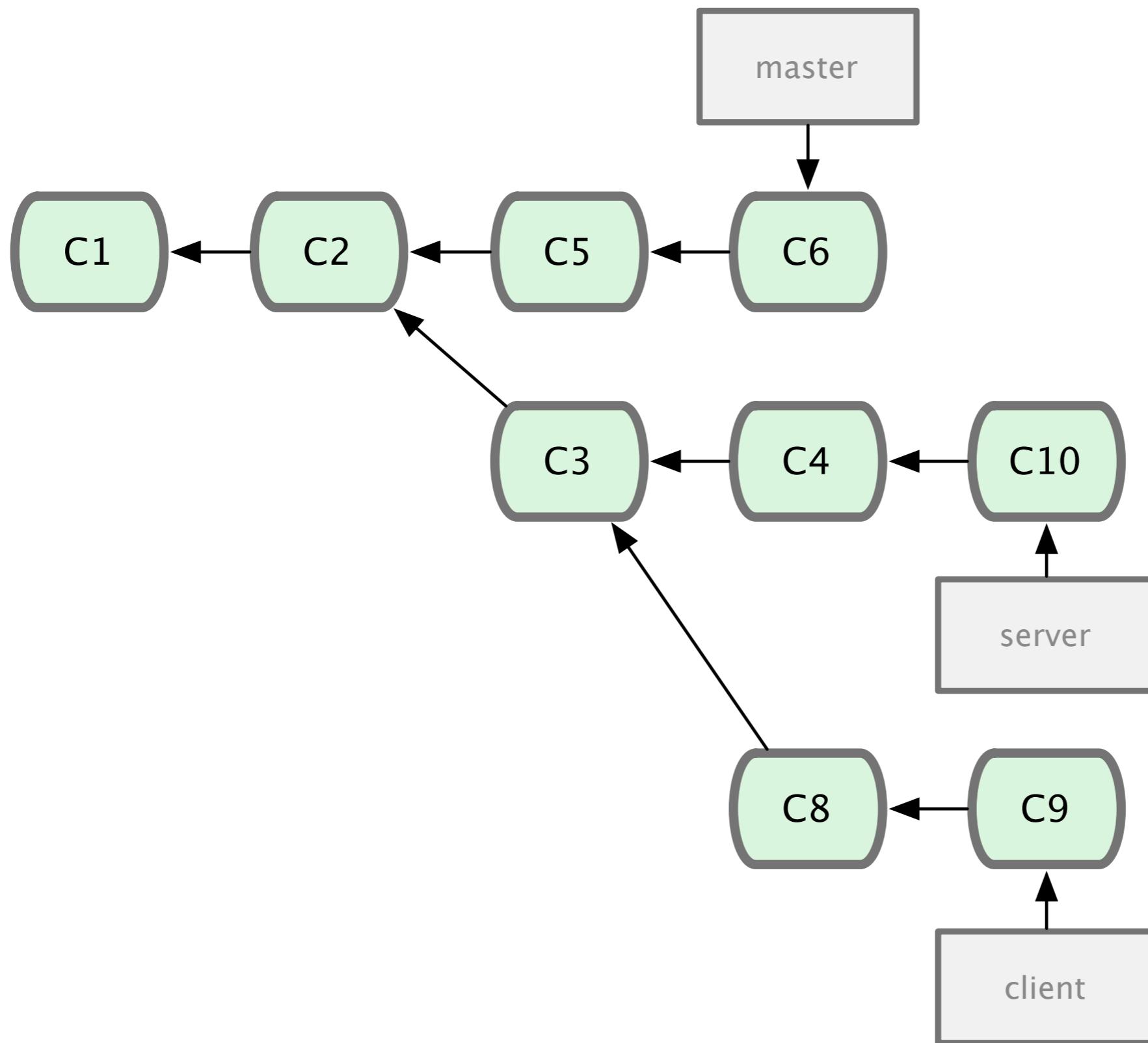
Transplanting Topic Branches



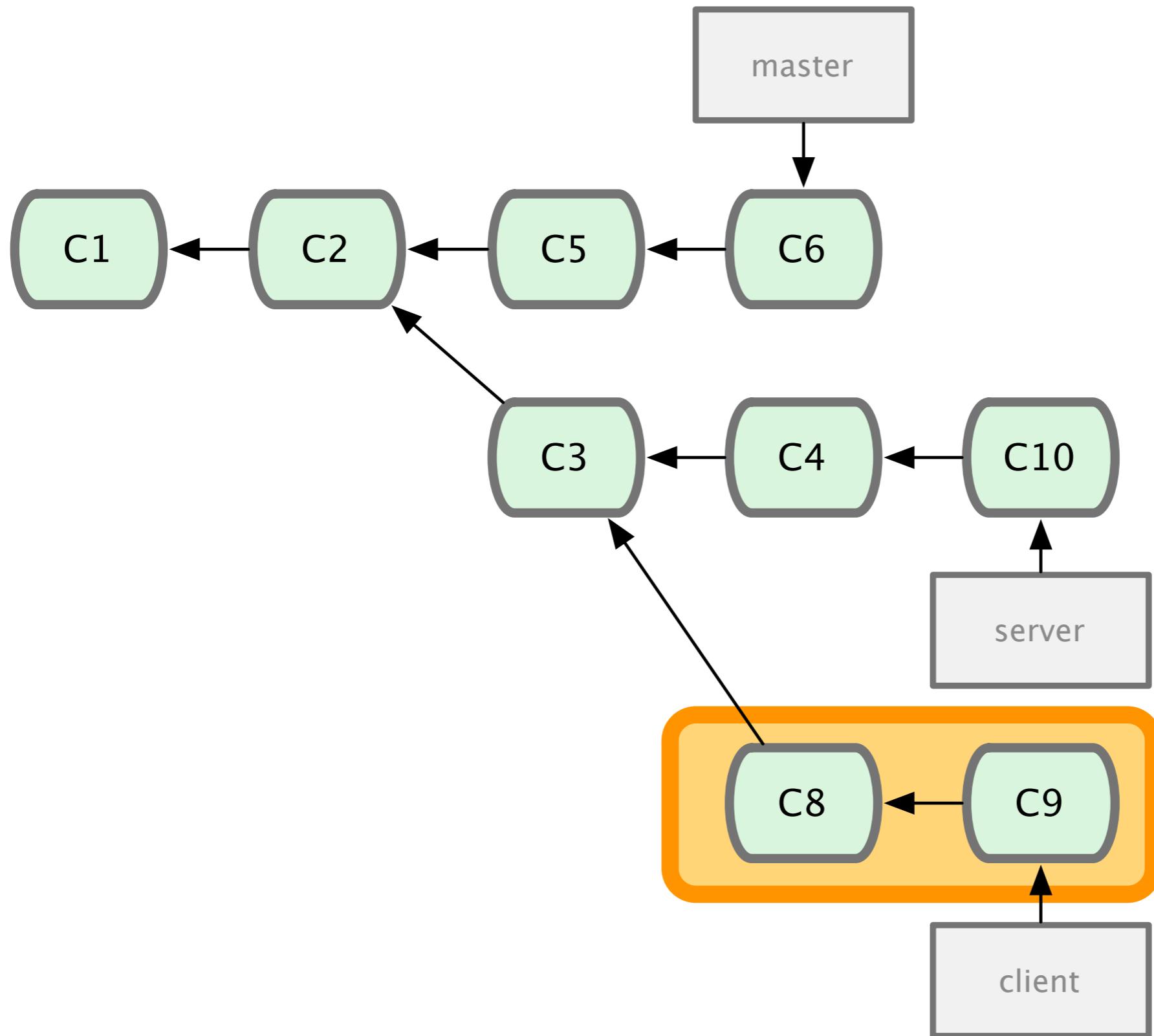


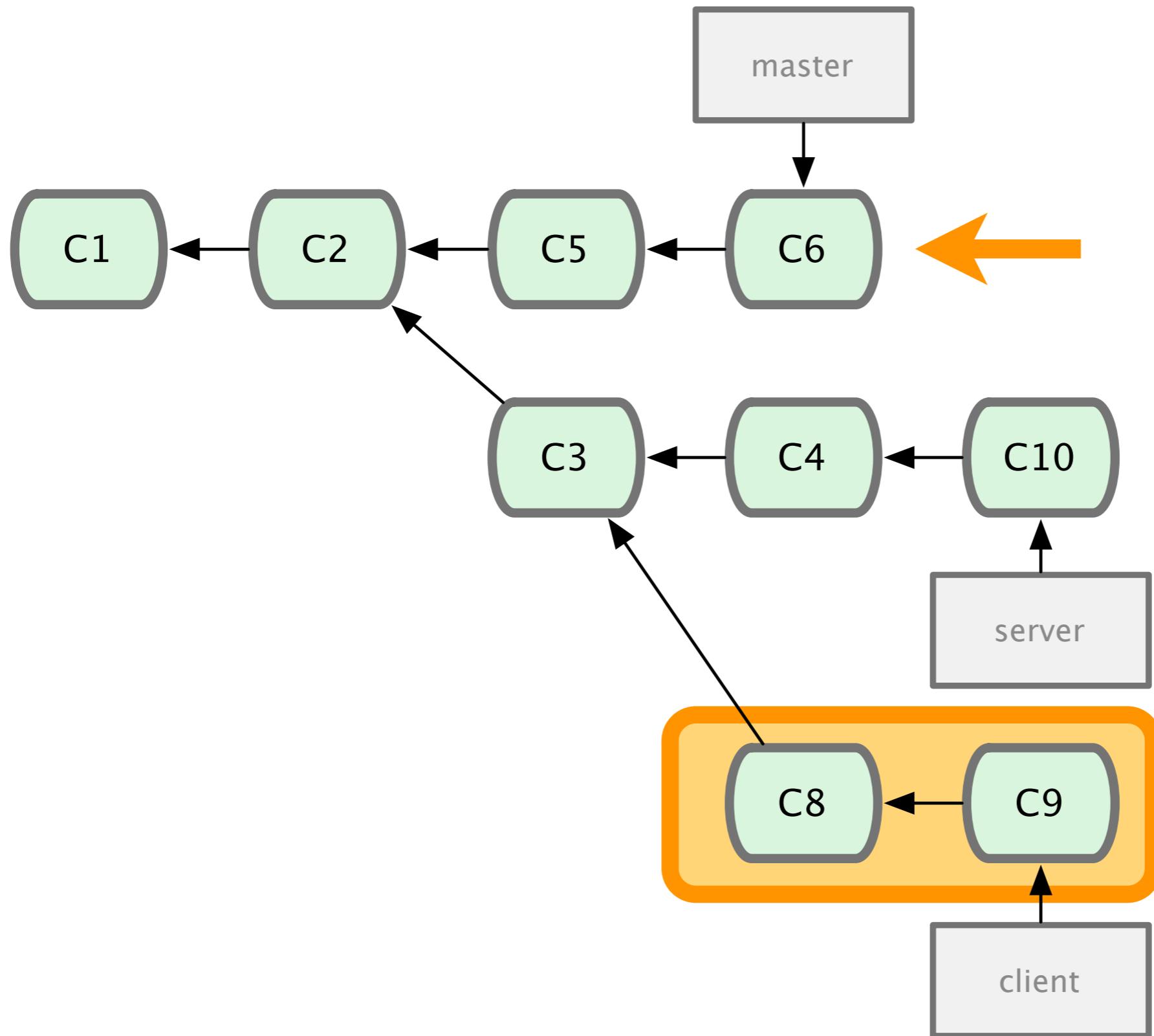


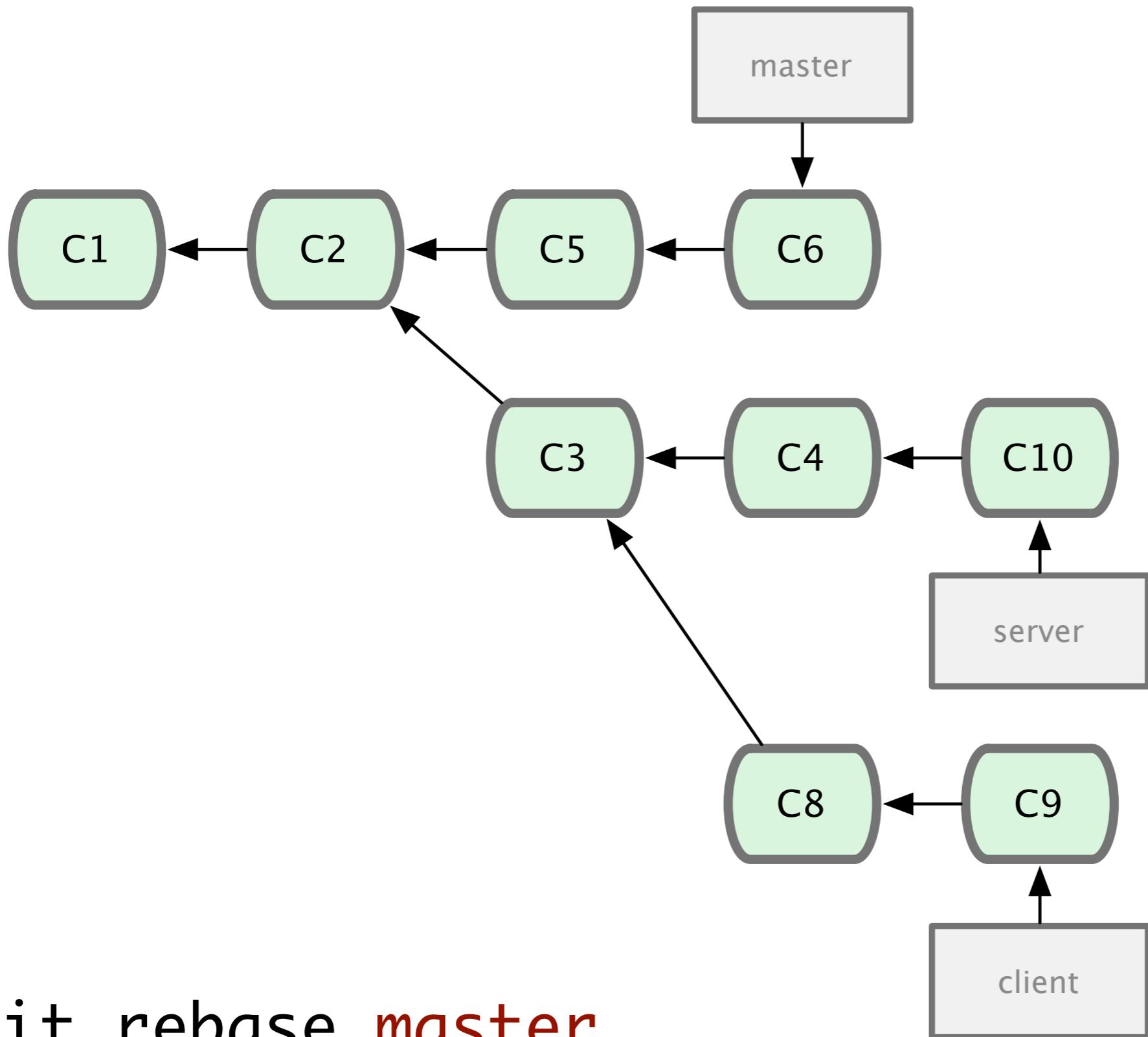




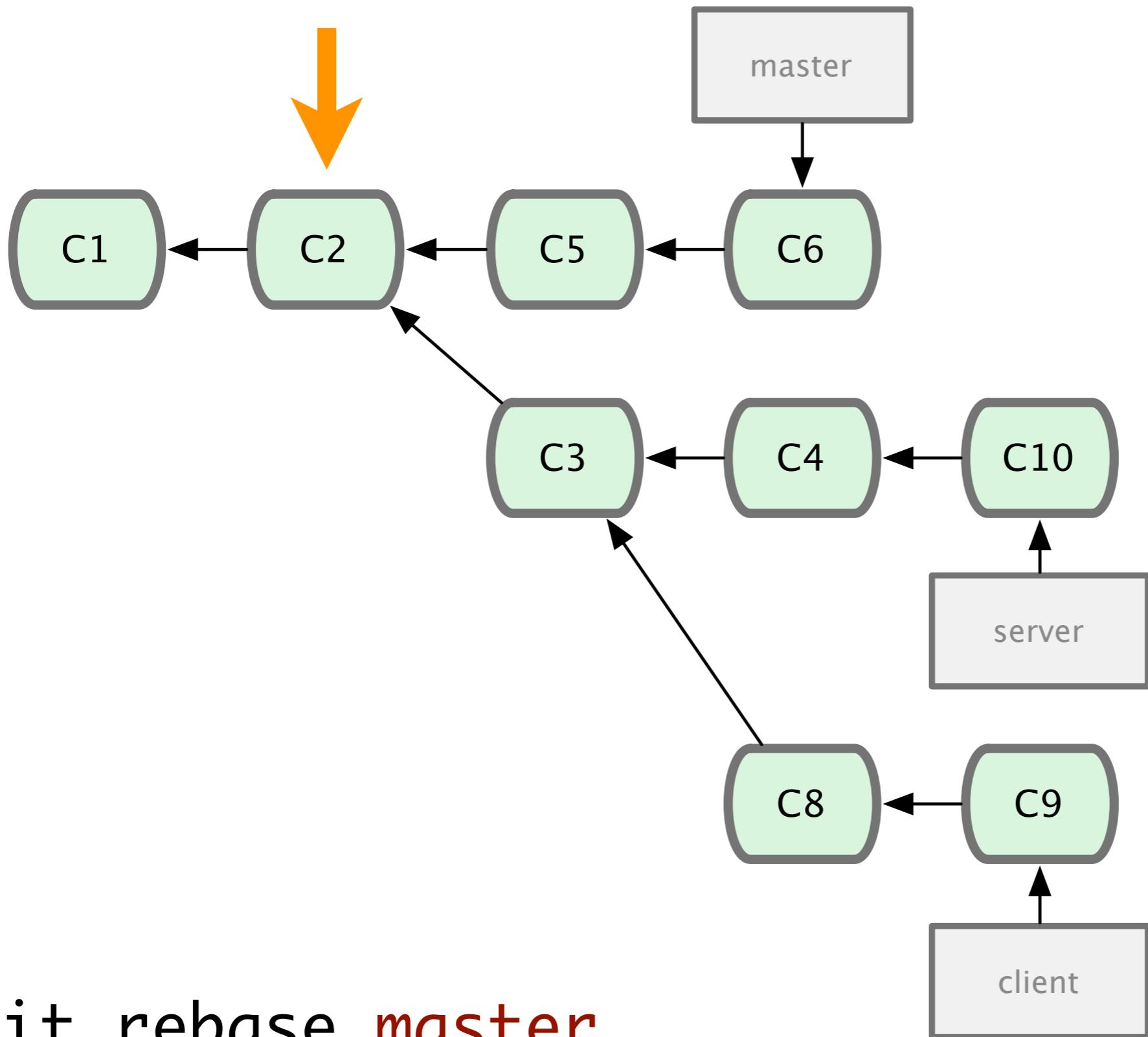
move your ‘client’
branch work to your
‘master’ branch



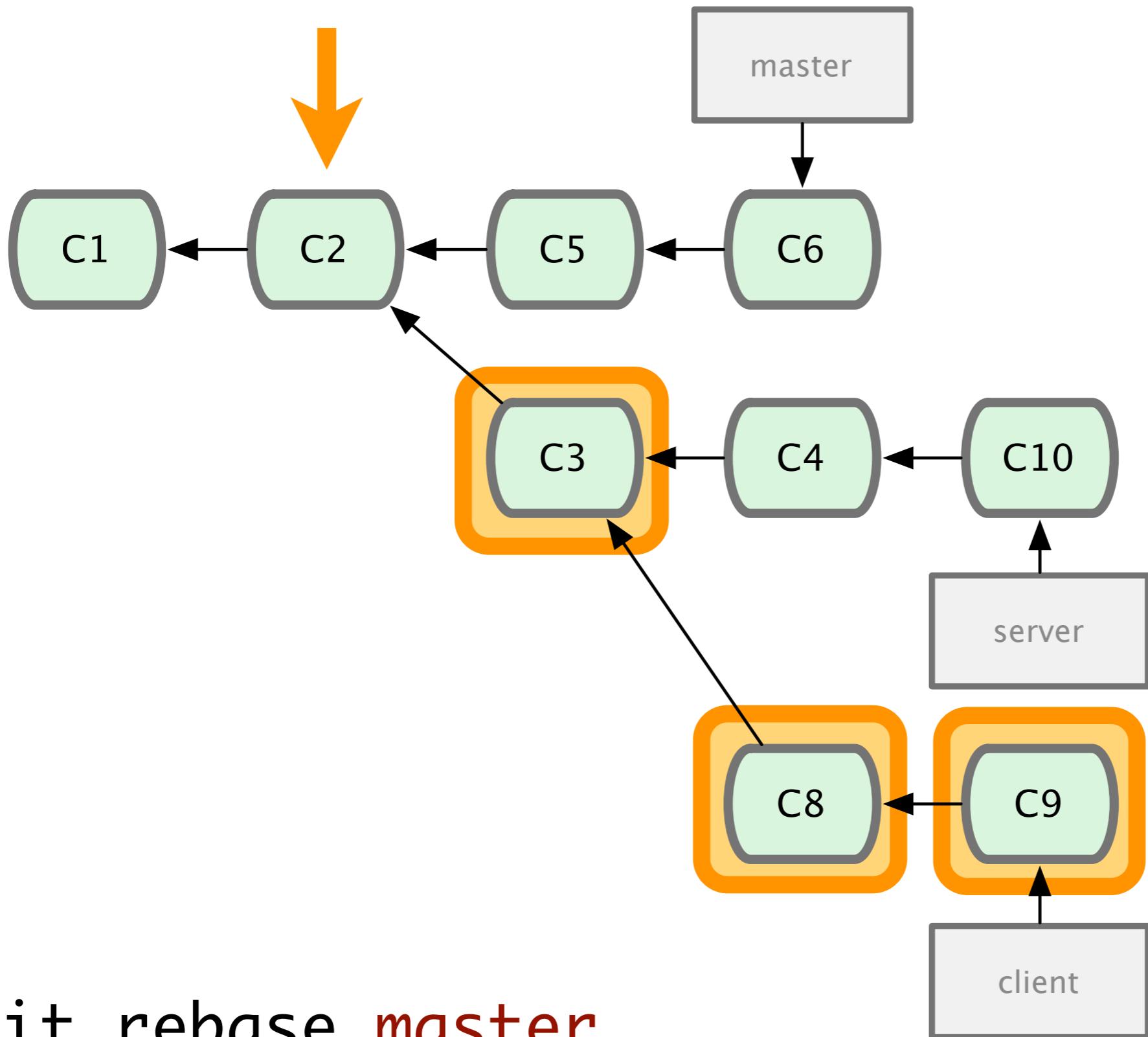




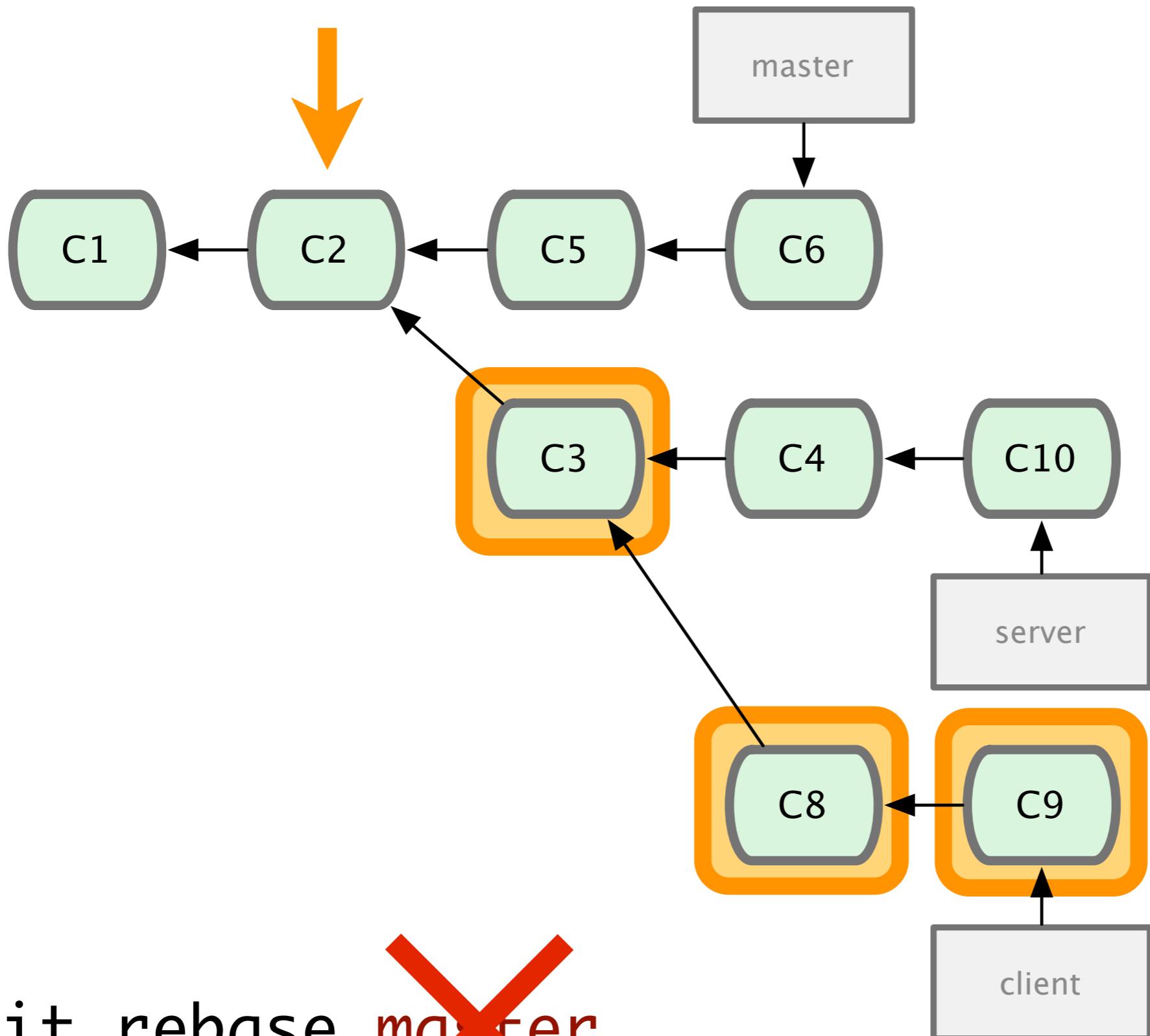
git rebase master



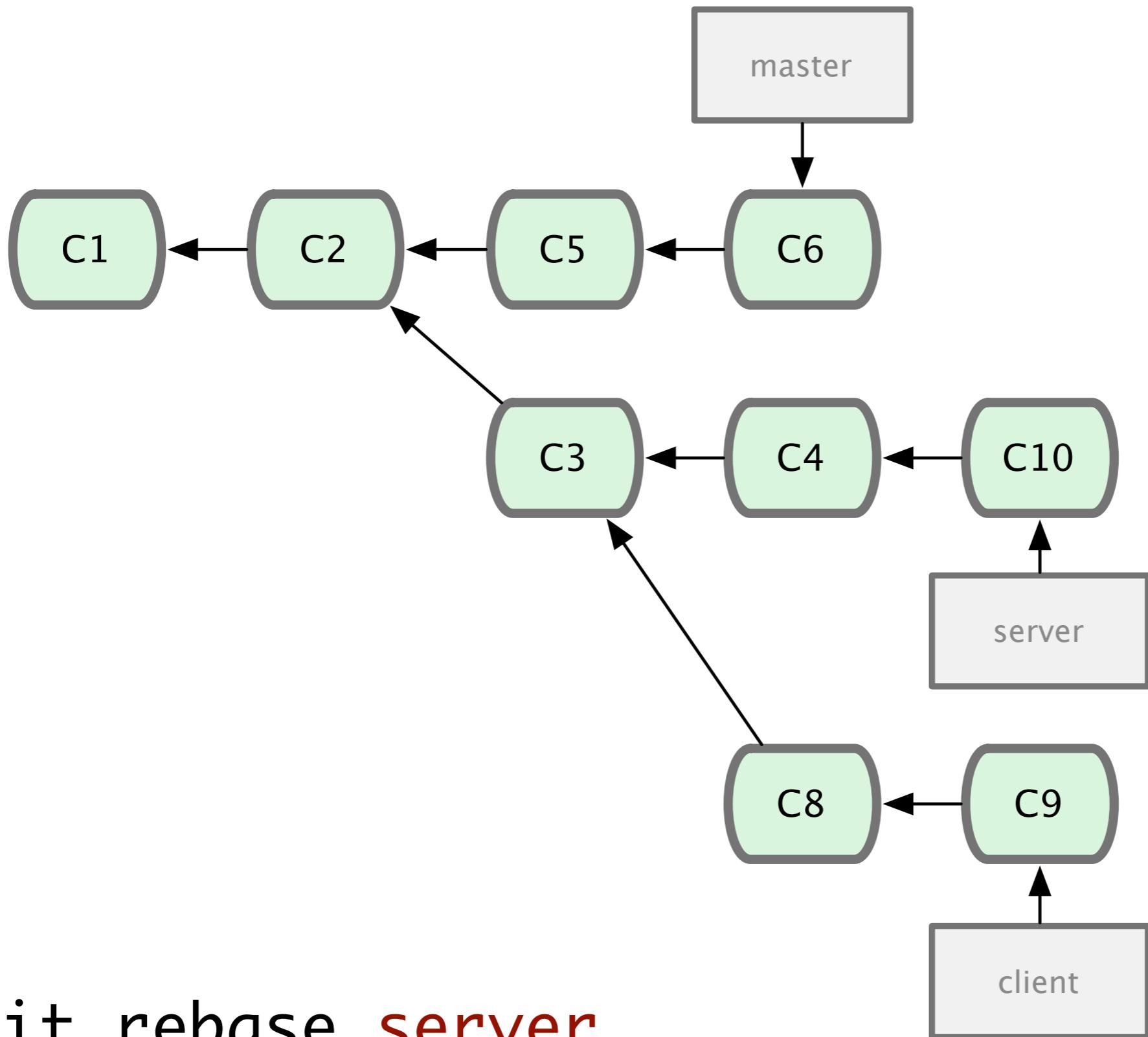
git rebase master



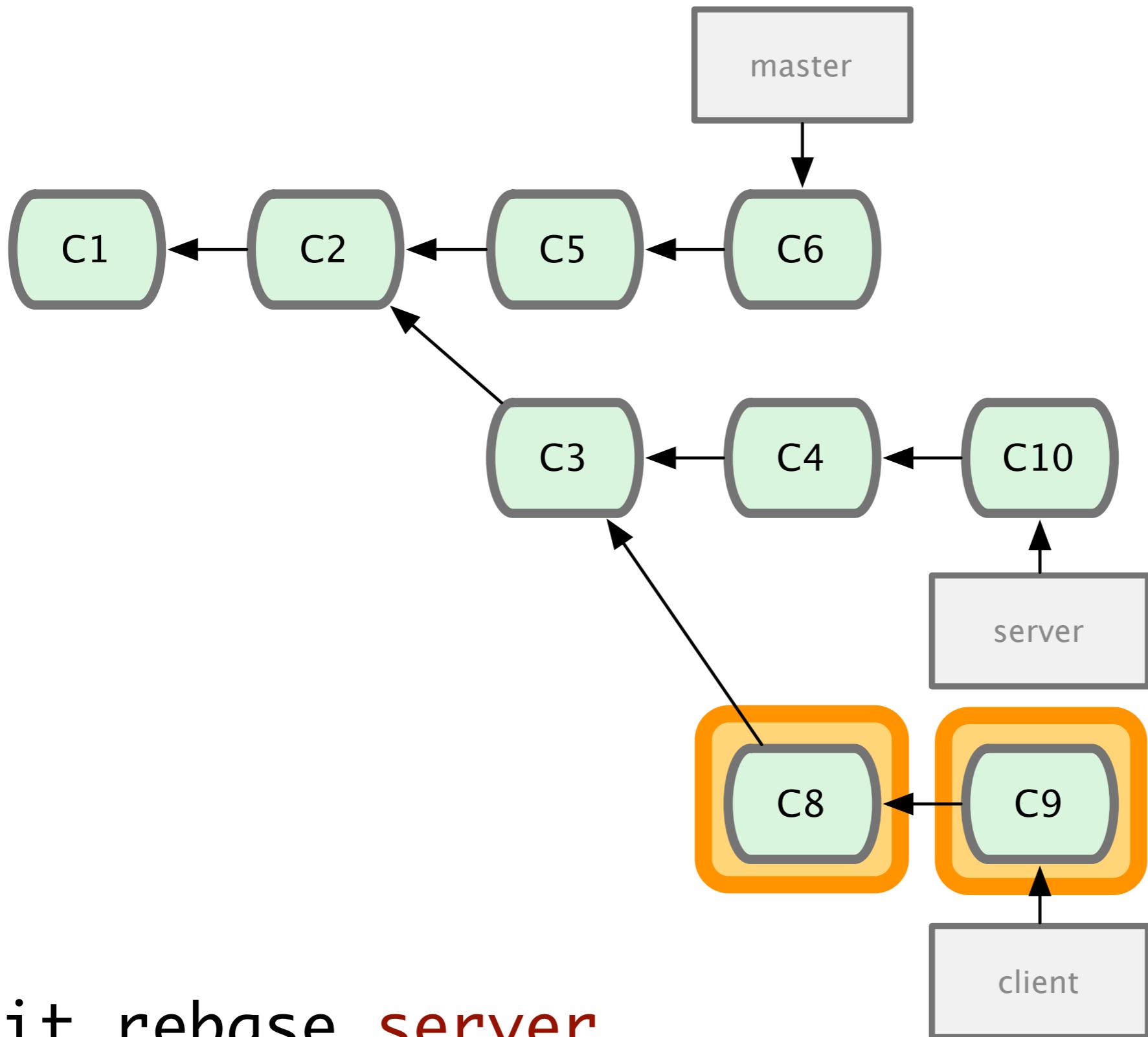
git rebase master



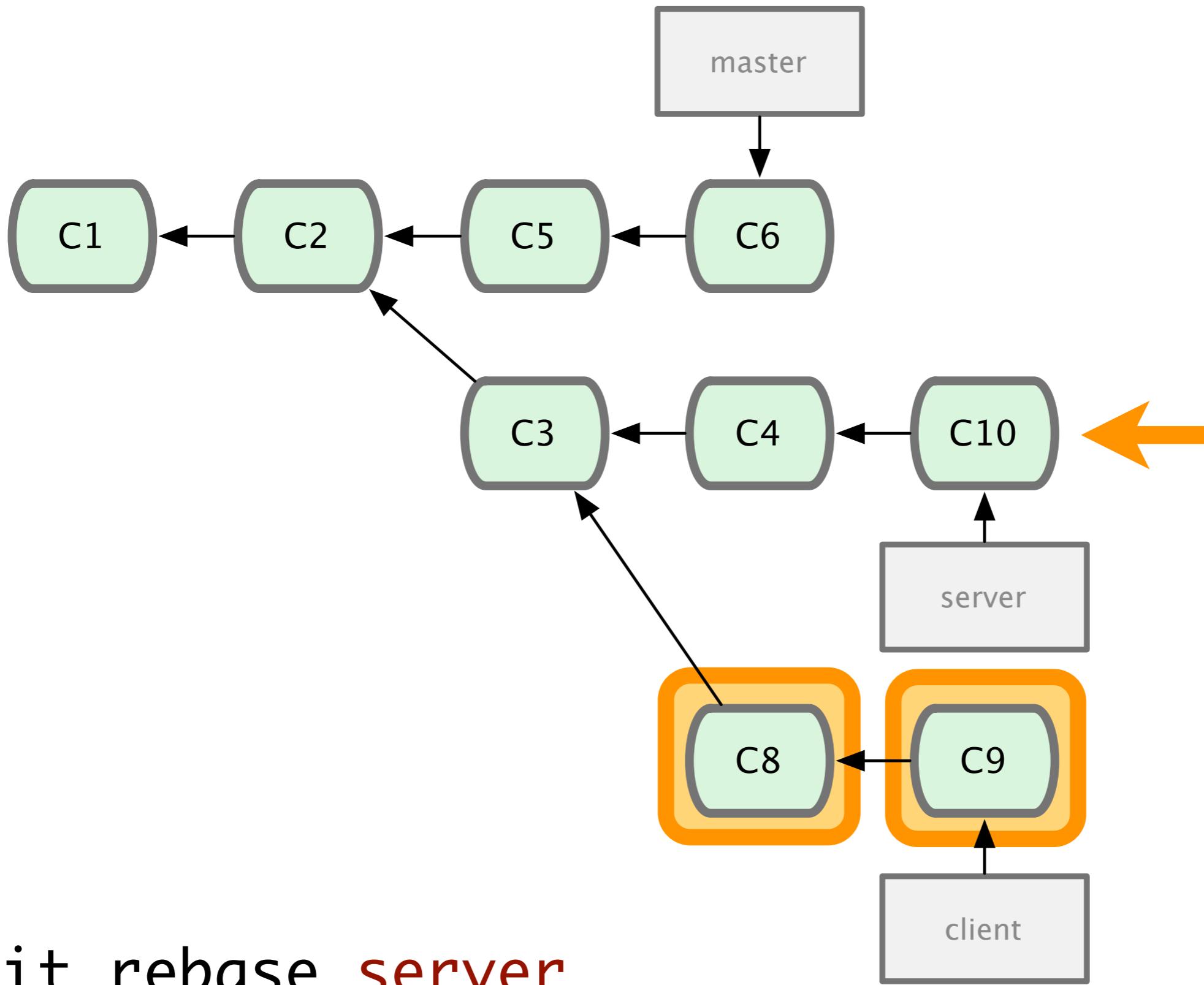
git rebase ~~master~~



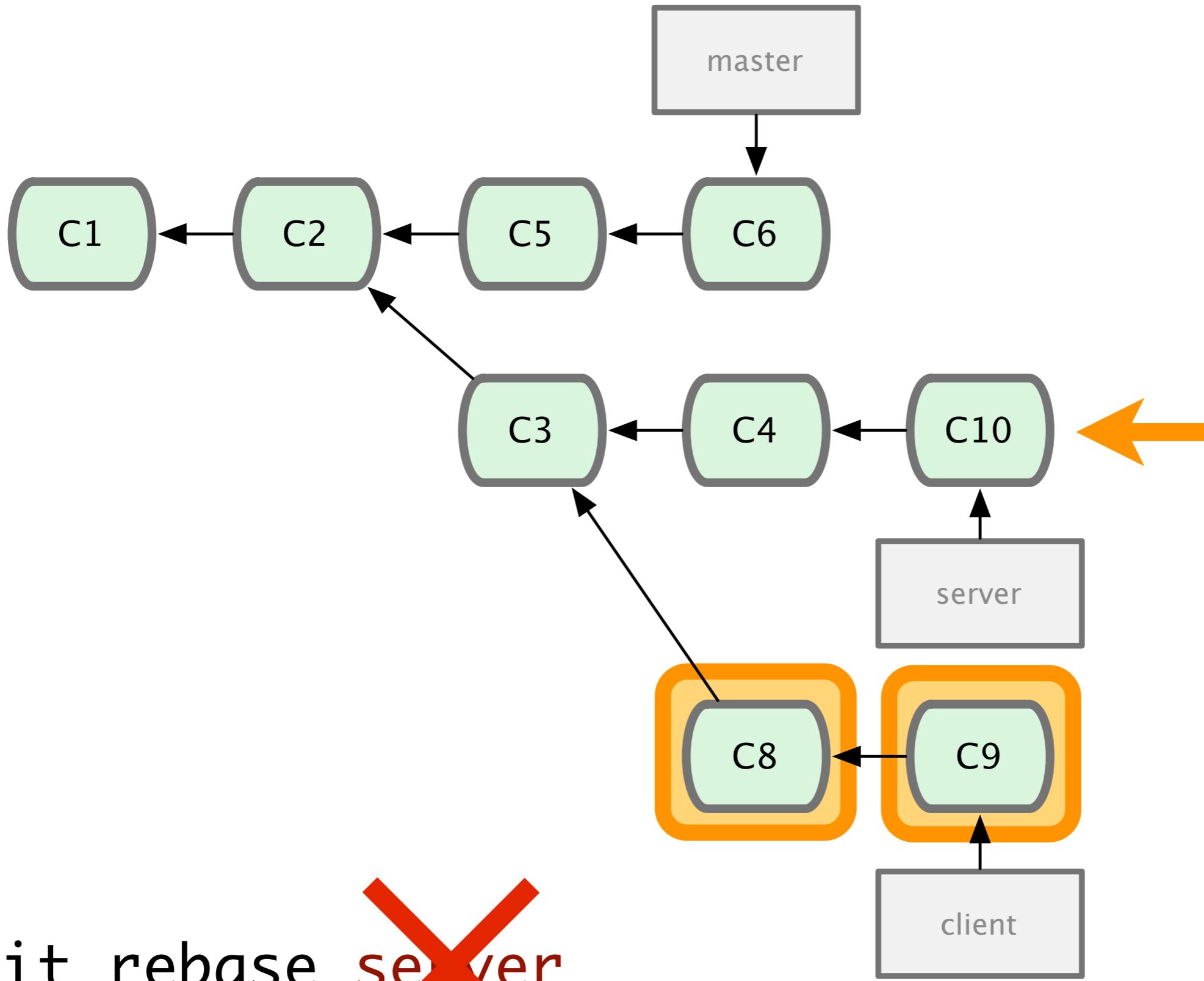
git rebase server



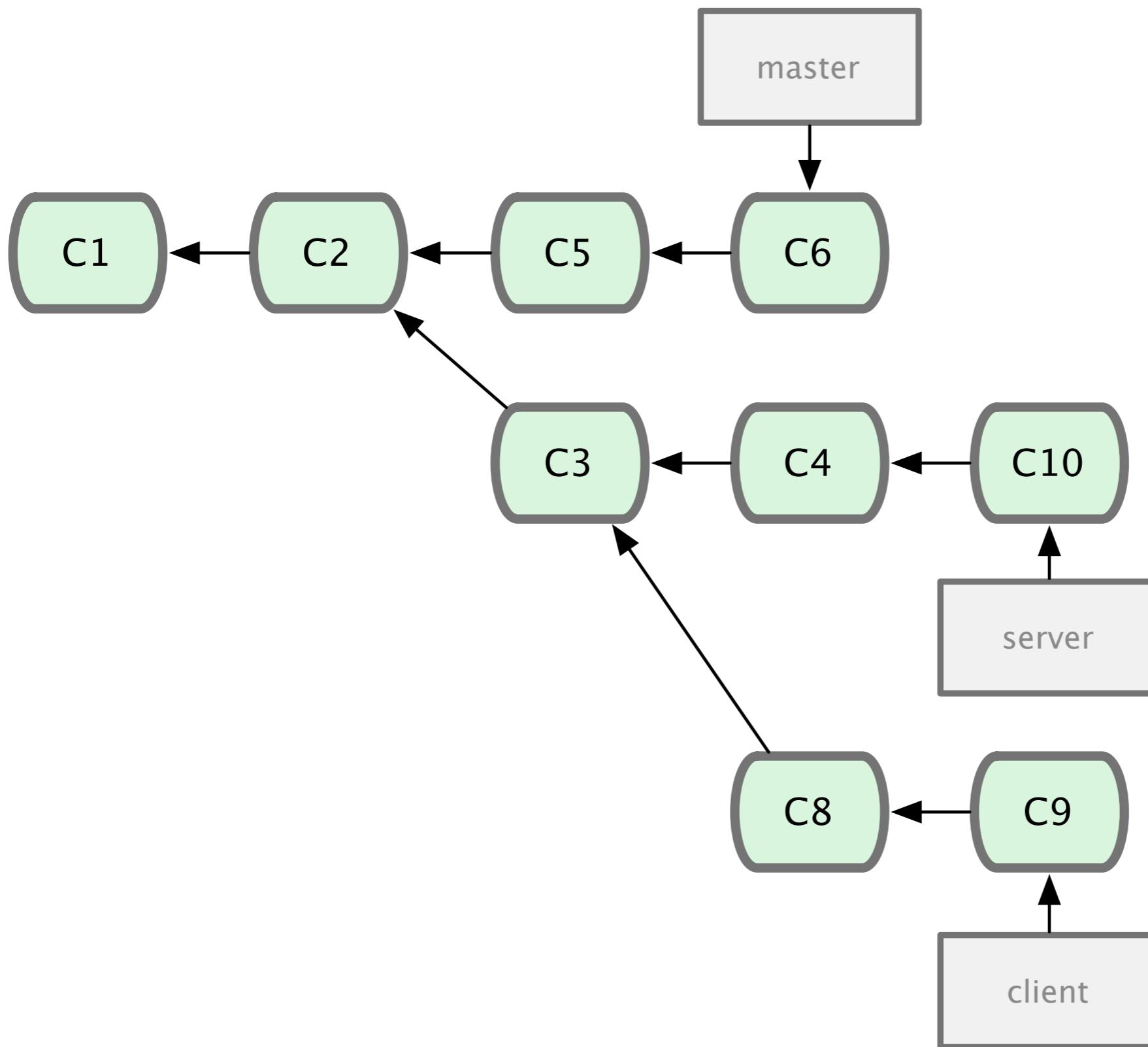
git rebase server



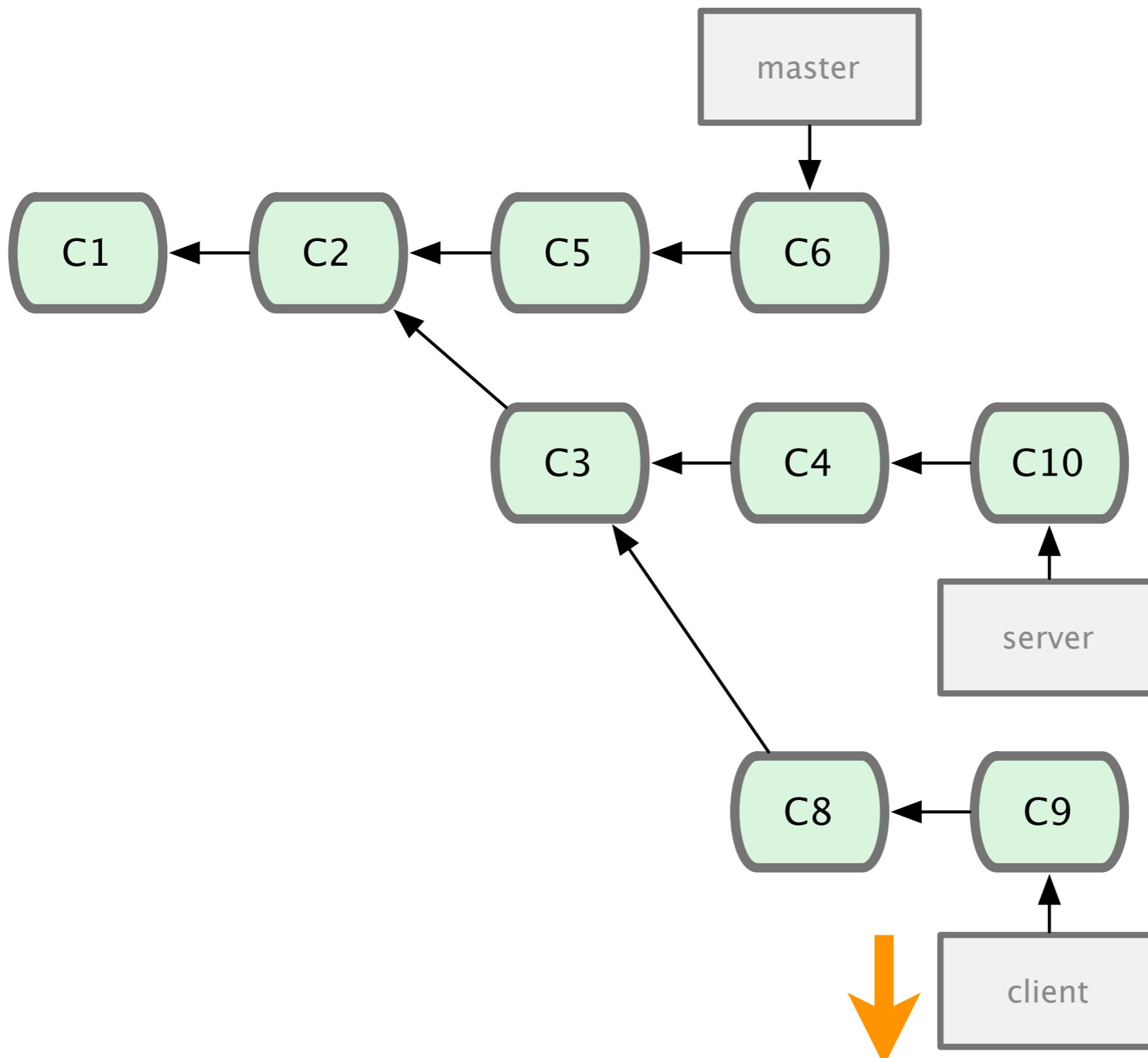
git rebase server



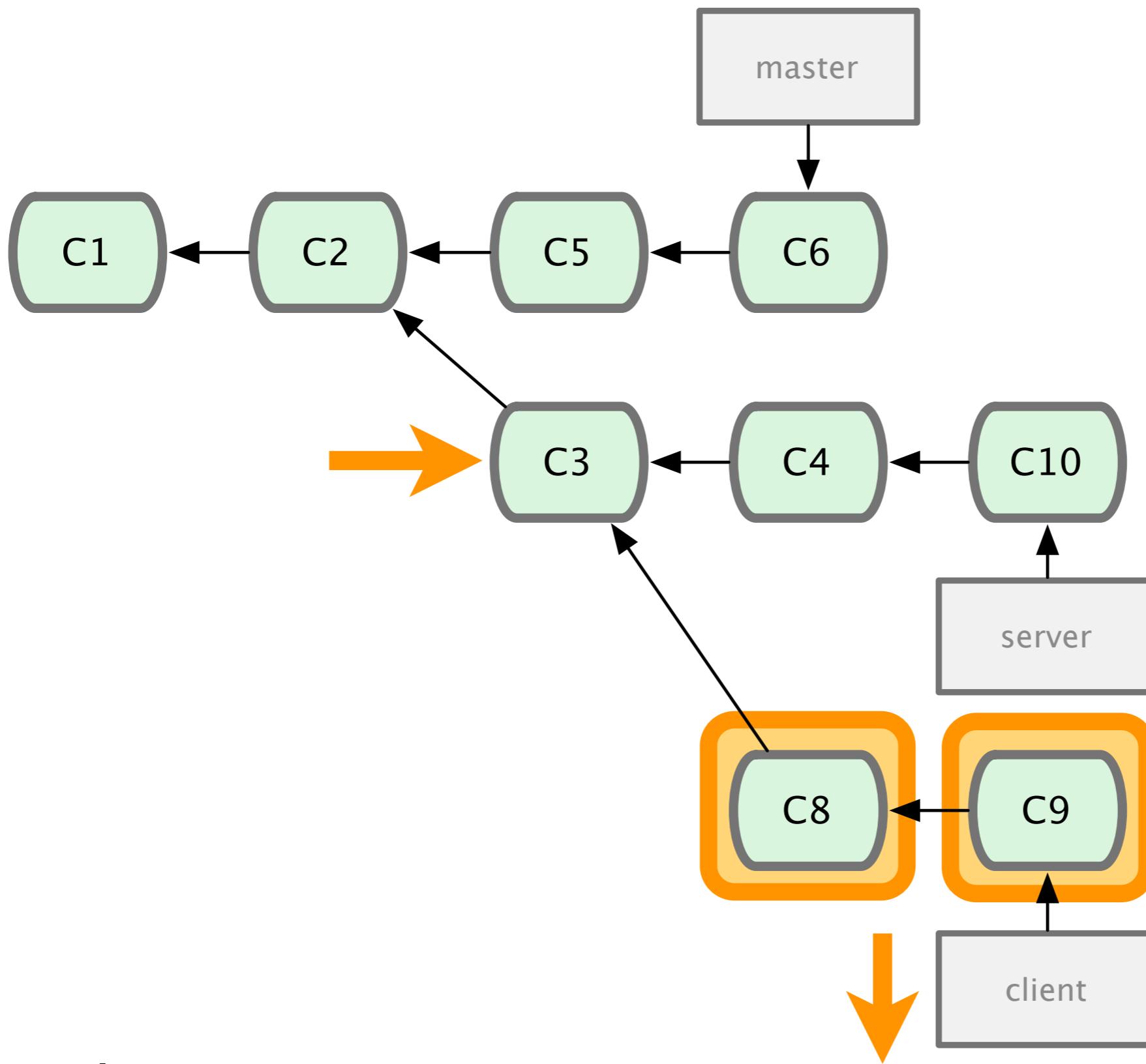
git rebase ~~server~~



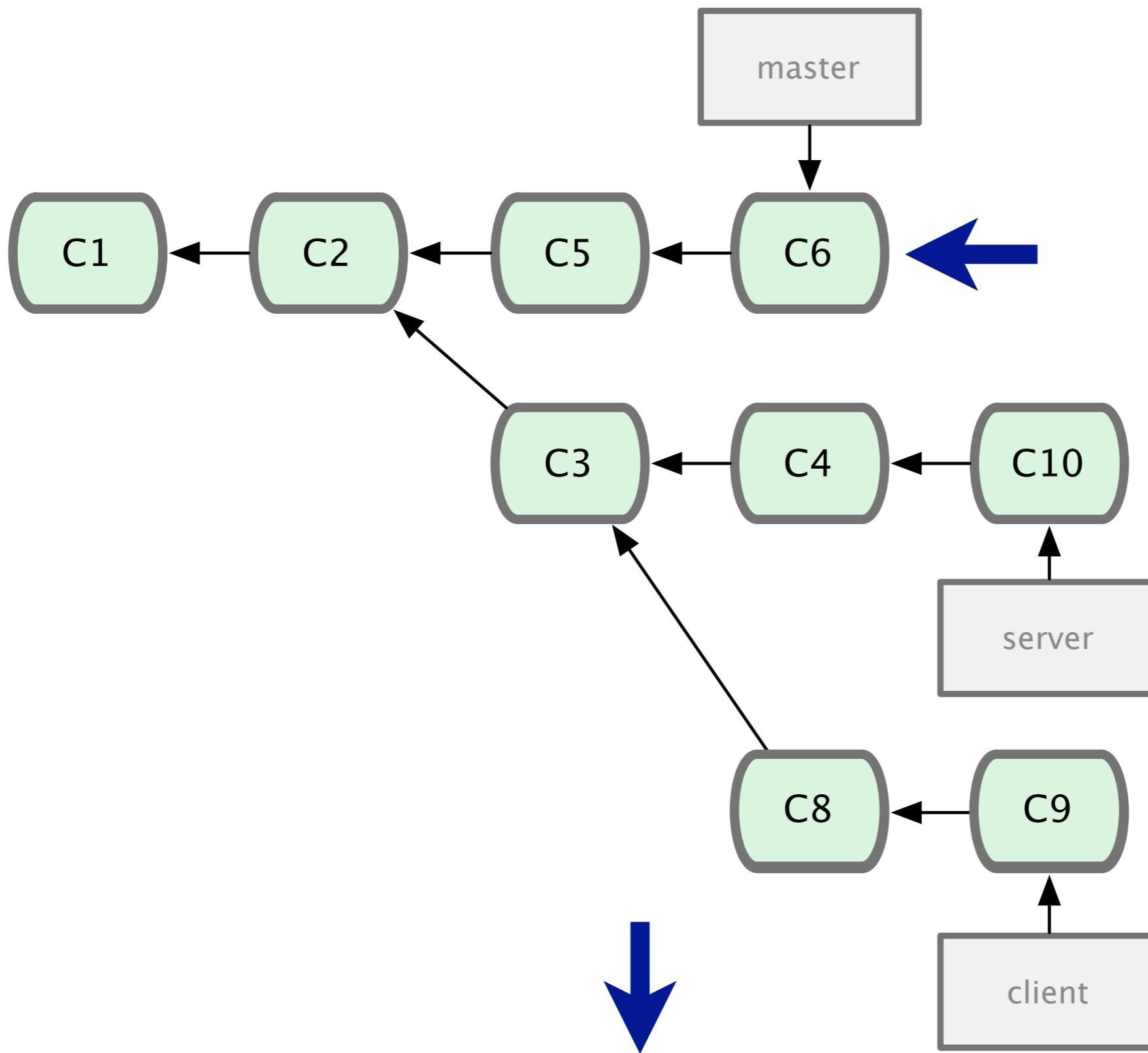
git rebase --onto master server



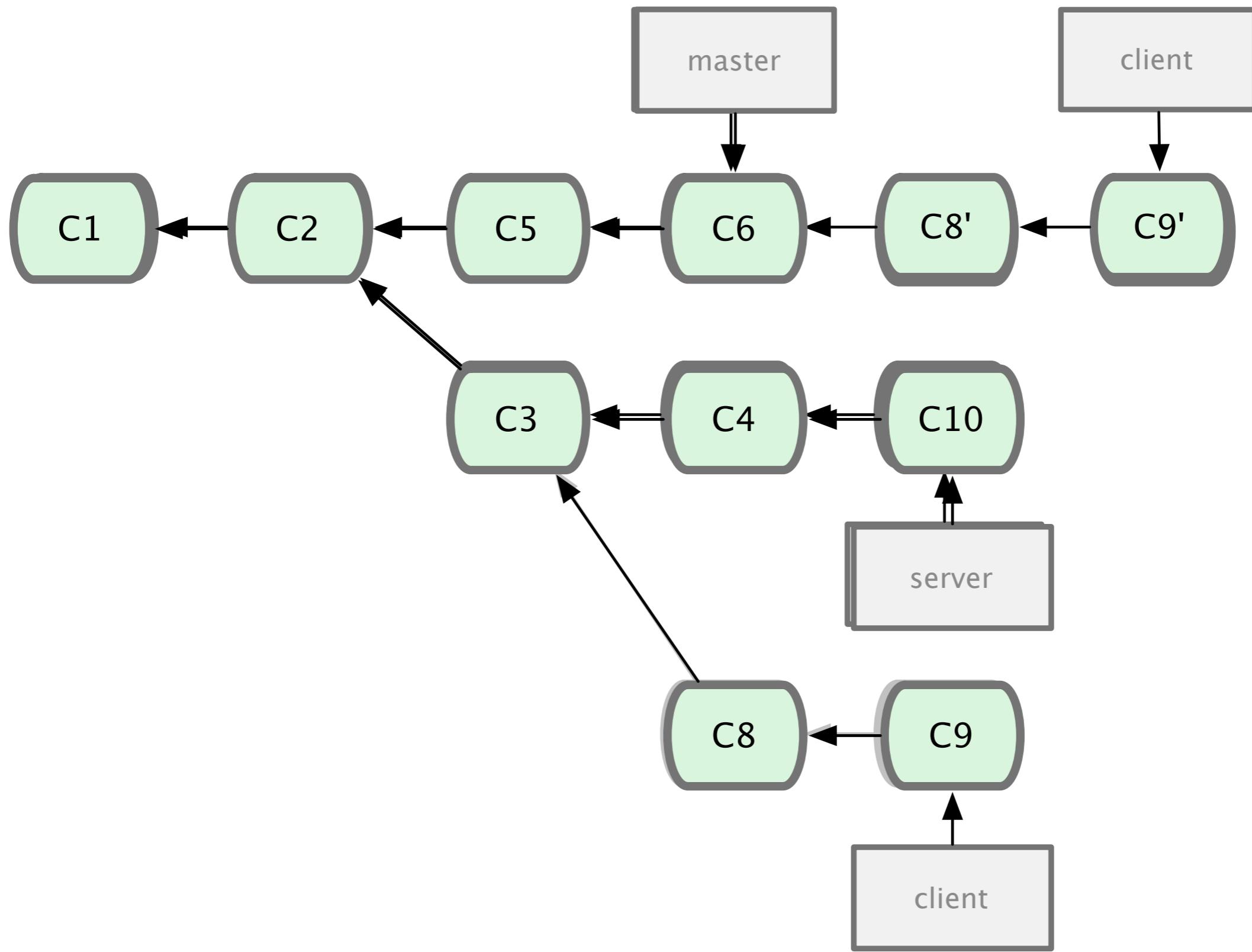
git rebase --onto master server

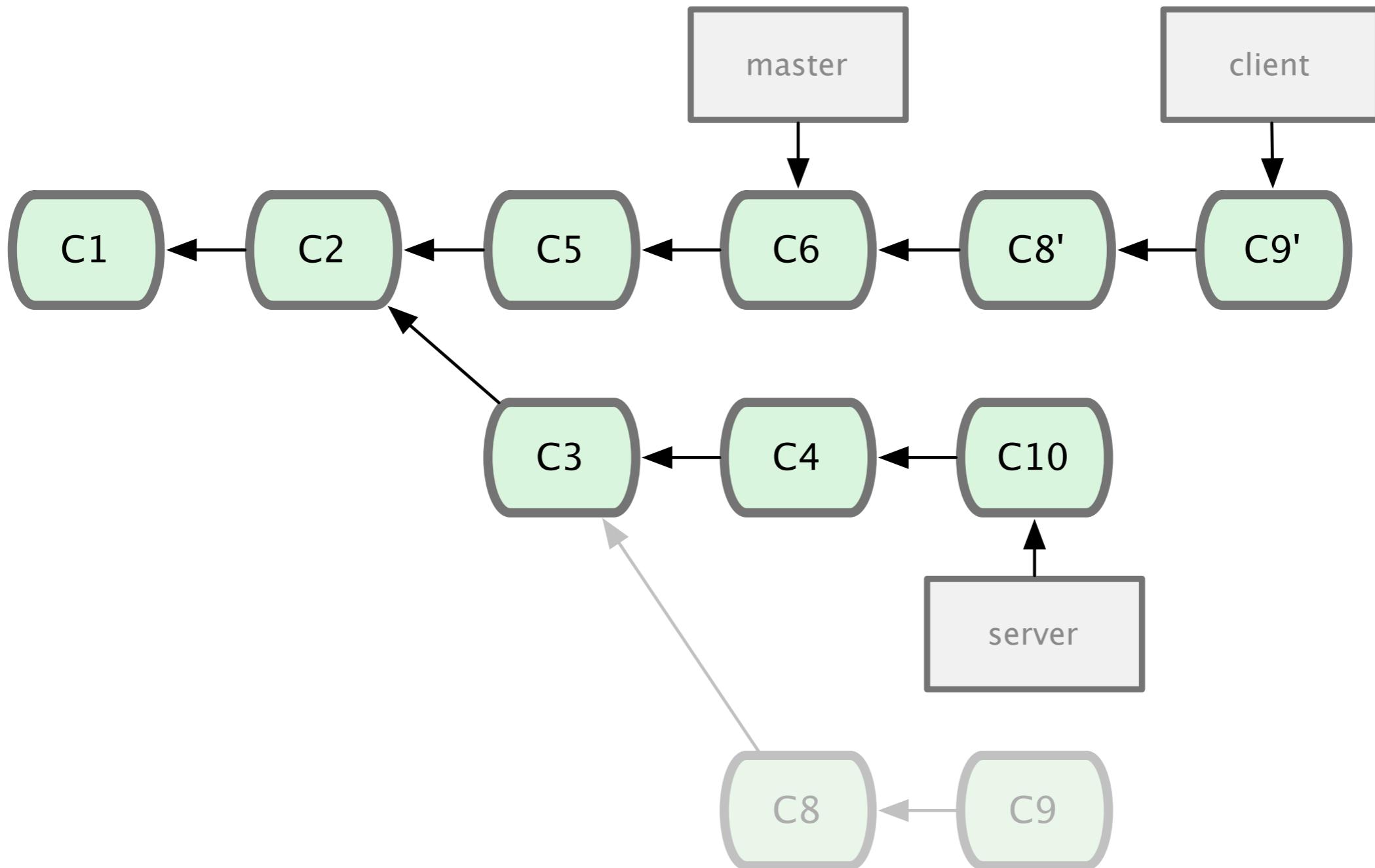


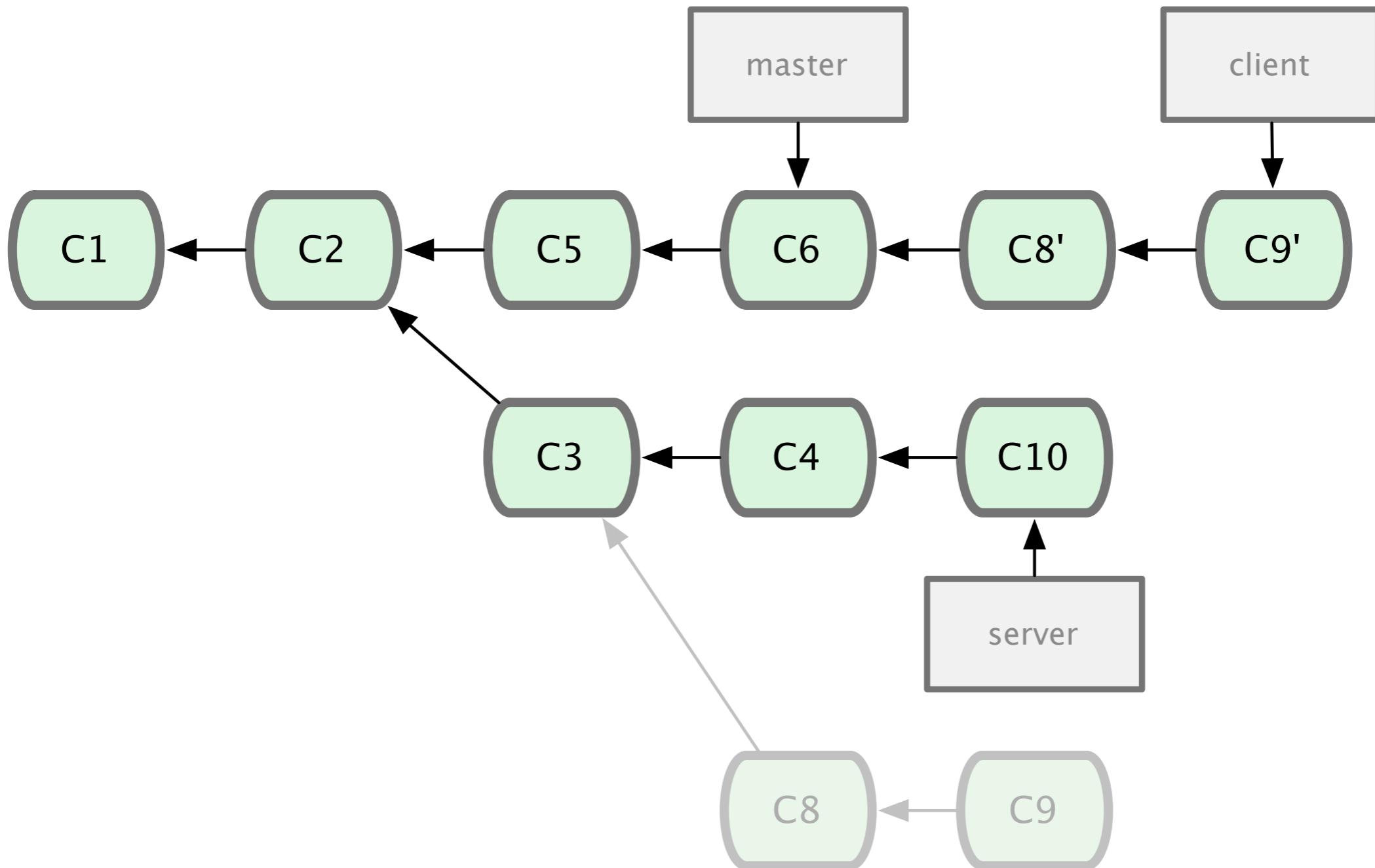
git rebase --onto master server



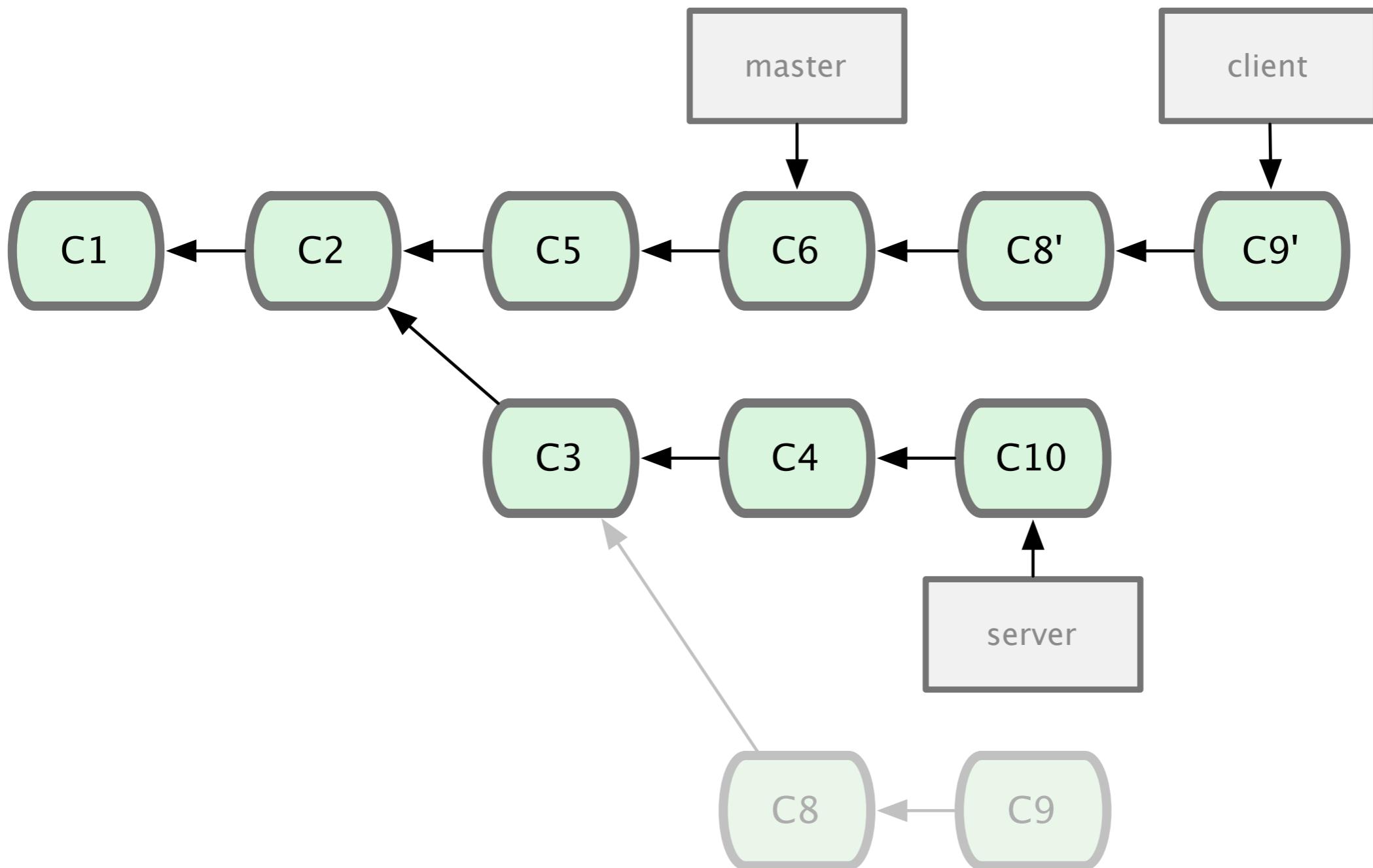
git rebase --onto master server



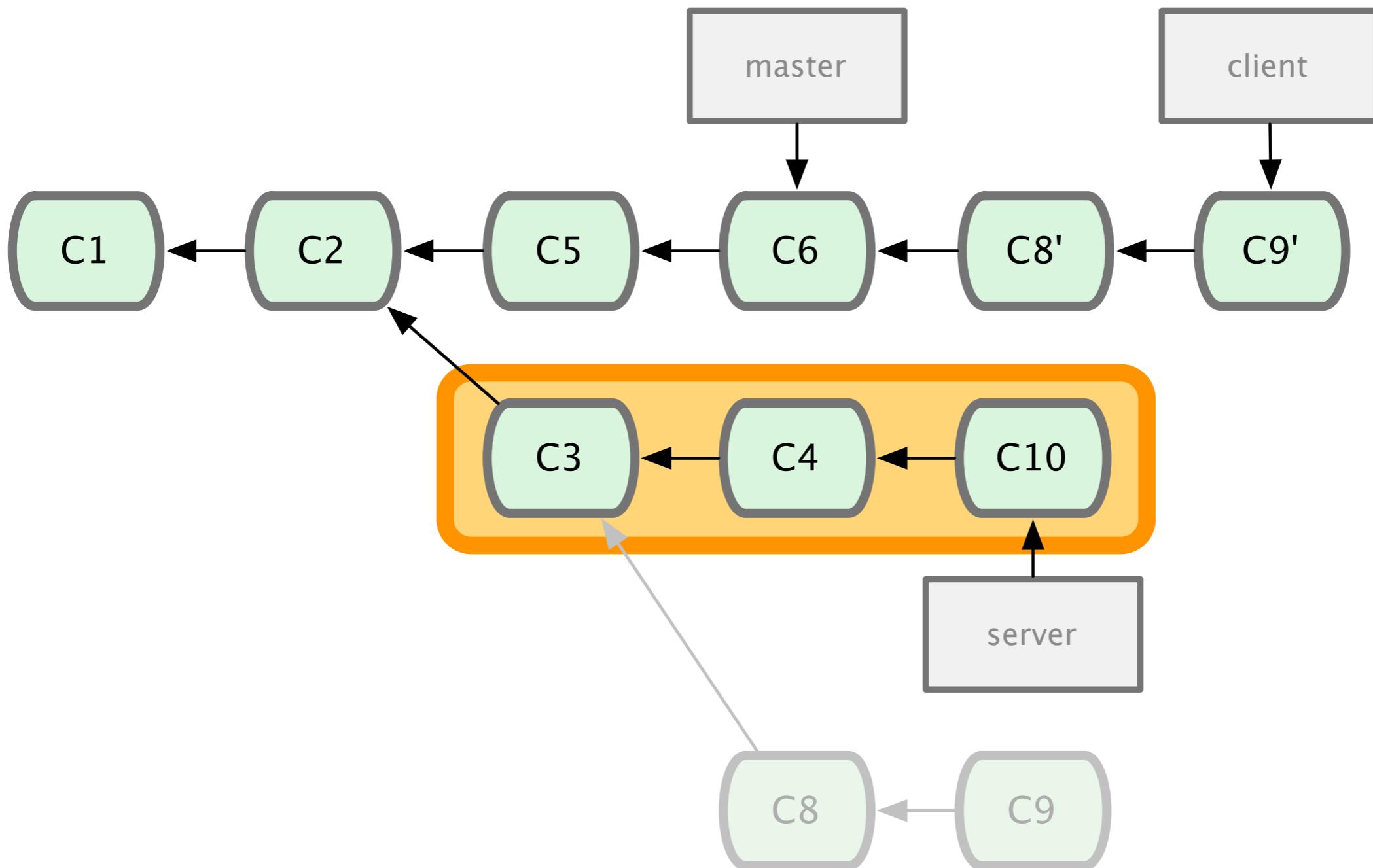




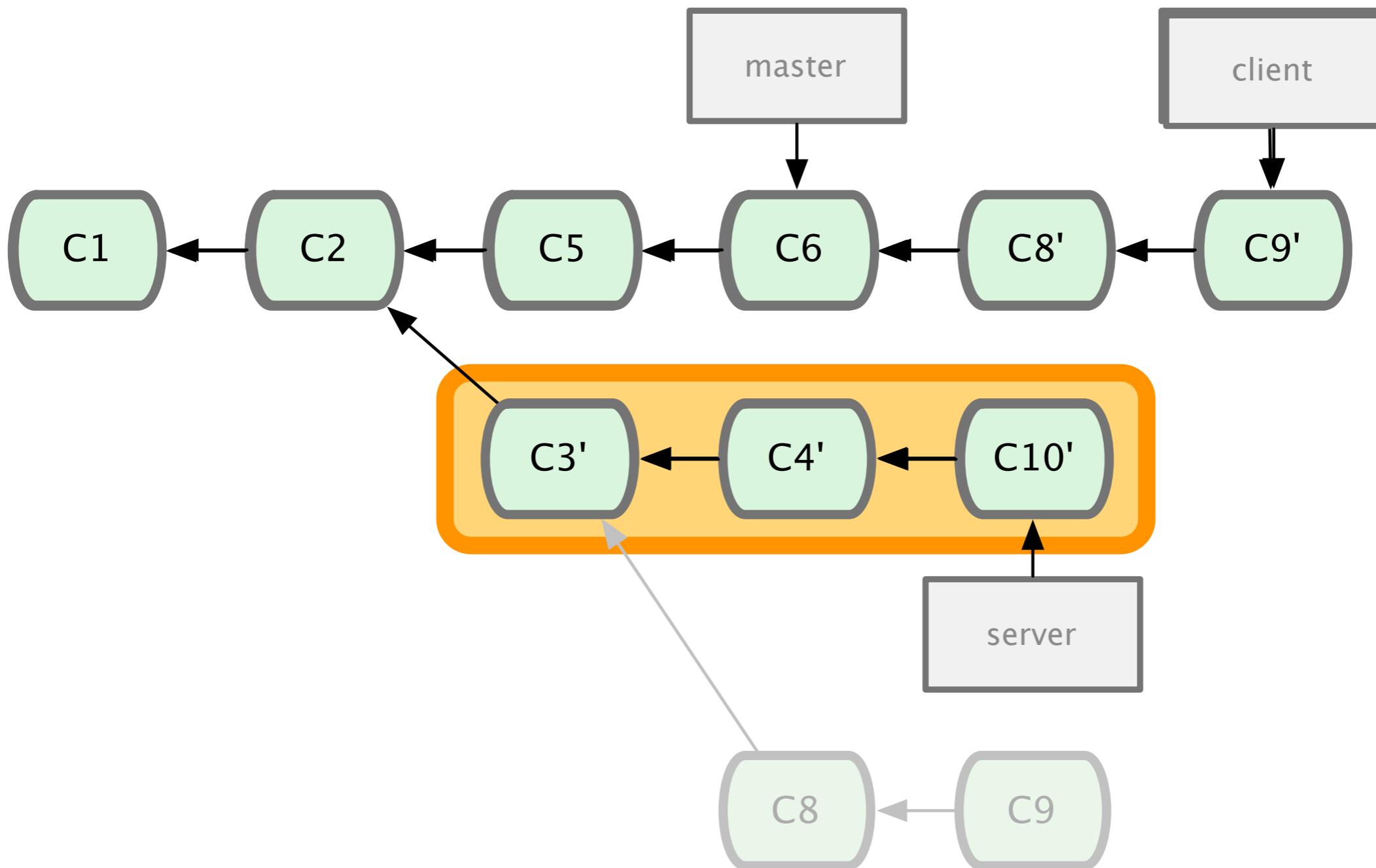
git checkout server



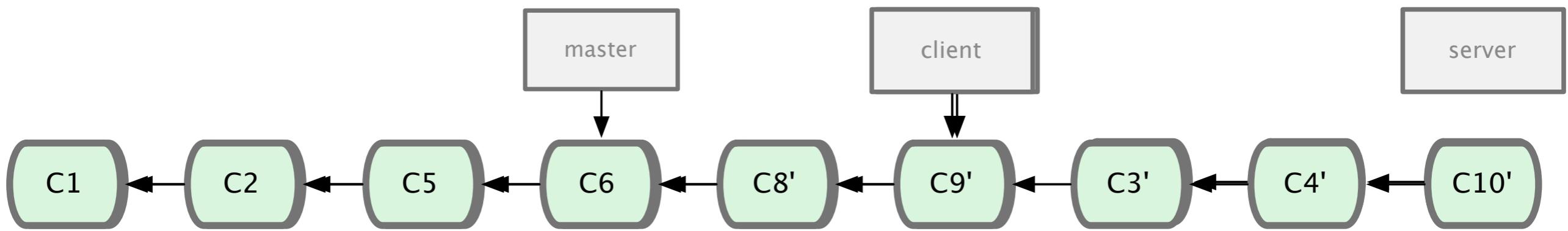
git checkout server
git rebase client



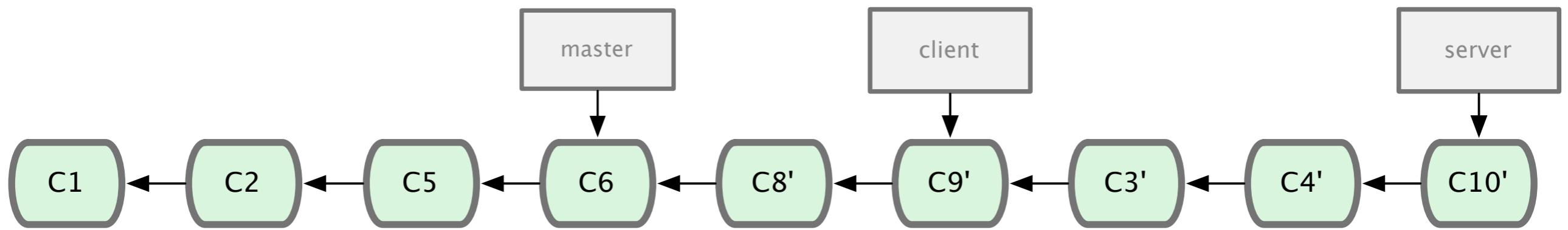
git checkout **server**
git rebase **client**



git checkout server
git rebase client

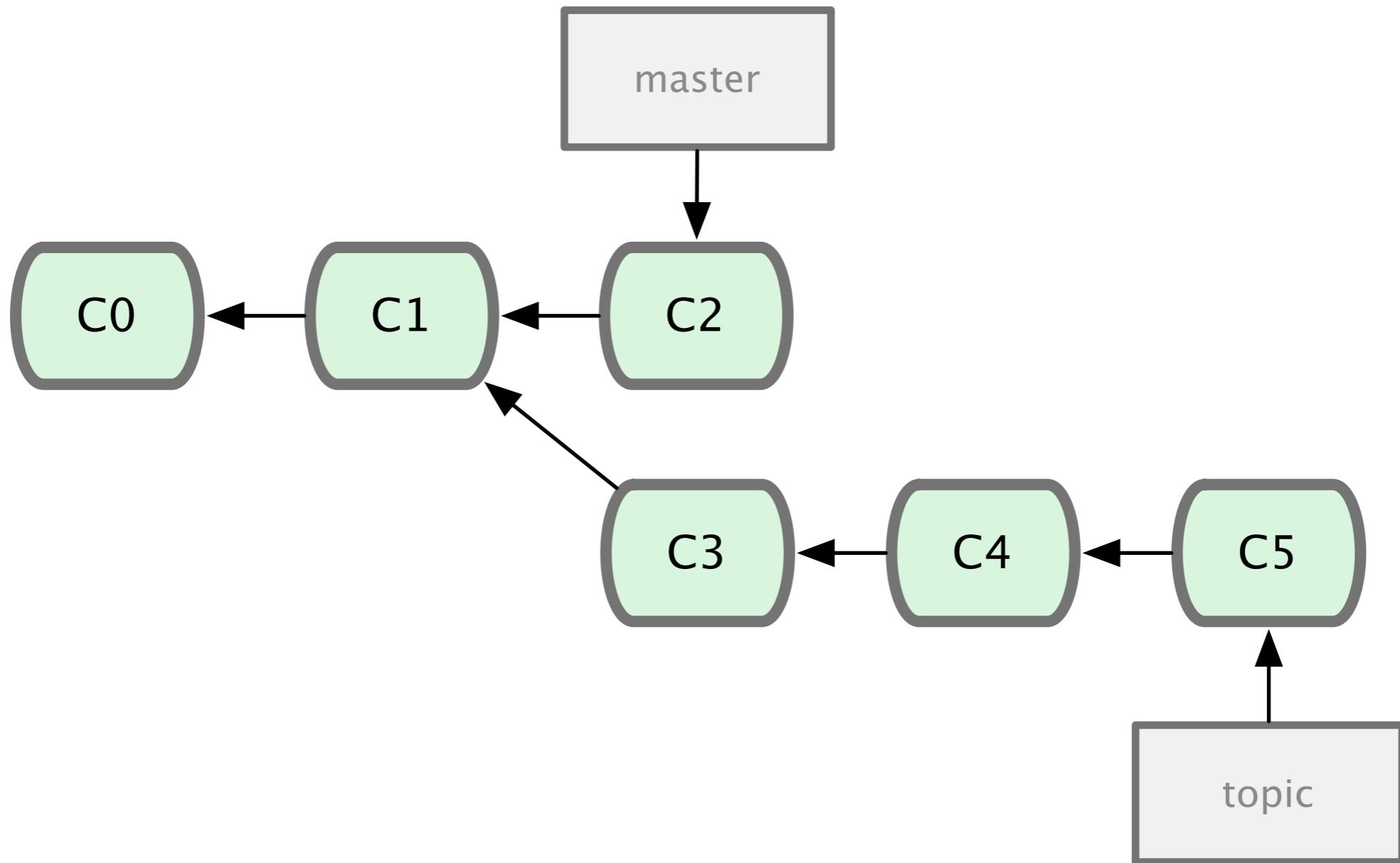


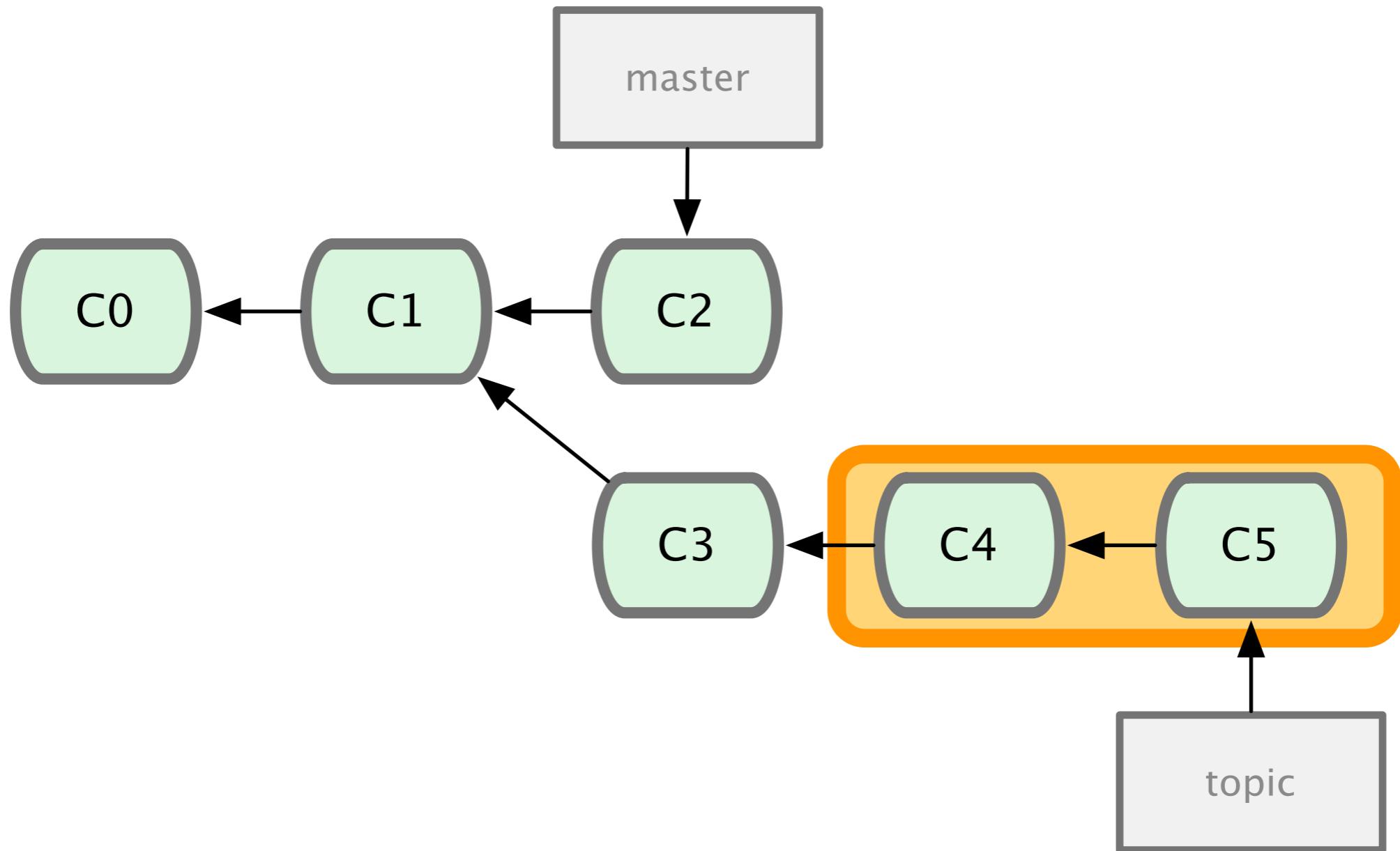
git checkout server
git rebase client

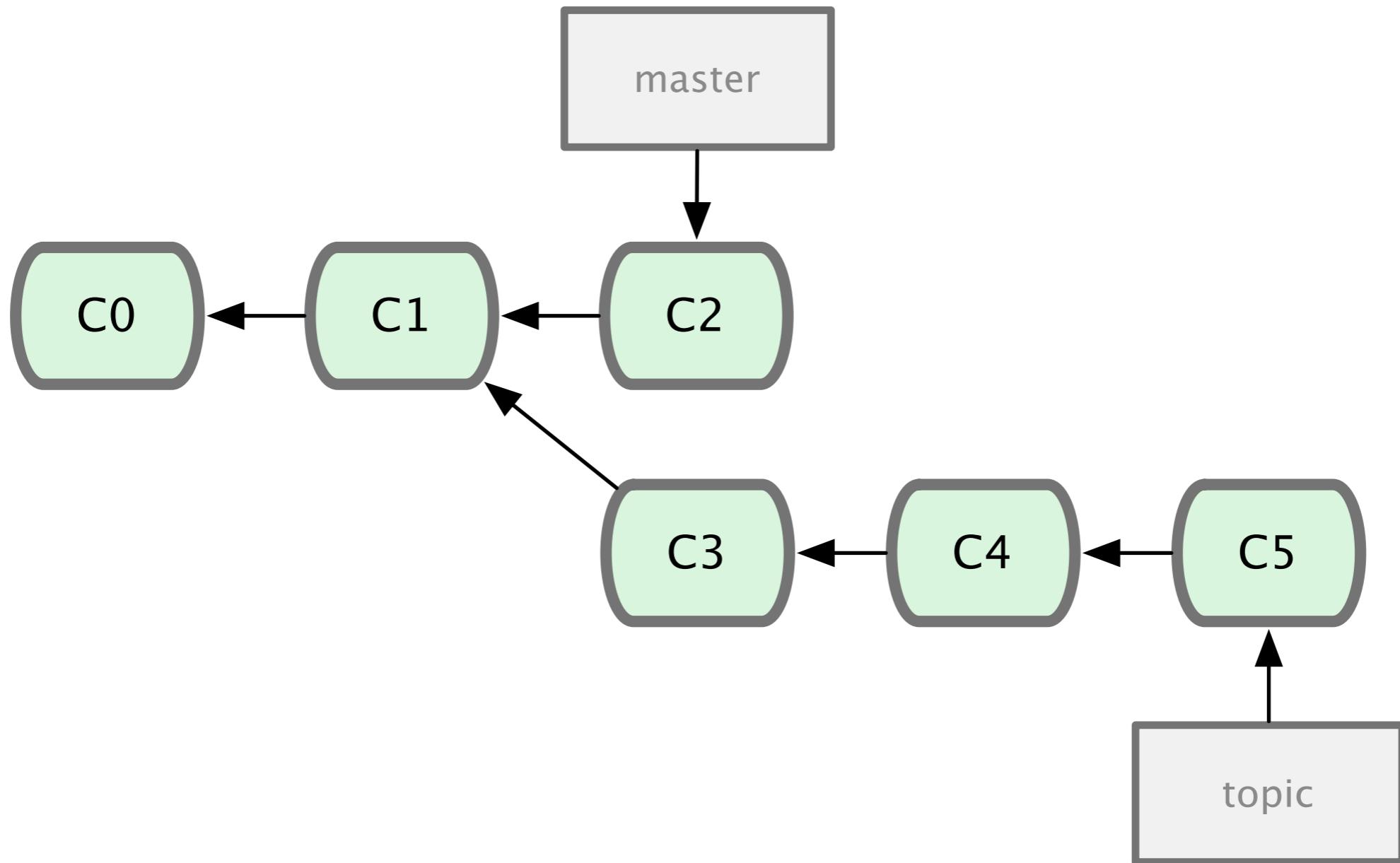


git checkout server
git rebase client

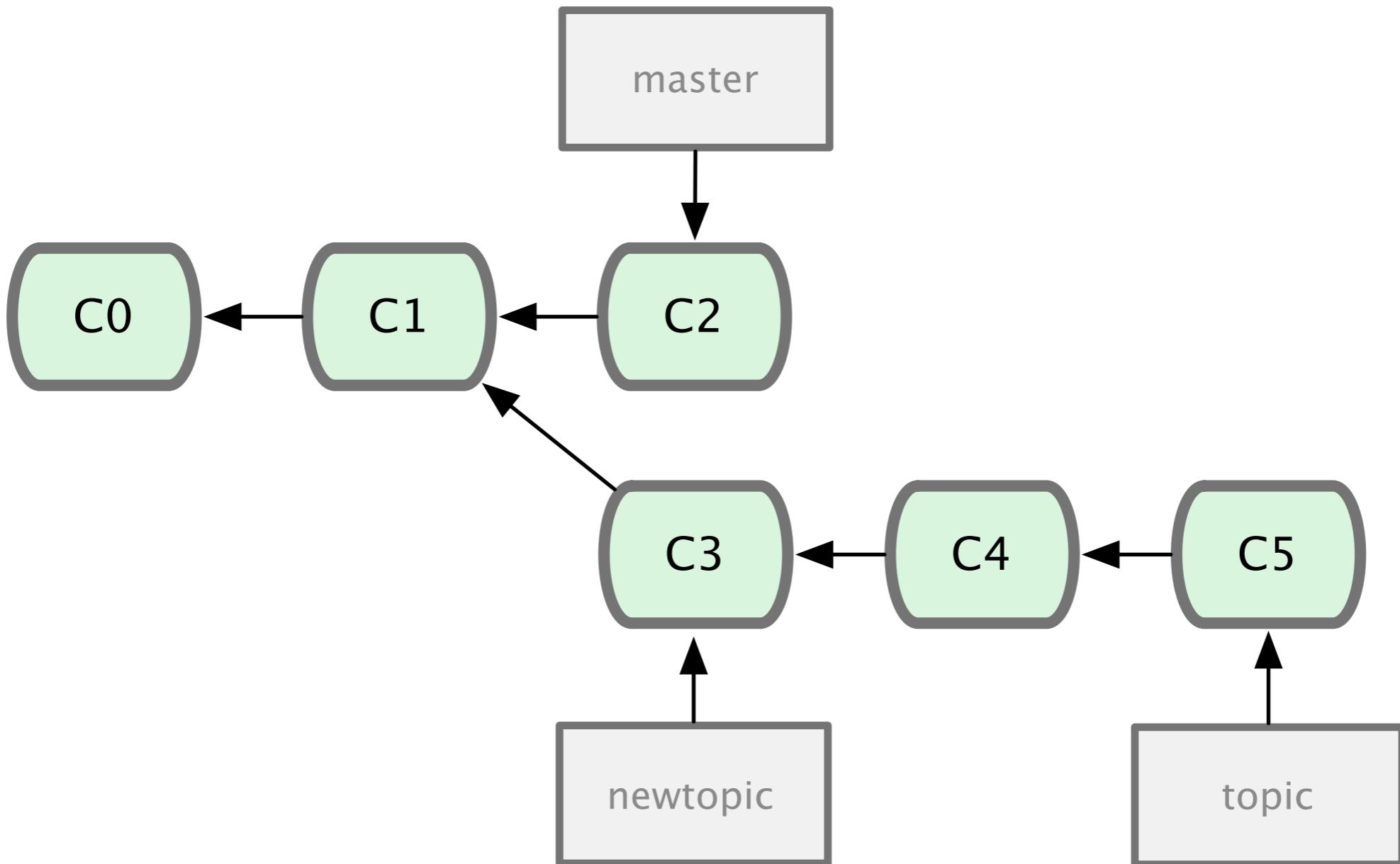
**transplant some of a
topic branch**



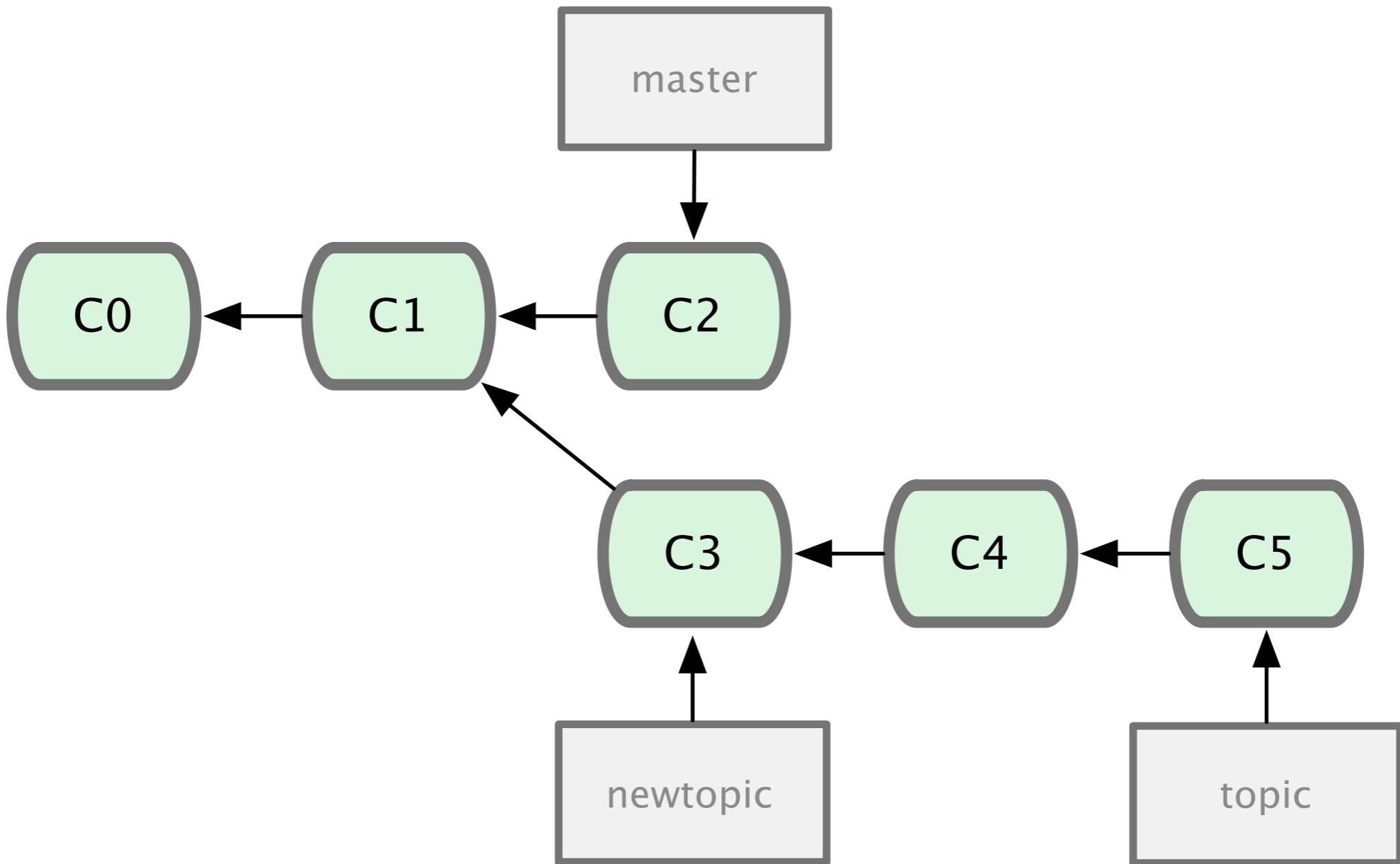




git branch newtopic C3

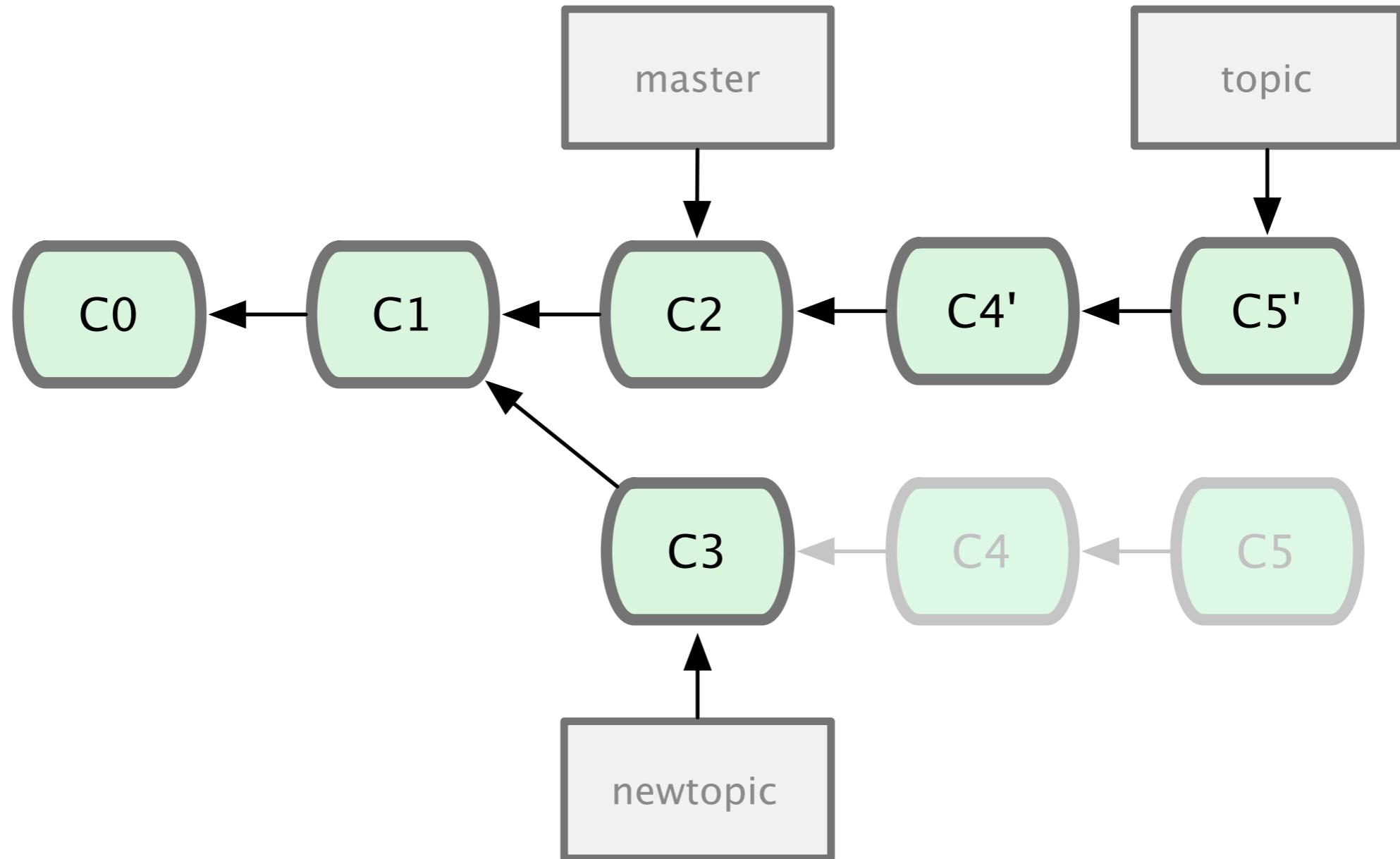


git branch newtopic C3



```
git branch newtopic C3
```

```
git rebase --onto master newtopic
```



```
git branch newtopic C3
```

```
git rebase --onto master newtopic
```

Squashing commits
together

```
pick 969c877 git apply --directory broken for new files
pick b75271d git diff <tree>{3,}: do not reverse order of args
pick 72d404d test-lib: fix broken printf
```

```
# Rebase f285a2d..5c283eb onto f285a2d
#
# Commands:
# p, pick = use commit
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
~ ~ ~ ~ ~
"~/projects/git/.git/rebase-merge/git-rebase-todo" 14L, 472C
```

```
pick 969c877 git apply --directory broken for new files
squash b75271d git diff <tree>{3,}: do not reverse order of args
squash 72d404d test-lib: fix broken printf
```

```
# Rebase f285a2d..5c283eb onto f285a2d
#
# Commands:
# p, pick = use commit
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
~ ~ ~ ~ ~
"~/projects/git/.git/rebase-merge/git-rebase-todo" 14L, 472C
```

```
# This is a combination of 3 commits.
# The first commit's message is:
git apply --directory broken for new files

# This is the 2nd commit message:

git diff <tree>{3,}: do not reverse order of args

# This is the 3rd commit message:

test-lib: fix broken printf

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Author:    Jeff King <peff@peff.net>
#
# Not currently on any branch.
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:    builtin-apply.c
# modified:    builtin-diff.c
# modified:    t/t4013-diff-various.sh
# new file:    t/t4013/diff.diff_master_master^_side
# modified:    t/t4128-apply-root.sh
# modified:    t/test-lib.sh
#
~
~

".git/COMMIT_EDITMSG" 39L, 1454C
```

```
# This is a combination of 3 commits.
# The first commit's message is:
git apply --directory broken for new files

# This is the 2nd commit message:

git diff <tree>{3,}: do not reverse order of args

# This is the 3rd commit message:

test-lib: fix broken printf

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Author:    Jeff King <peff@peff.net>
#
# Not currently on any branch.
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:    builtin-apply.c
# modified:    builtin-diff.c
# modified:    t/t4013-diff-various.sh
# new file:    t/t4013/diff.diff_master_master^_side
# modified:    t/t4128-apply-root.sh
# modified:    t/test-lib.sh
#
~
~

".git/COMMIT_EDITMSG" 39L, 1454C
```

```
# This is a combination of 3 commits.
# The first commit's message is:
git apply --directory broken for new files

# This is the 2nd commit message:

git diff <tree>{3,}: do not reverse order of args

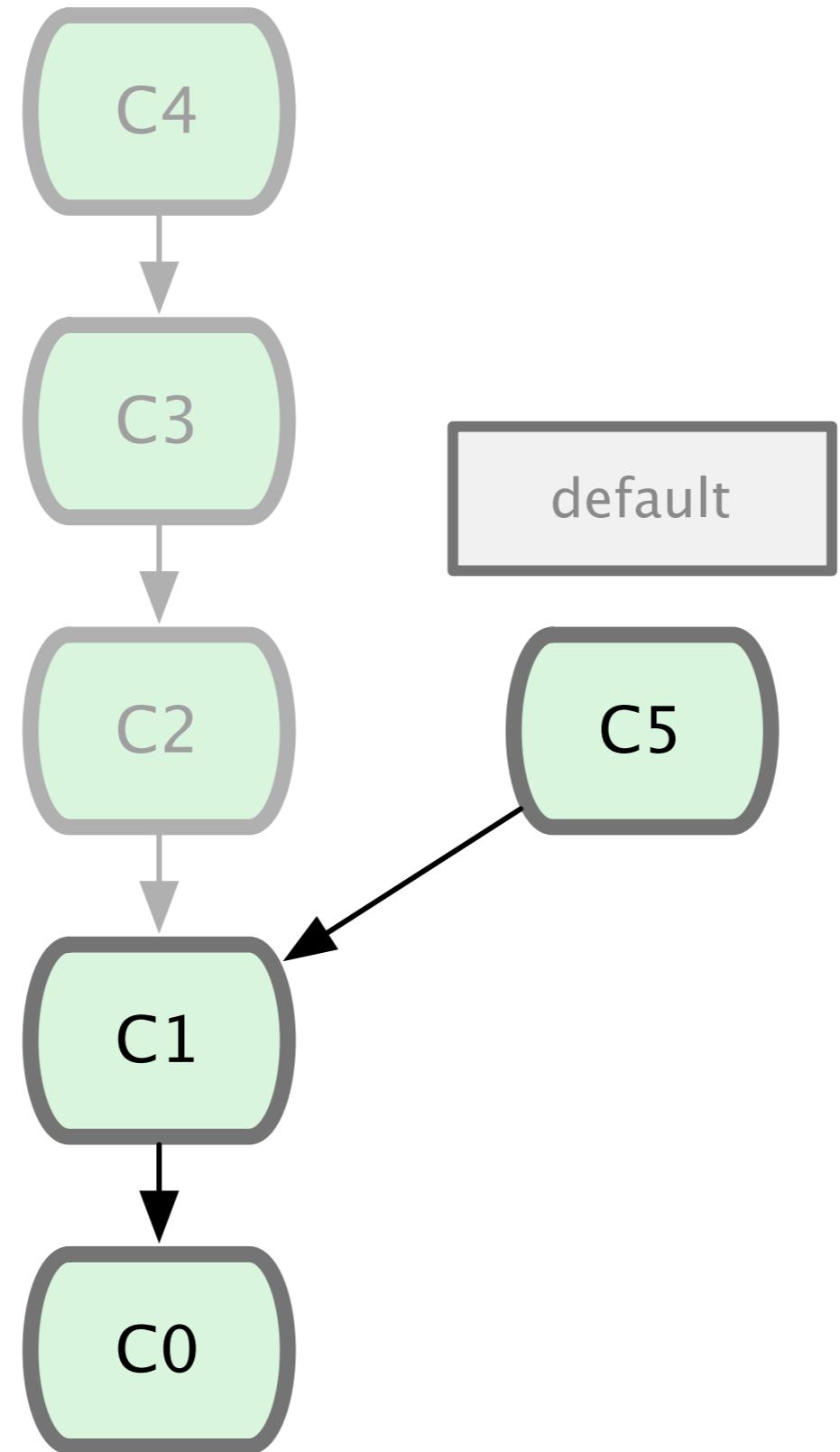
# This is the 3rd commit message:

test-lib: fix broken printf

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Author:    Jeff King <peff@peff.net>
#
# Not currently on any branch.
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:    builtin-apply.c
# modified:    builtin-diff.c
# modified:    t/t4013-diff-various.sh
# new file:    t/t4013/diff.diff_master_master^_side
# modified:    t/t4128-apply-root.sh
# modified:    t/test-lib.sh
#
~
~

".git/COMMIT_EDITMSG" 39L, 1454C
```

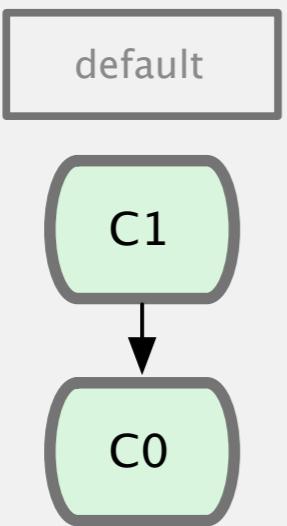
```
# This is a combination of 3 commits.  
# The first commit's message is:  
git apply --directory broken for new files  
  
# This is the 2nd commit message:  
  
git diff <tree>{3,}: do not reverse order of args  
  
# This is the 3rd commit message:  
  
test-lib: fix broken printf  
  
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the commit.  
#  
# Author: Jeff King <peff@peff.net>  
#  
# Not currently on any branch.  
# Changes to be committed:  
#   (use "git reset HEAD <file>..." to unstage)  
#  
# modified: builtin-apply.c  
# modified: builtin-diff.c  
# modified: t/t4013-diff-various.sh  
# new file: t/t4013/diff.diff_master_master^_side  
# modified: t/t4128-apply-root.sh  
# modified: t/test-lib.sh  
#  
~  
~  
".git/COMMIT_EDITMSG" 39L, 1454C
```



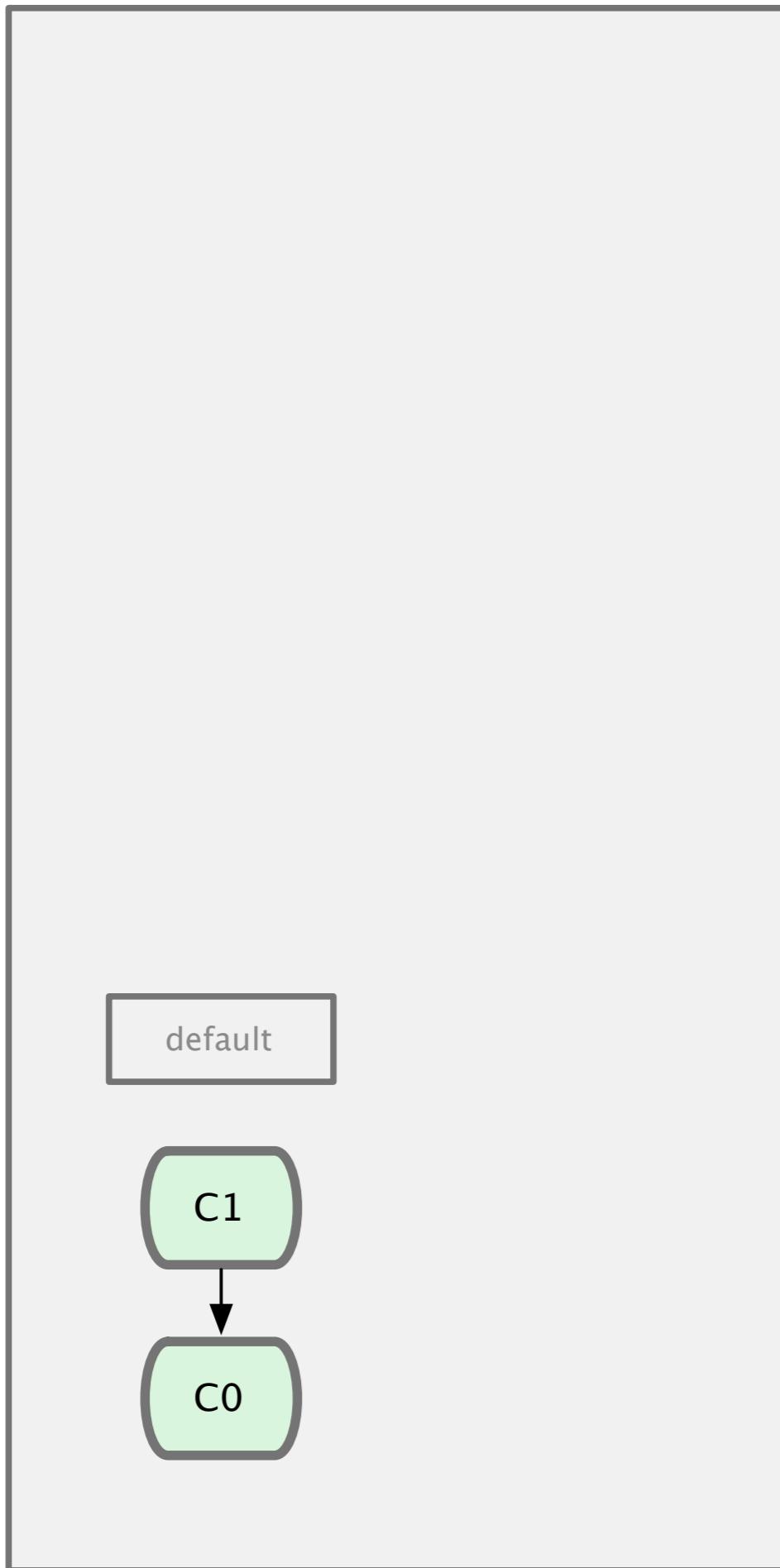
The Perils

scott

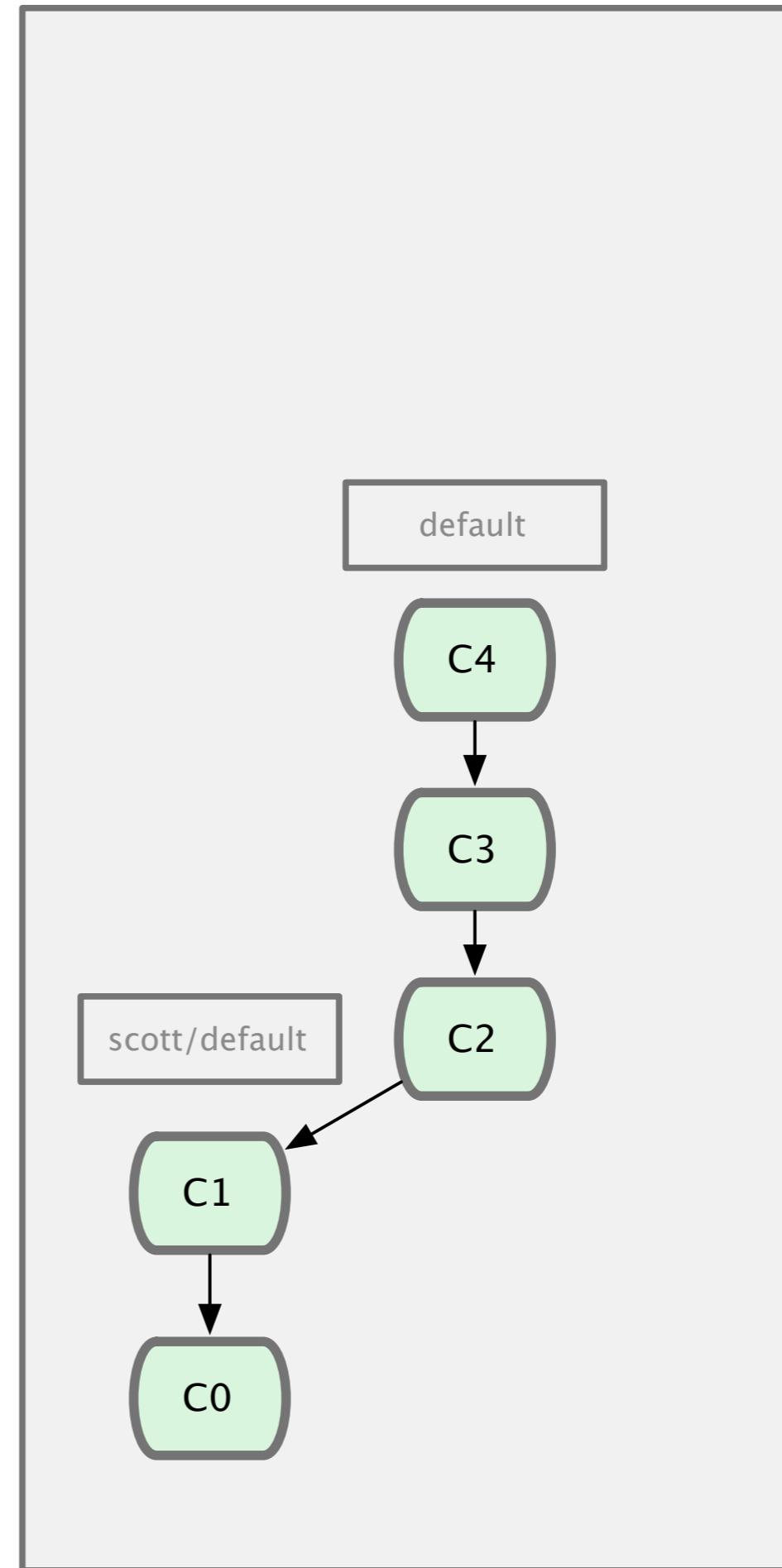
jessica



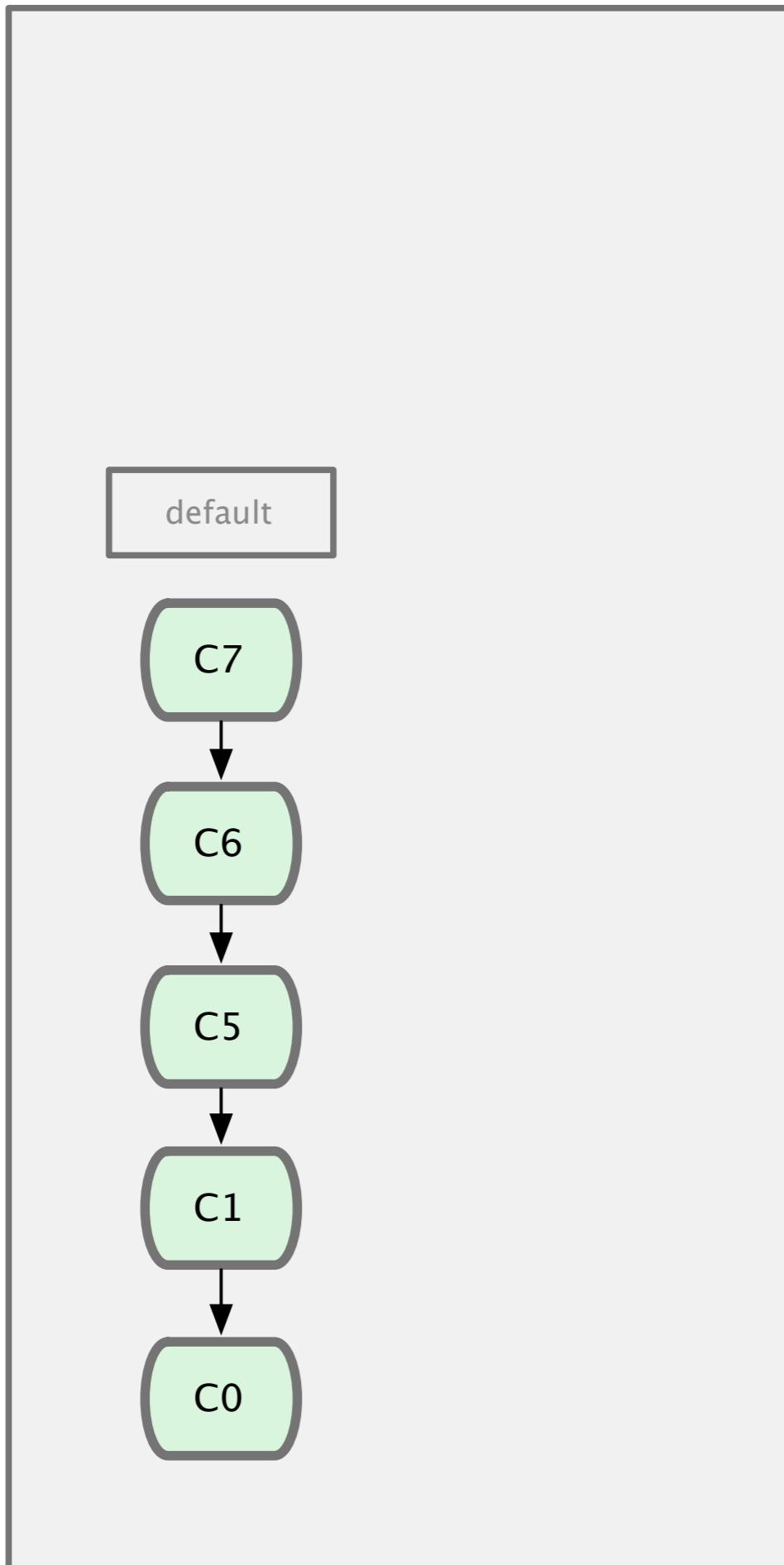
scott



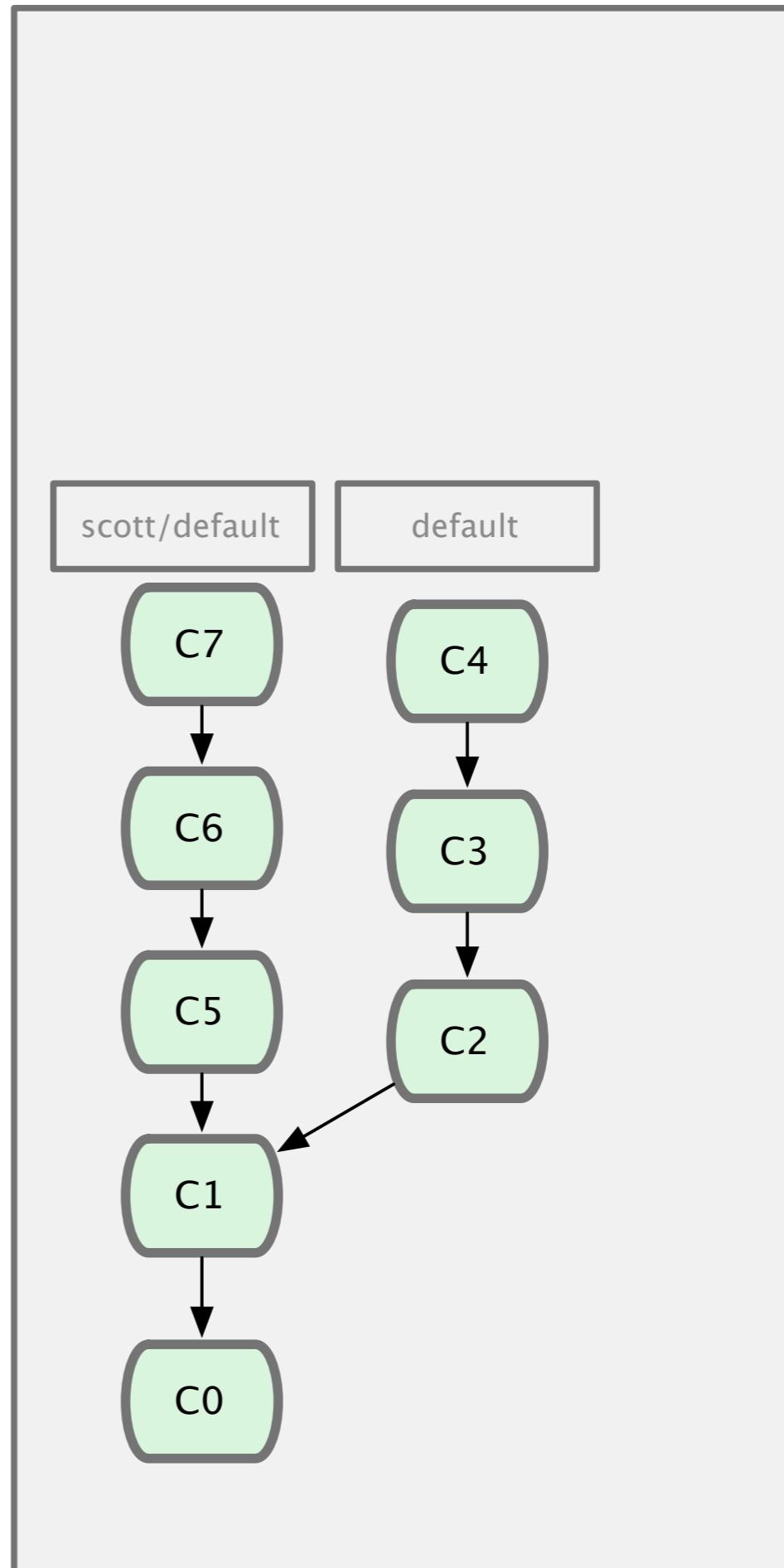
jessica



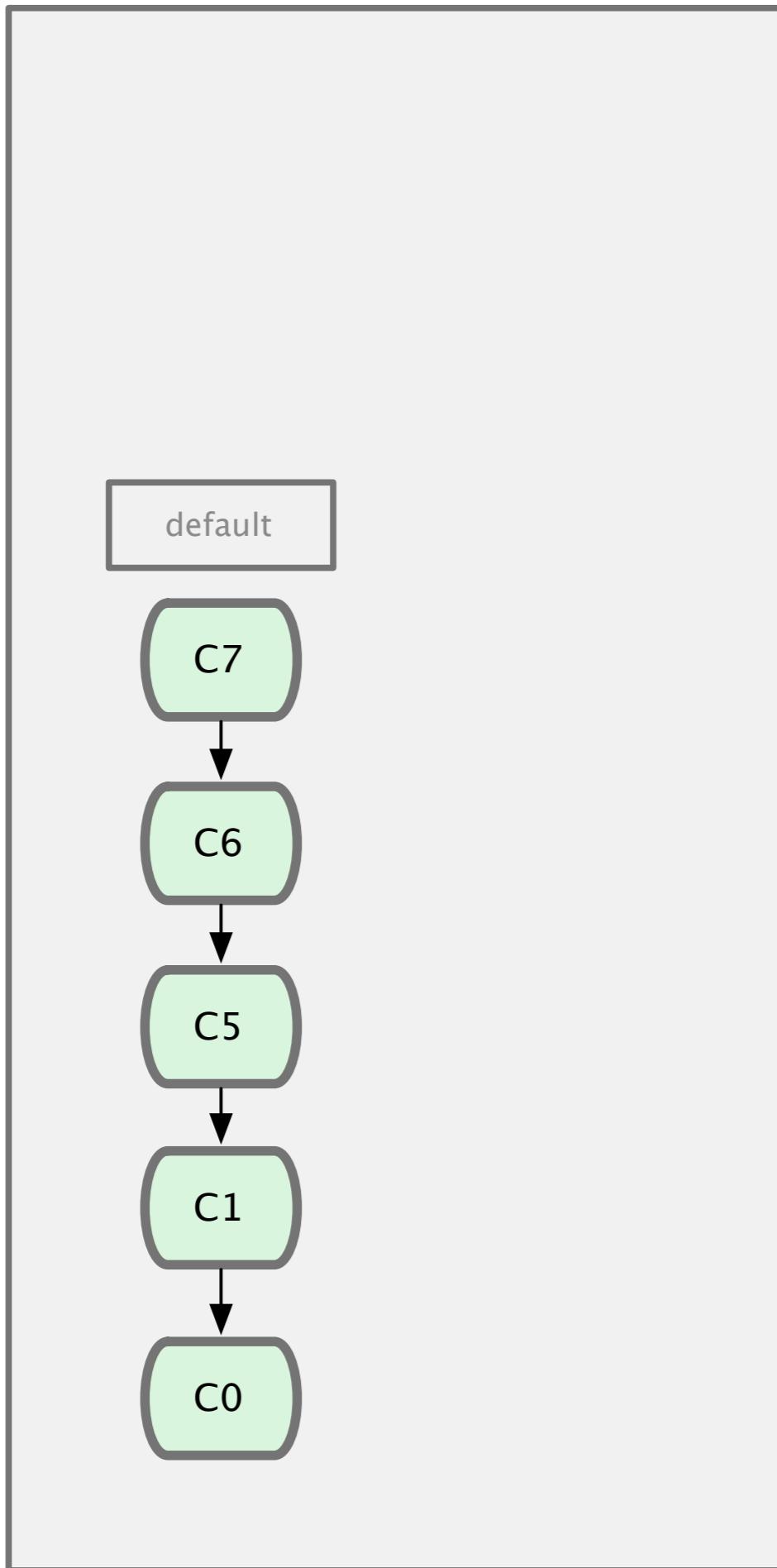
scott



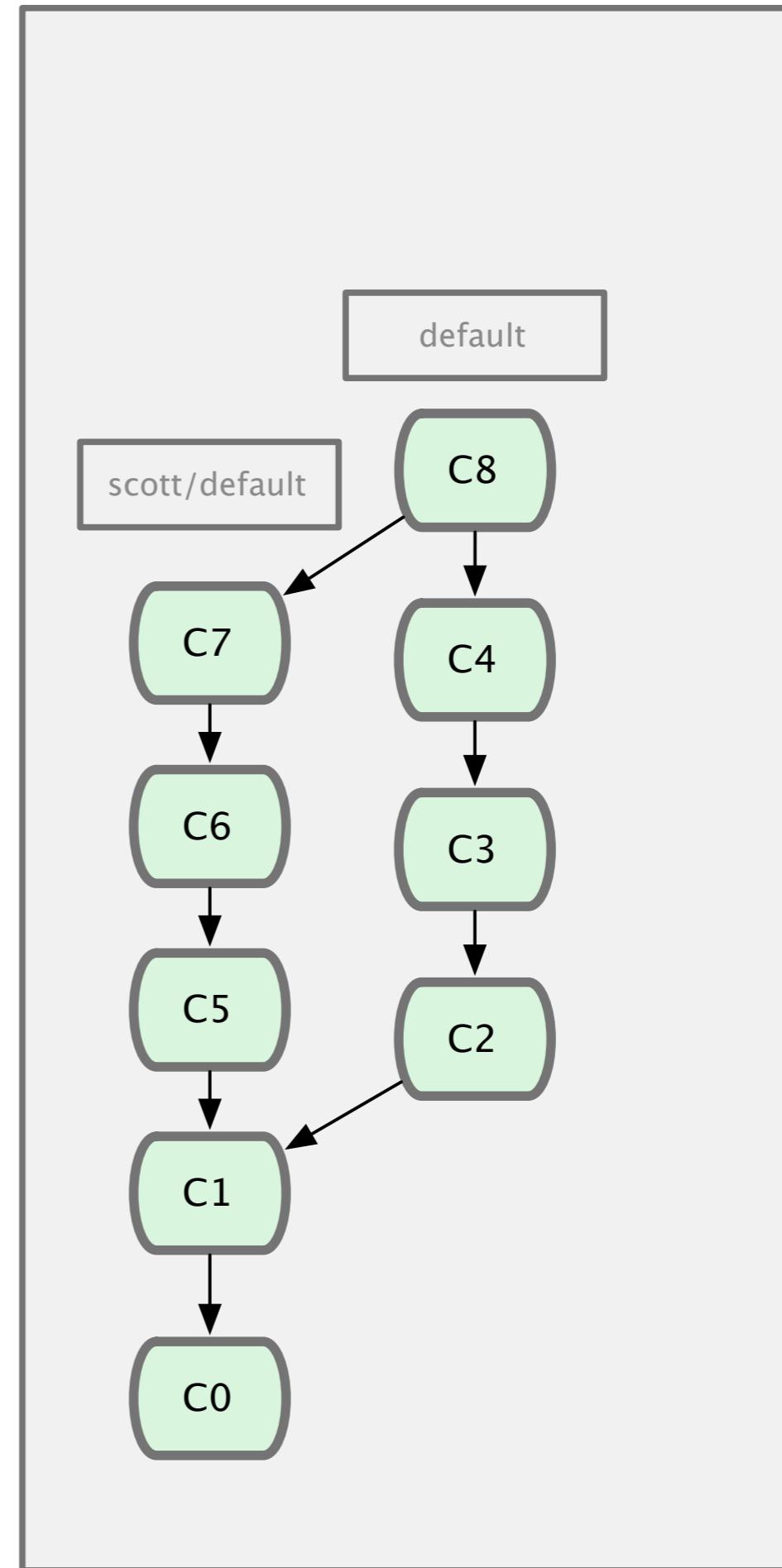
jessica



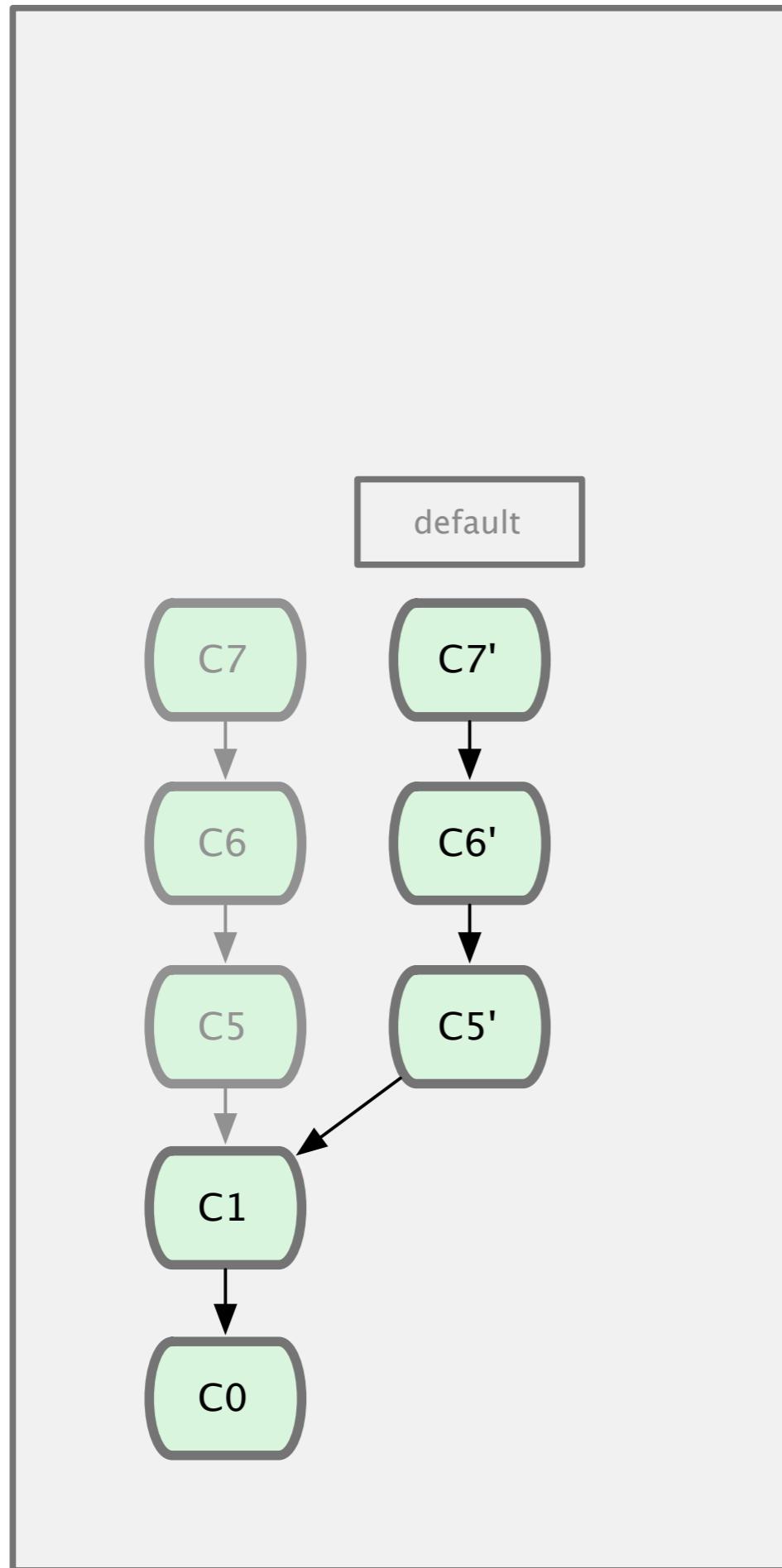
scott



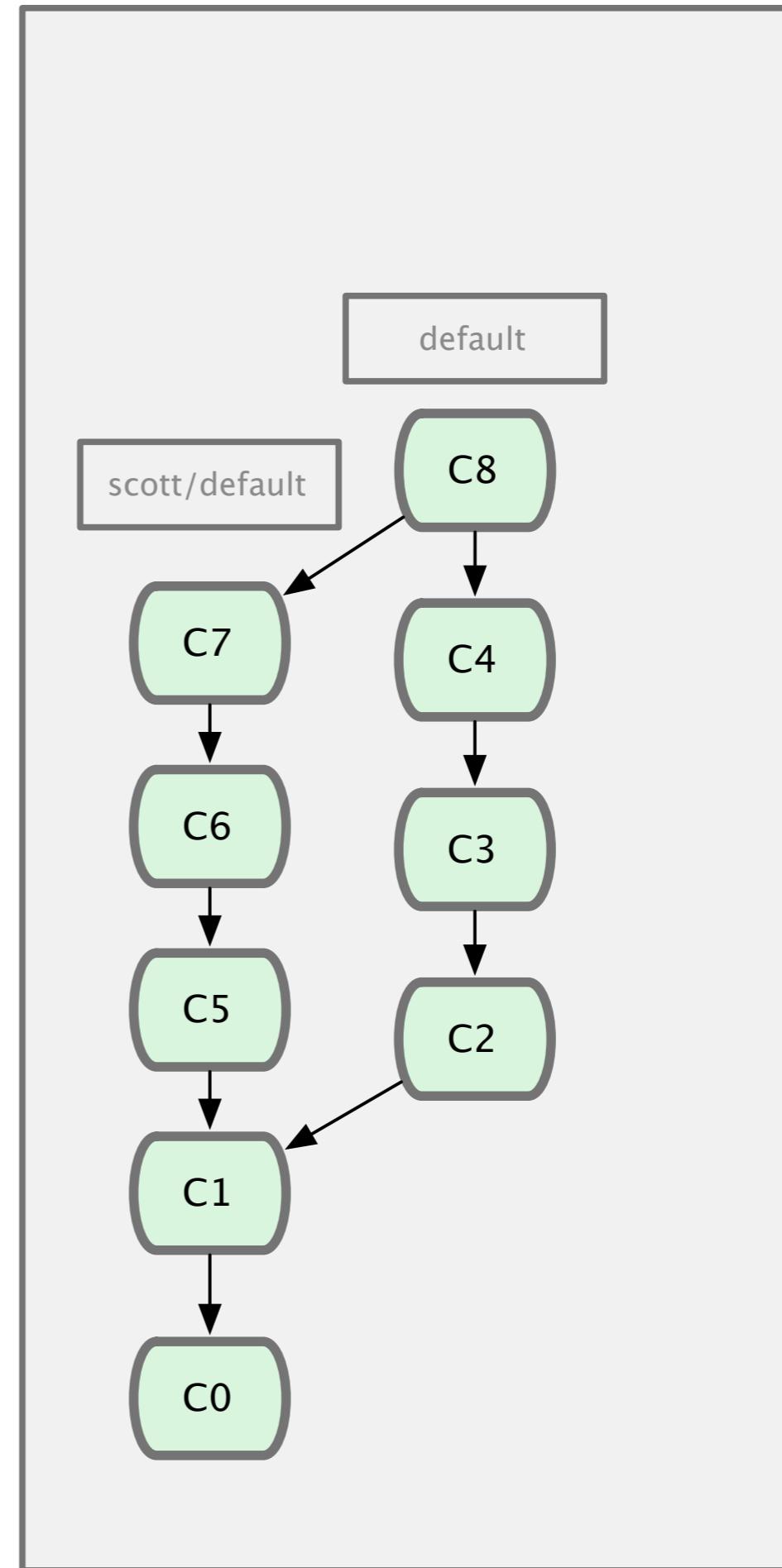
jessica



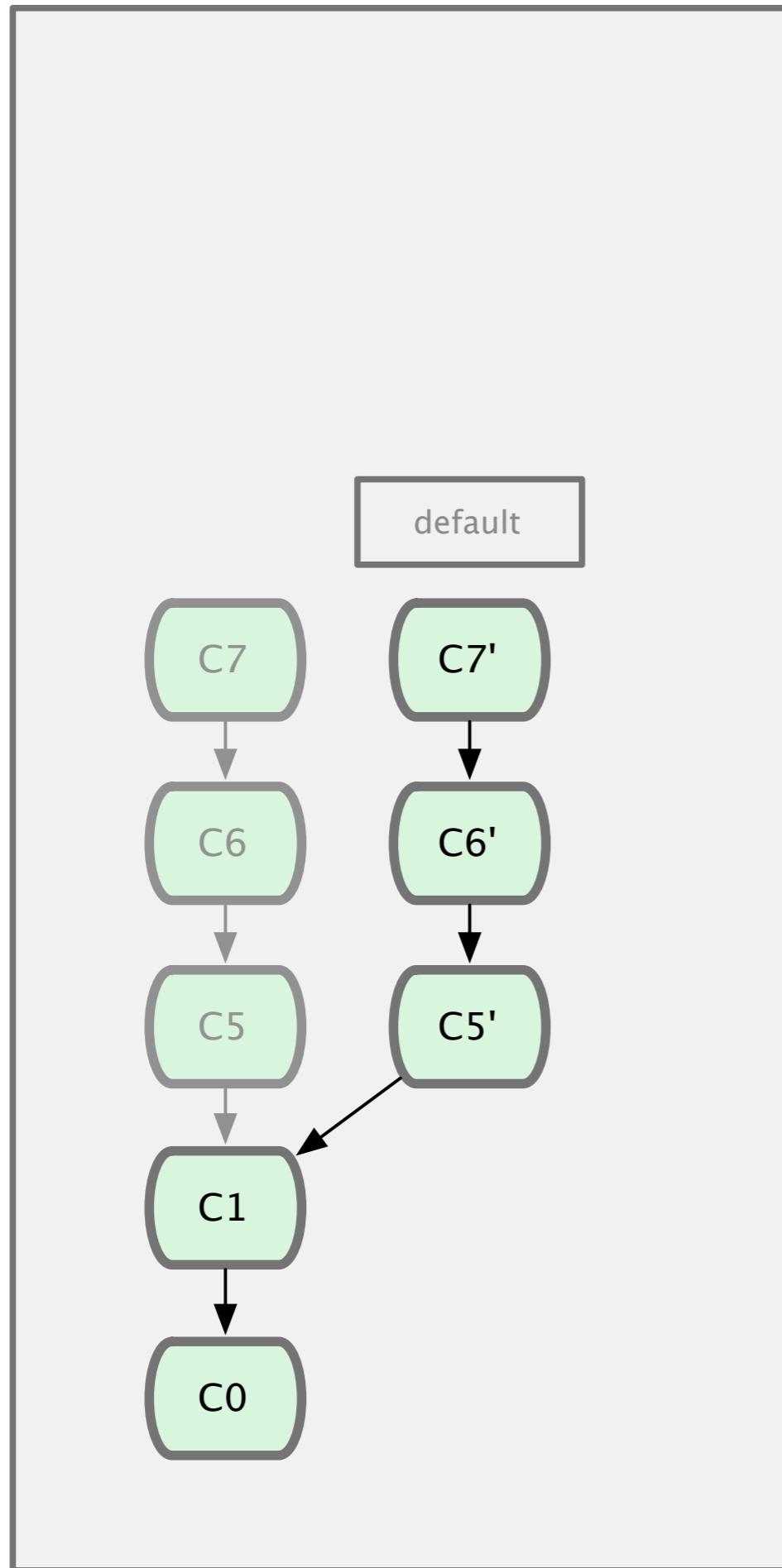
scott



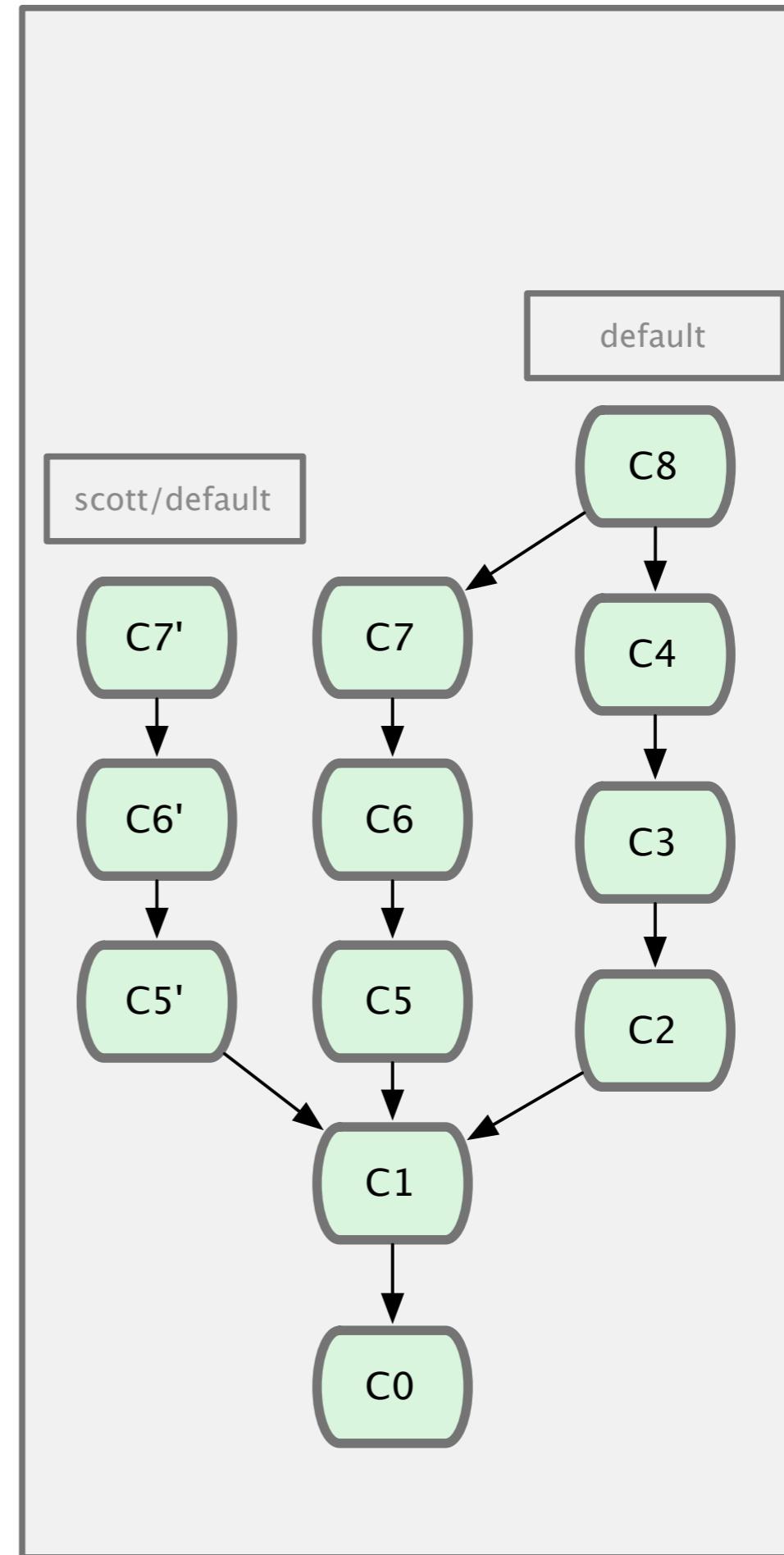
jessica



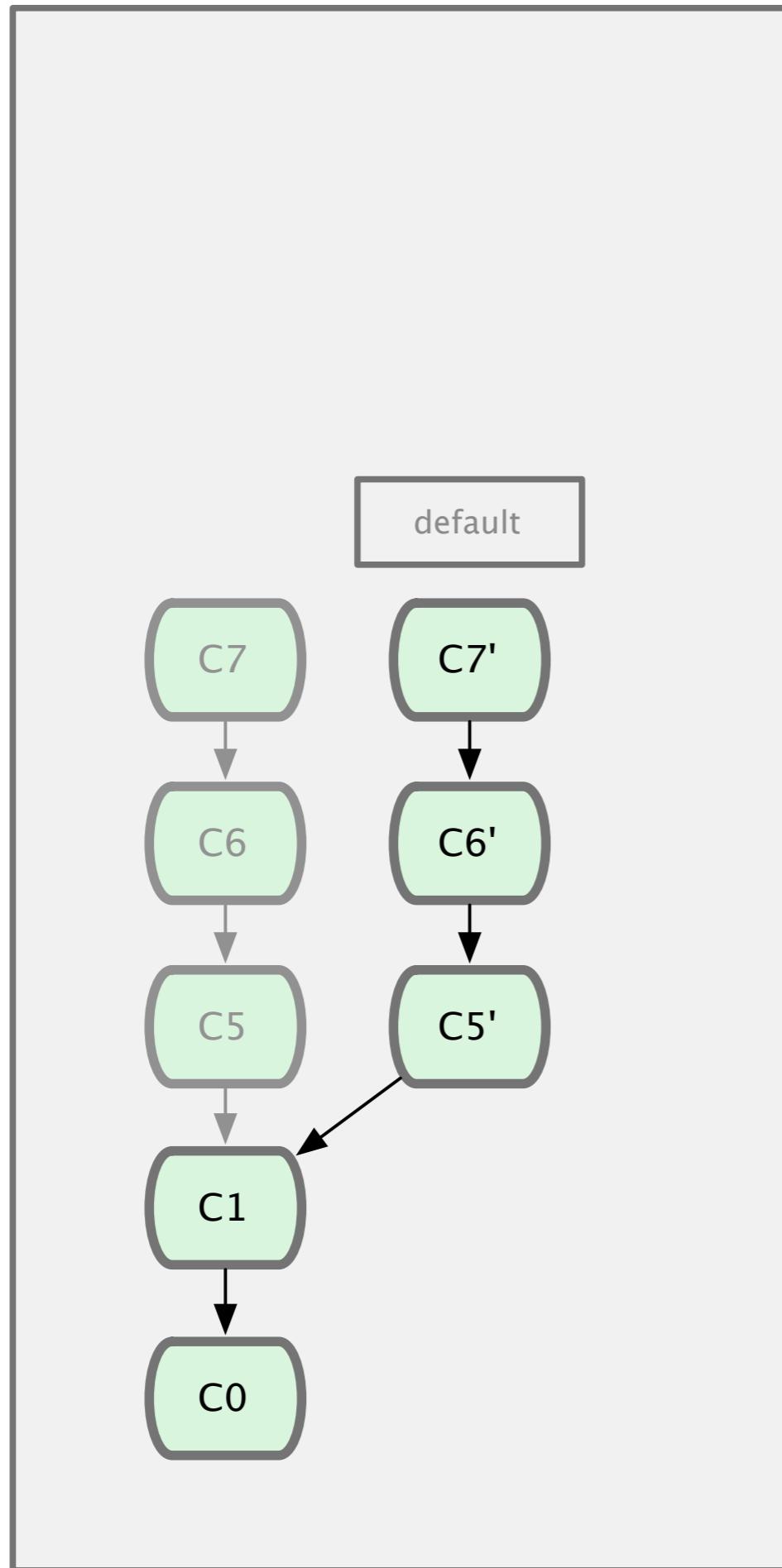
scott



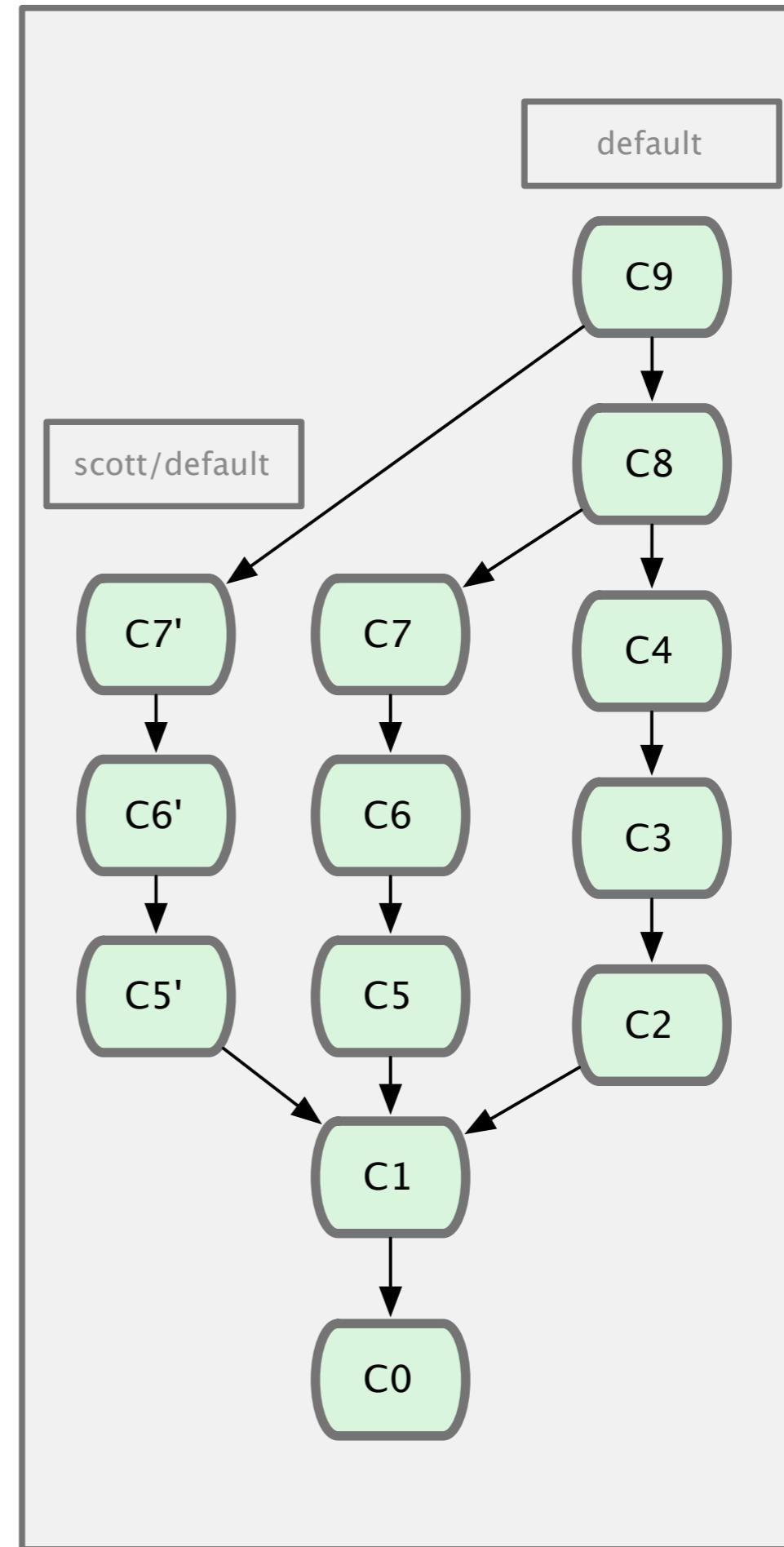
jessica



scott



jessica

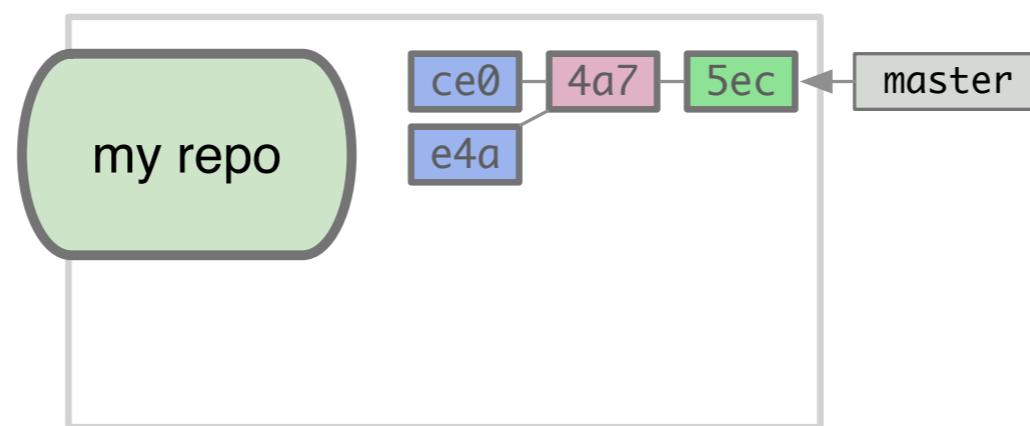


**use your branches as
patch queues!**

Remotes

developer
nick

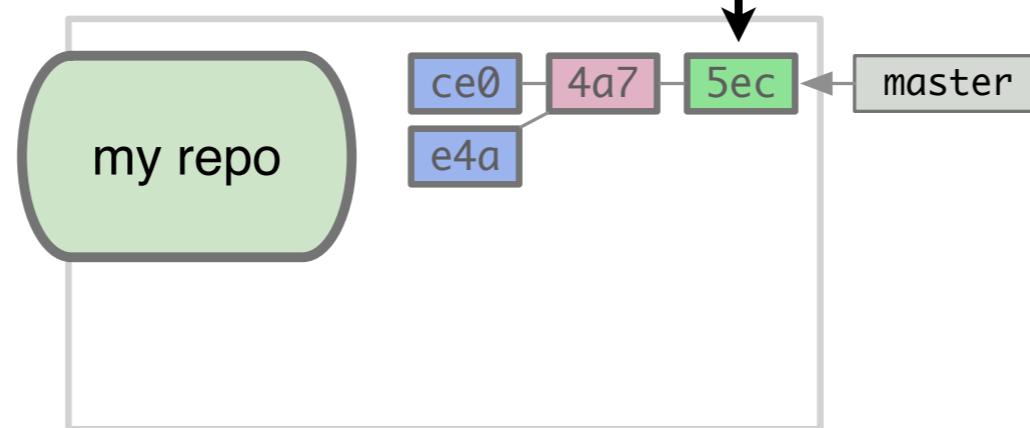
developer
jessica



developer
nick

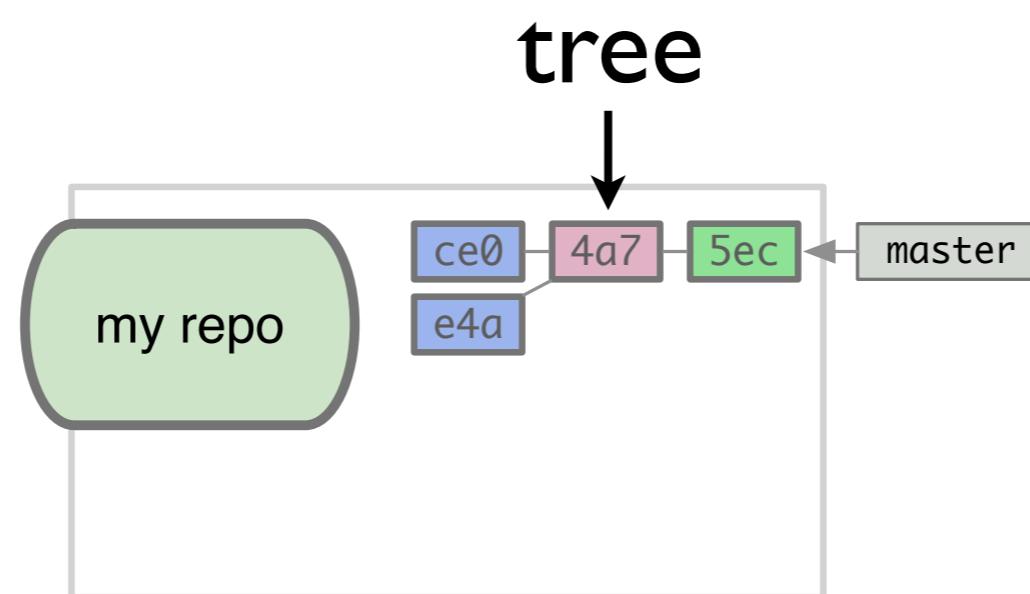
developer
jessica

commit



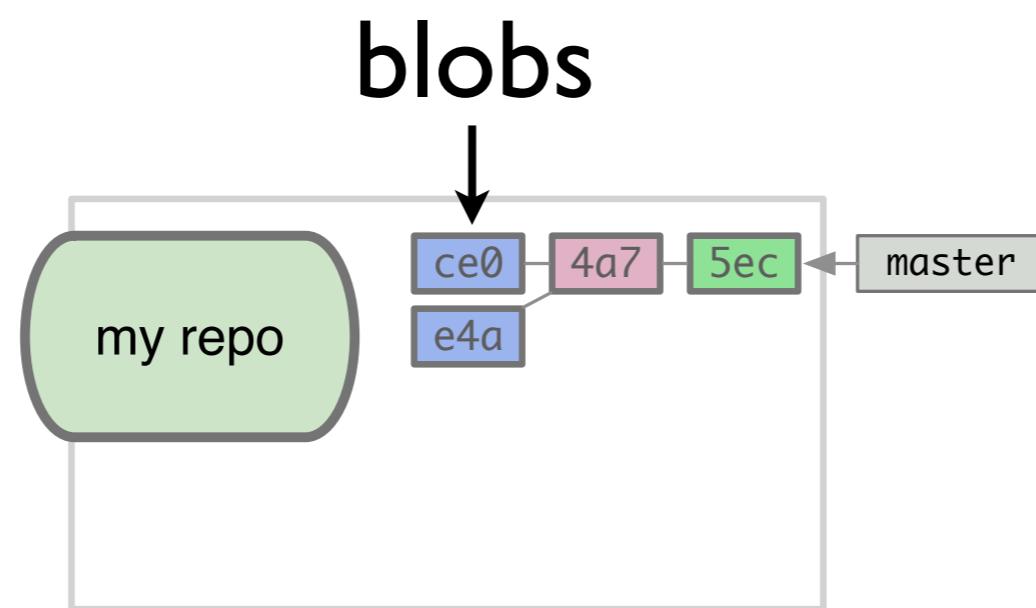
developer
nick

developer
jessica



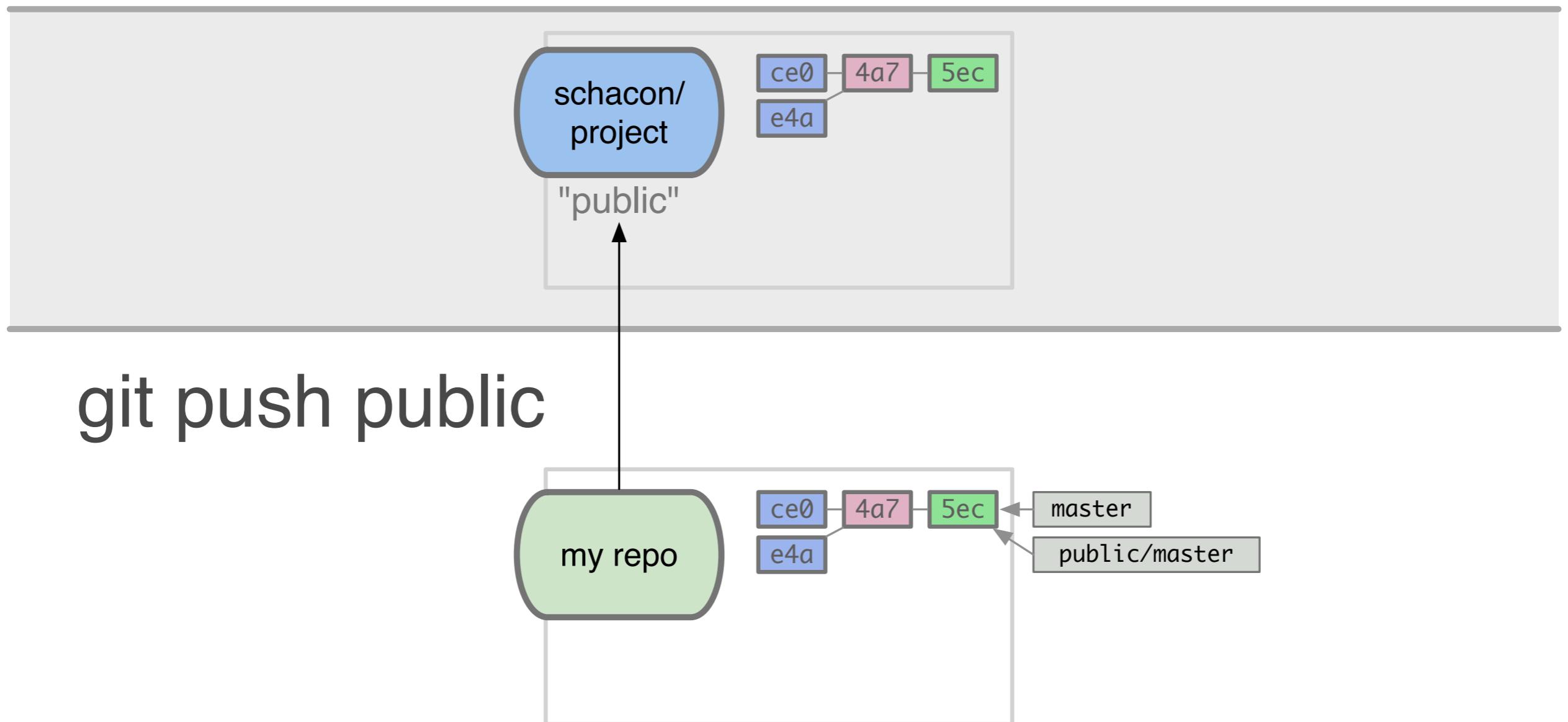
developer
nick

developer
jessica

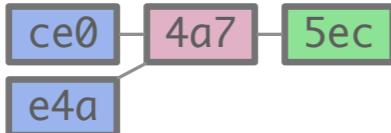


developer
nick

developer
jessica



developer
nick

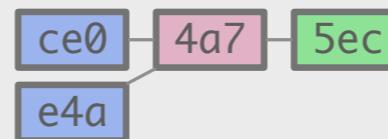


developer
jessica

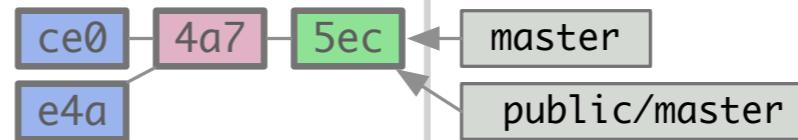
git clone (url)

schacon/
project

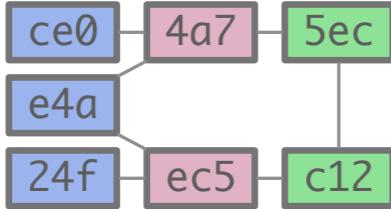
"public"



my repo



developer
nick

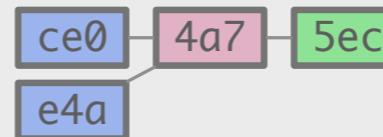


developer
jessica

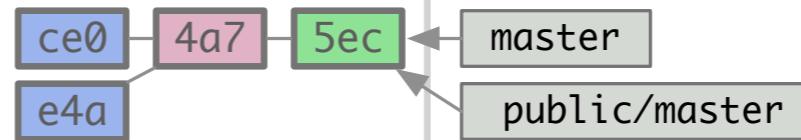
git commit

schacon/
project

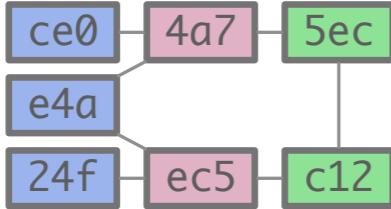
"public"



my repo

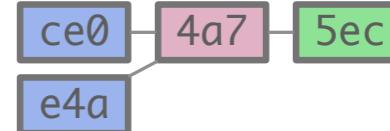


developer
nick



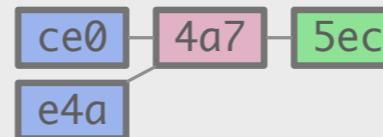
git clone (url)

developer
jessica

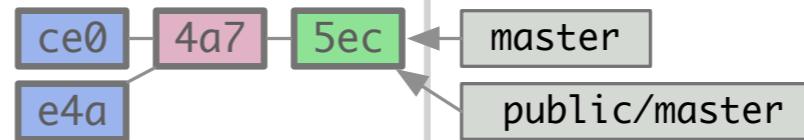


schacon/
project

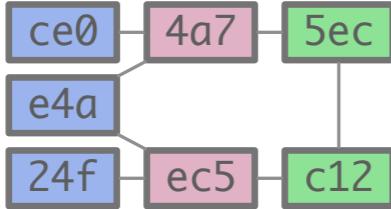
"public"



my repo

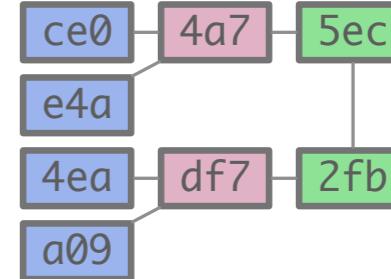


developer
nick



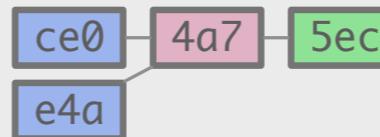
git commit

developer
jessica

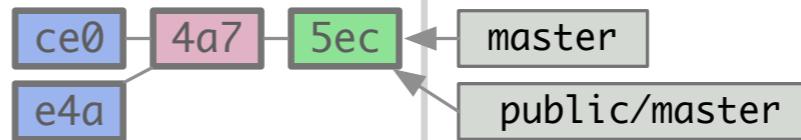


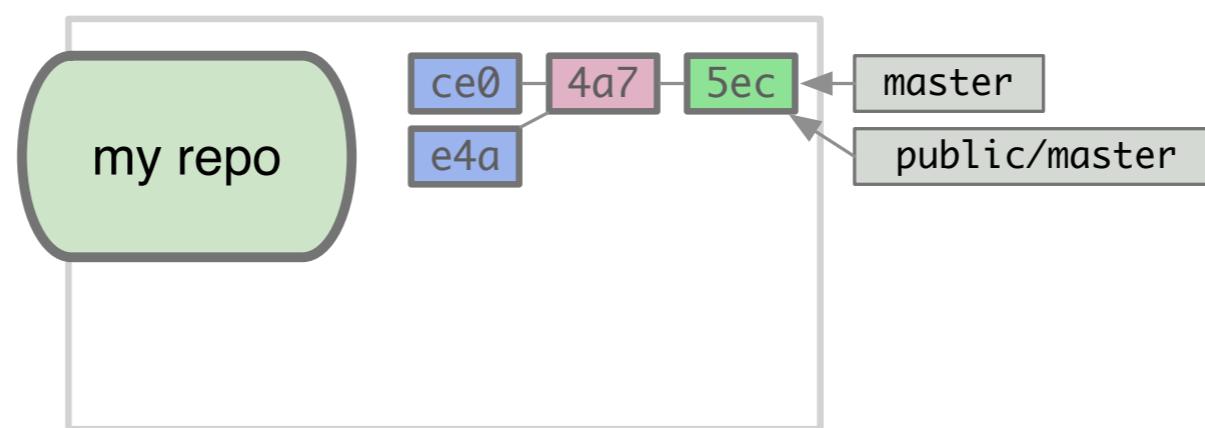
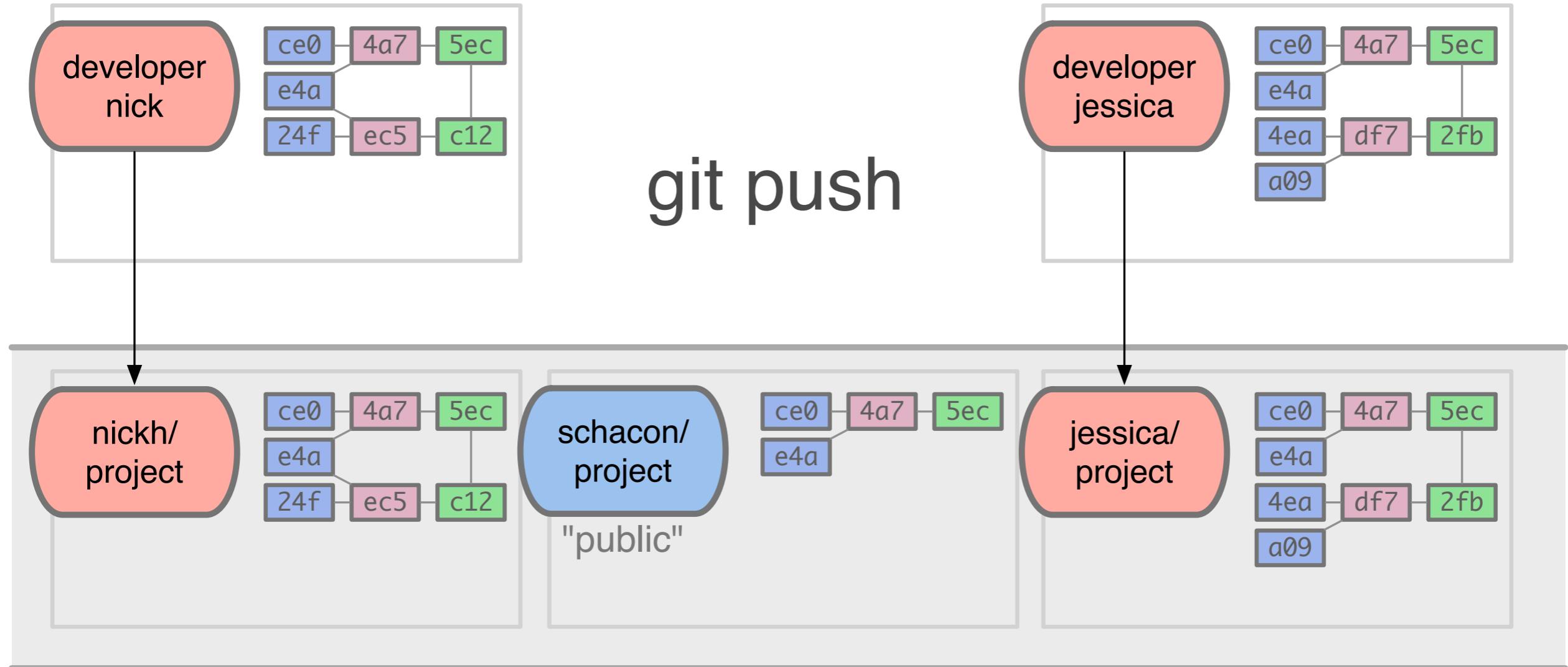
schacon/
project

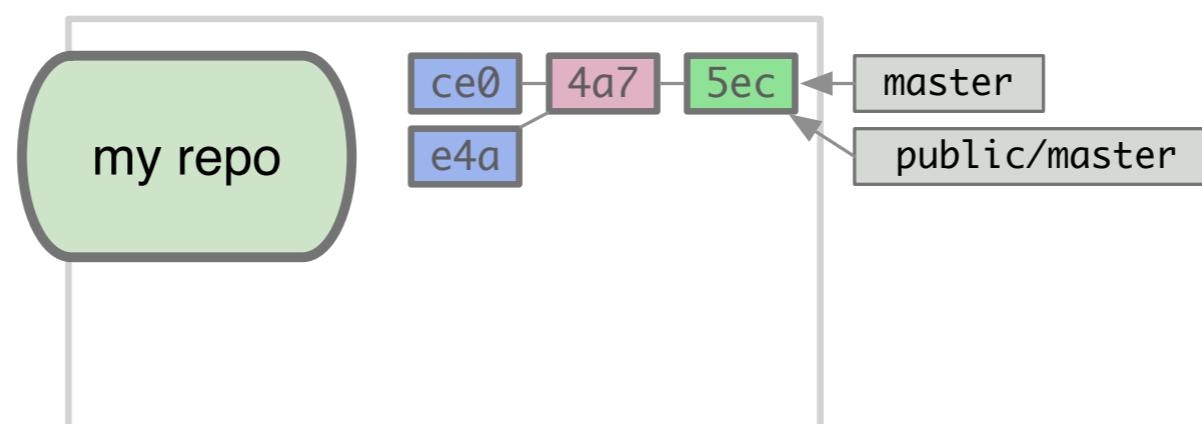
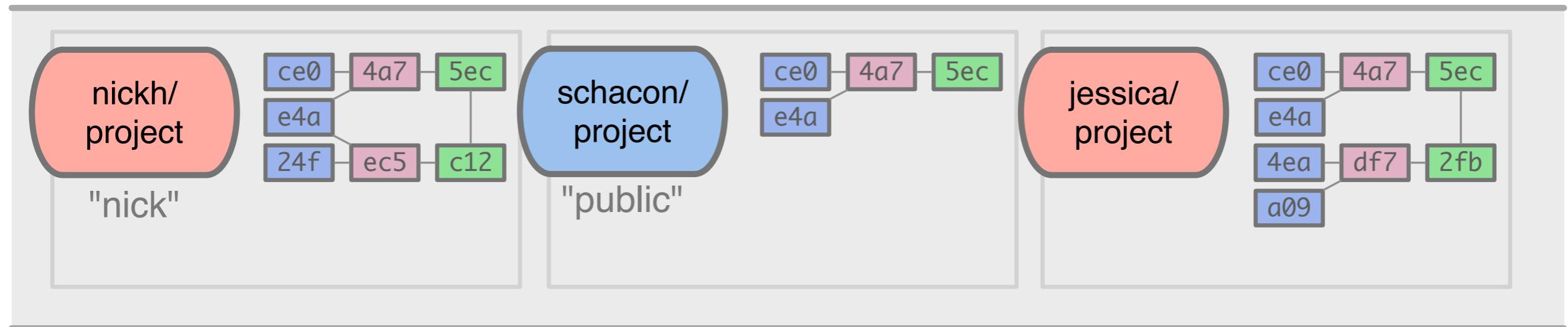
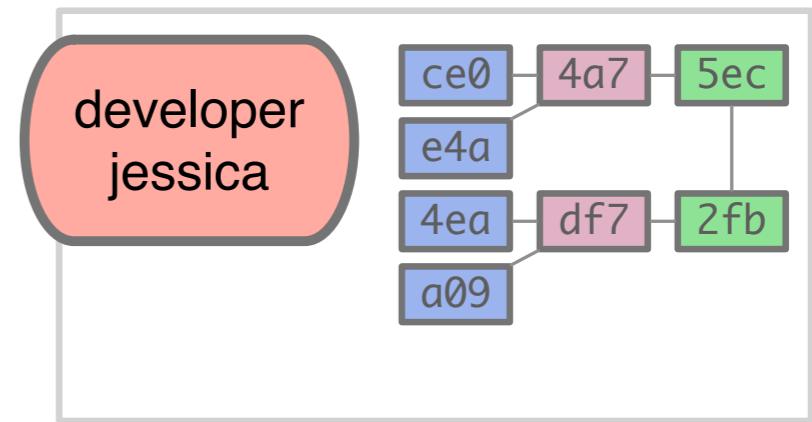
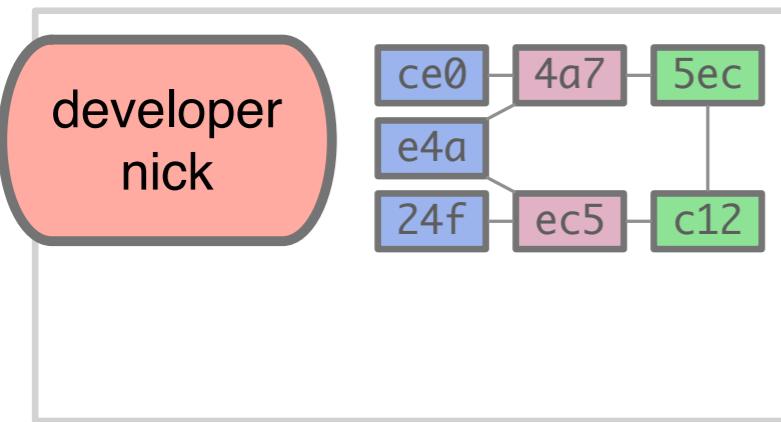
"public"



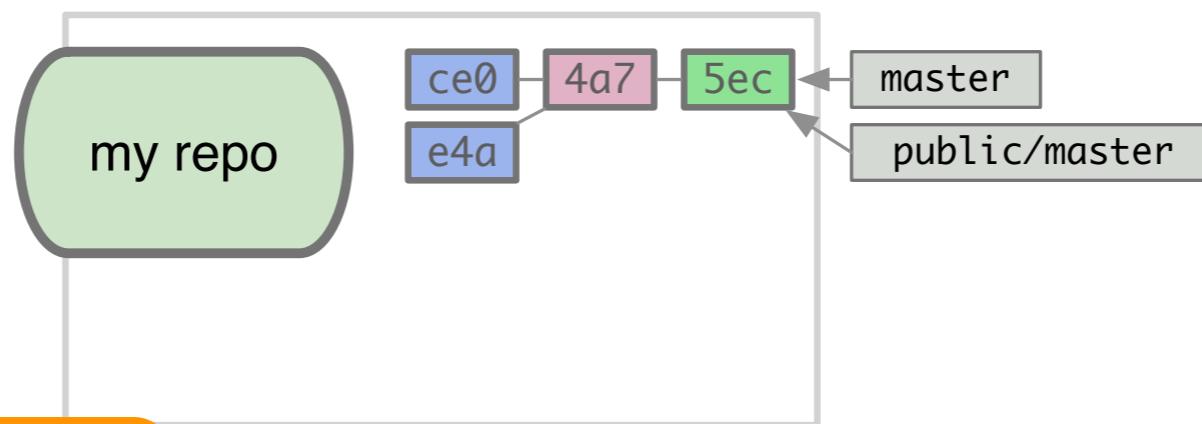
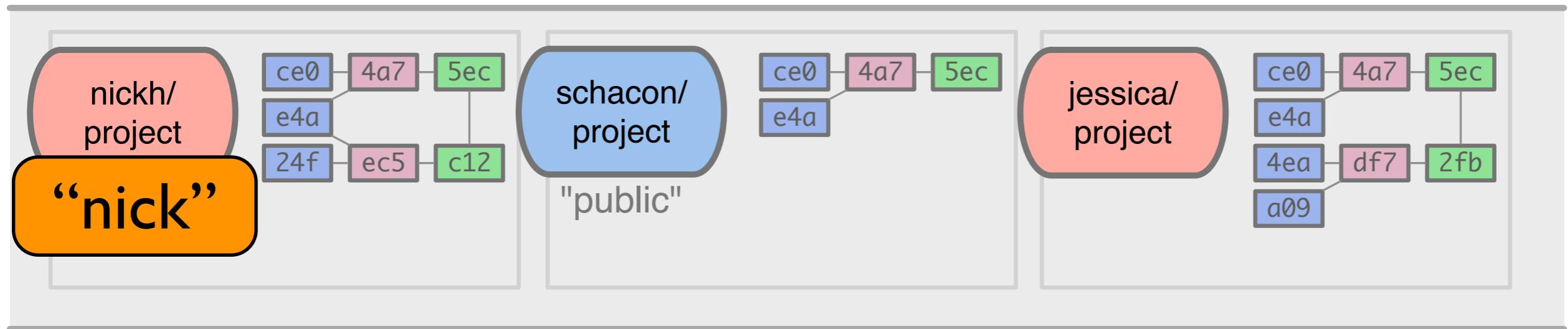
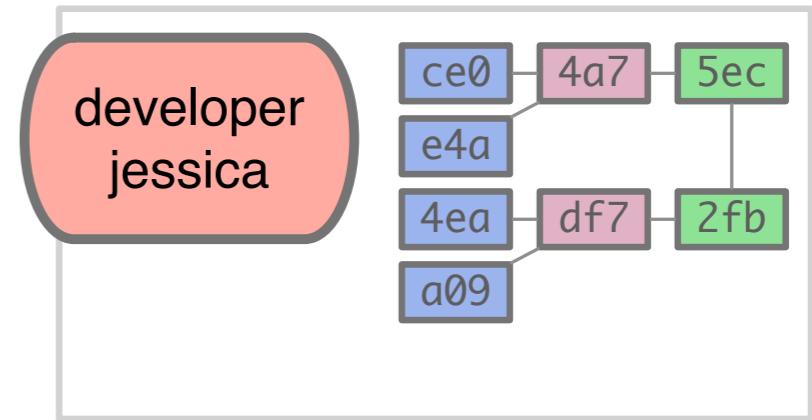
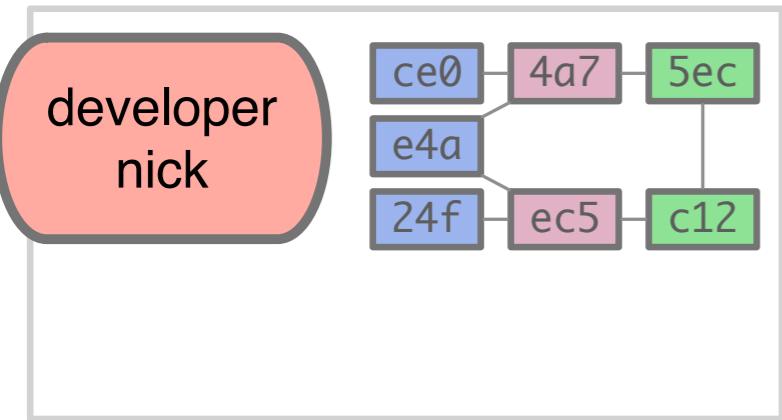
my repo



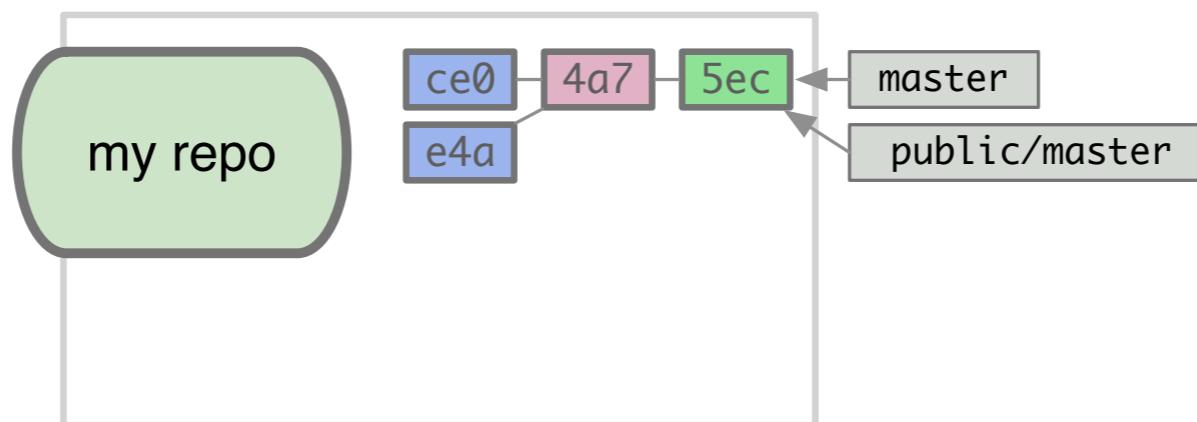
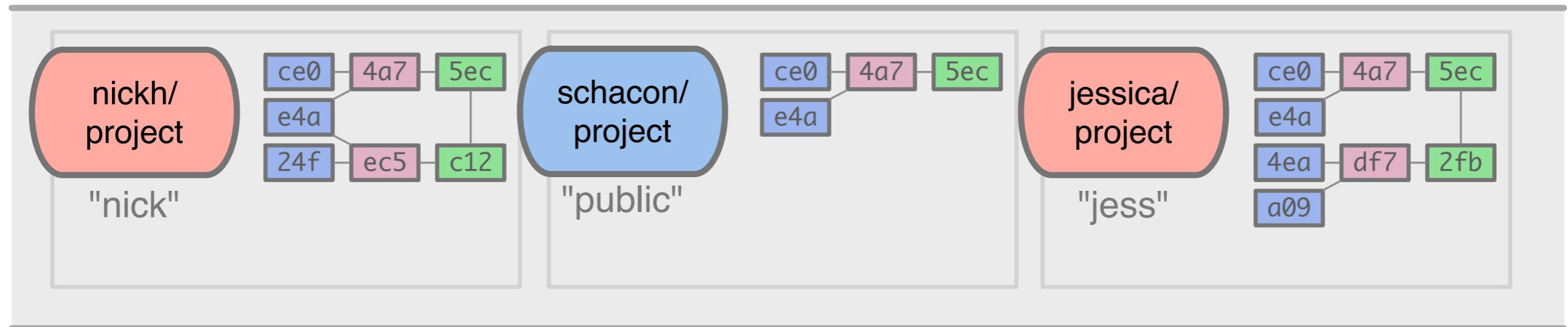
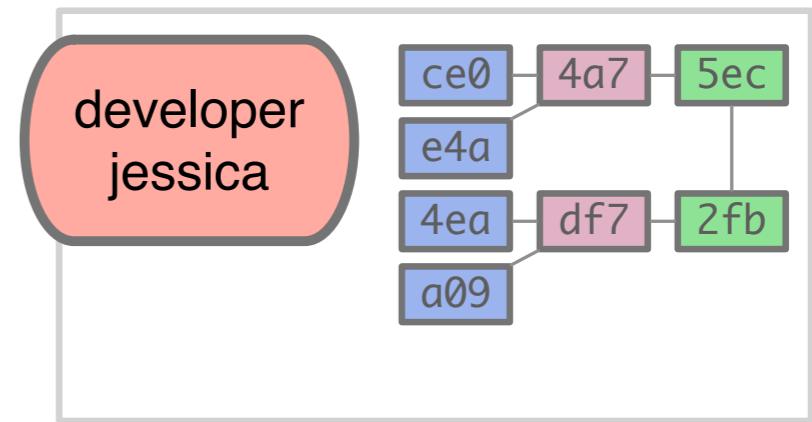
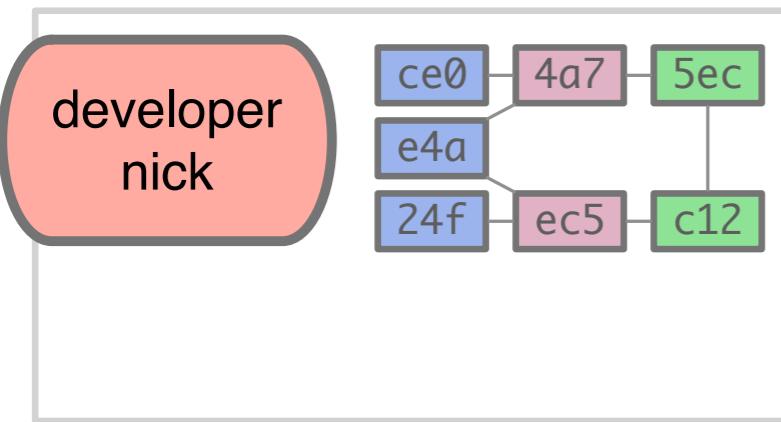




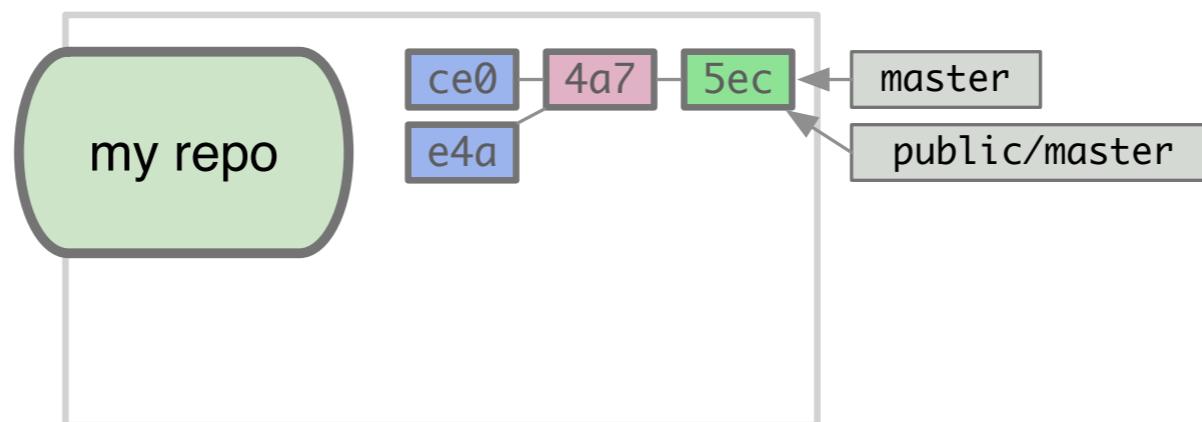
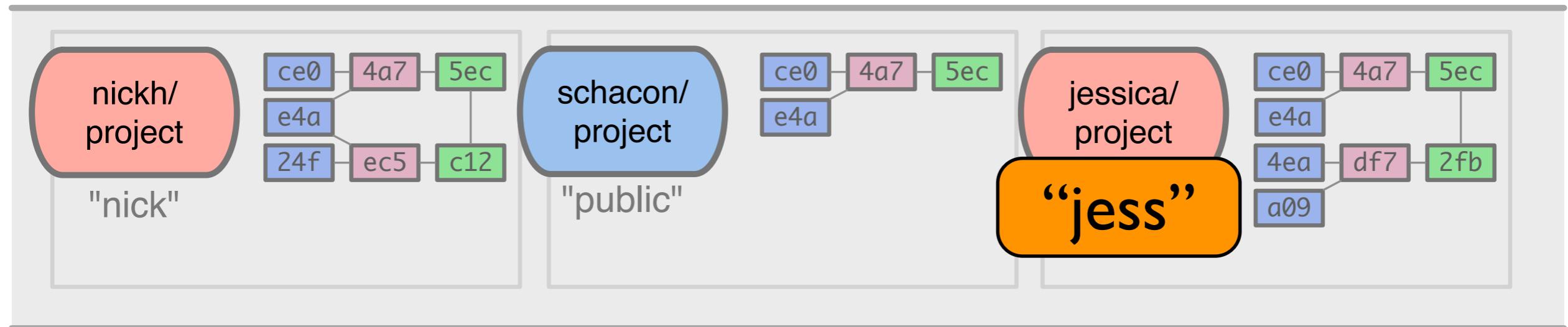
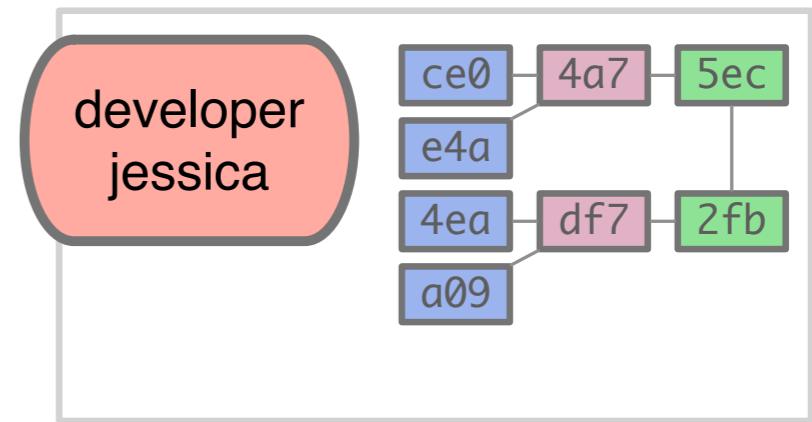
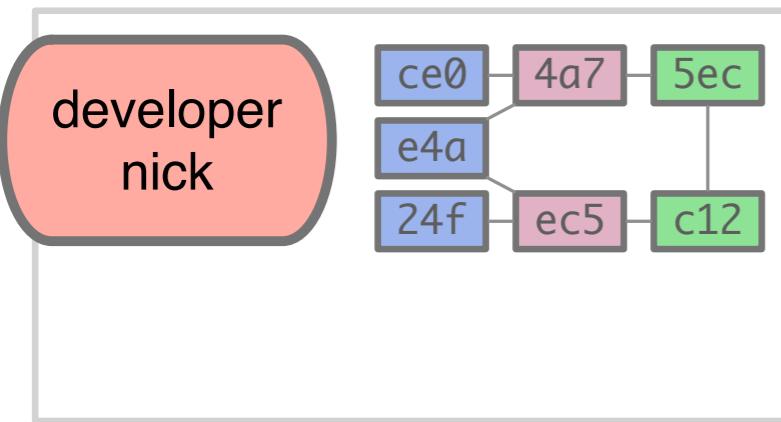
`git remote add nick git://github.com/nickh/project.git`



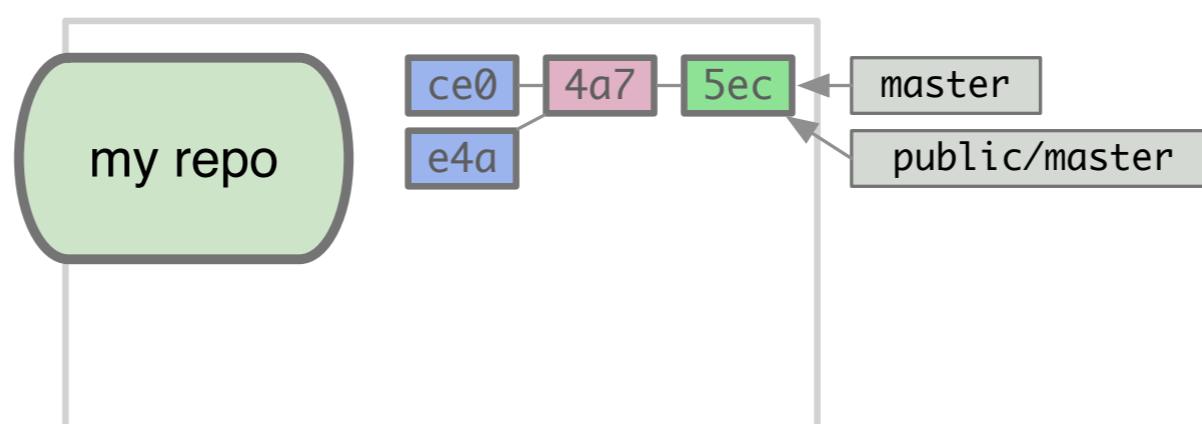
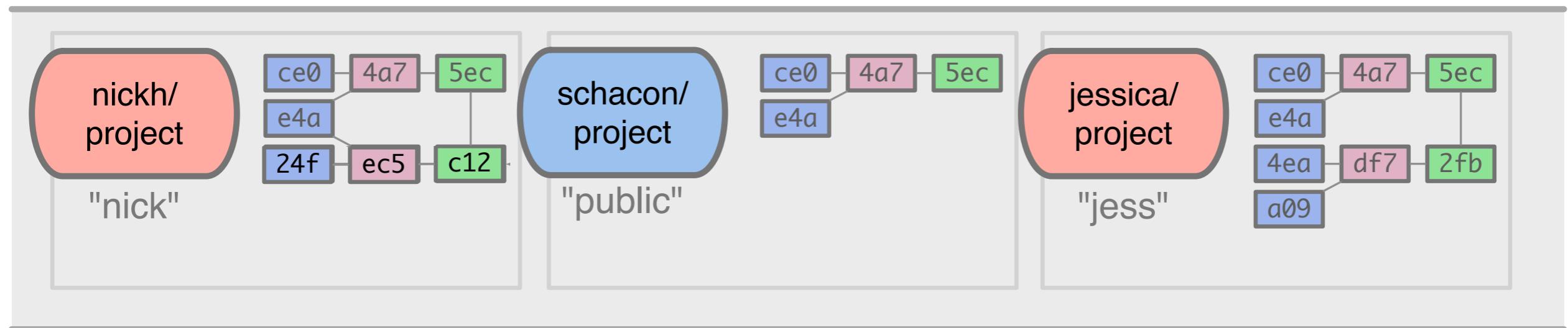
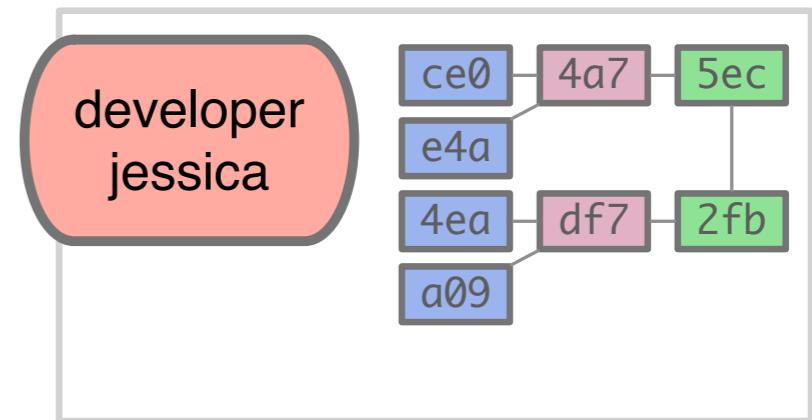
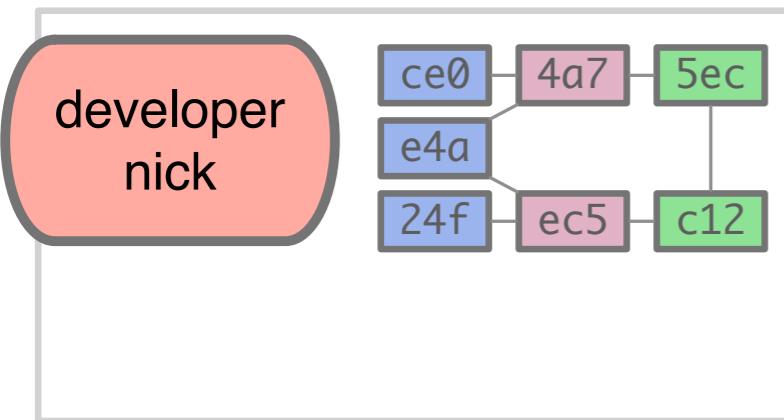
`git remote add nick git://github.com/nickh/project.git`



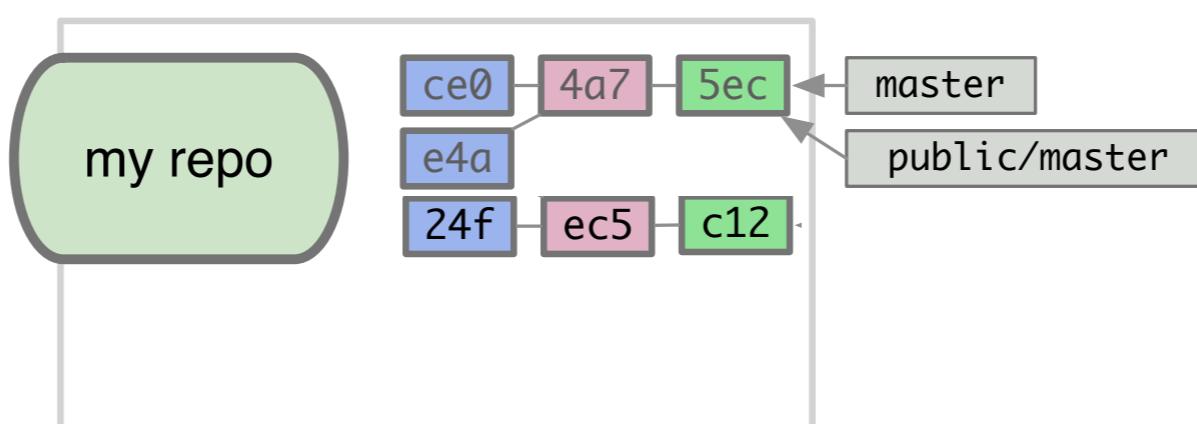
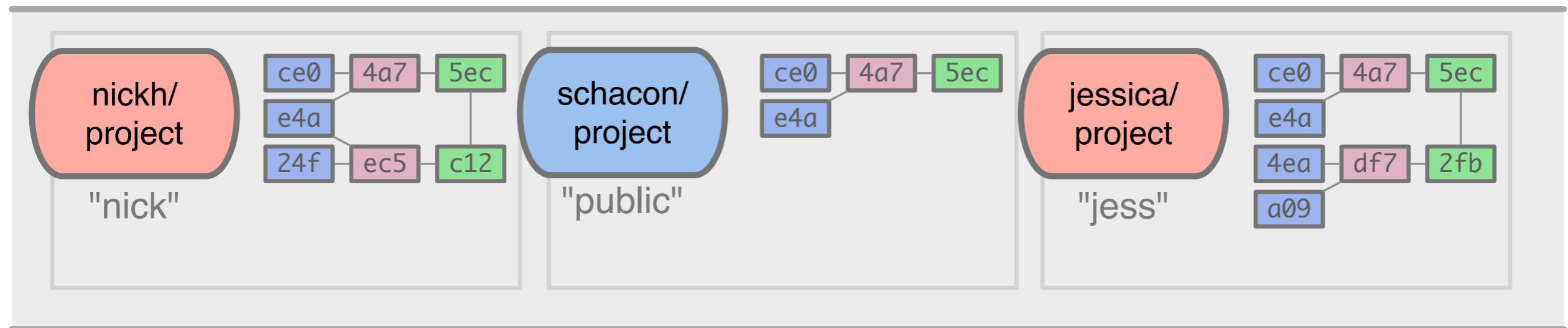
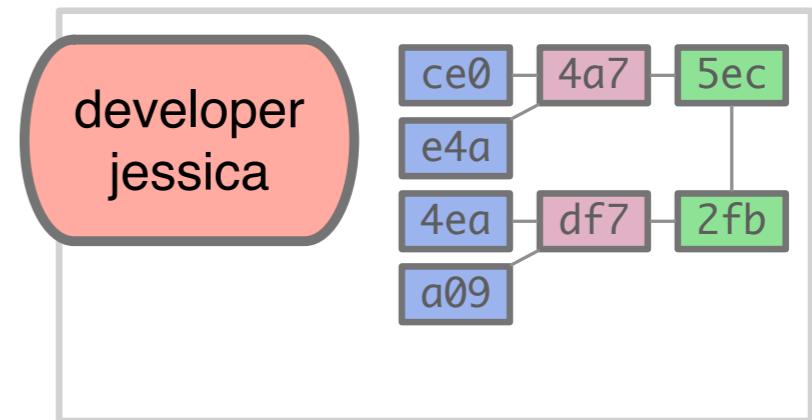
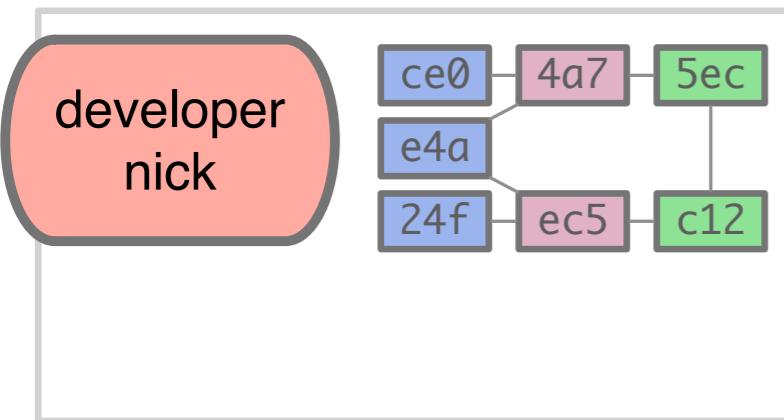
`git remote add jess git://github.com/jessica/project.git`



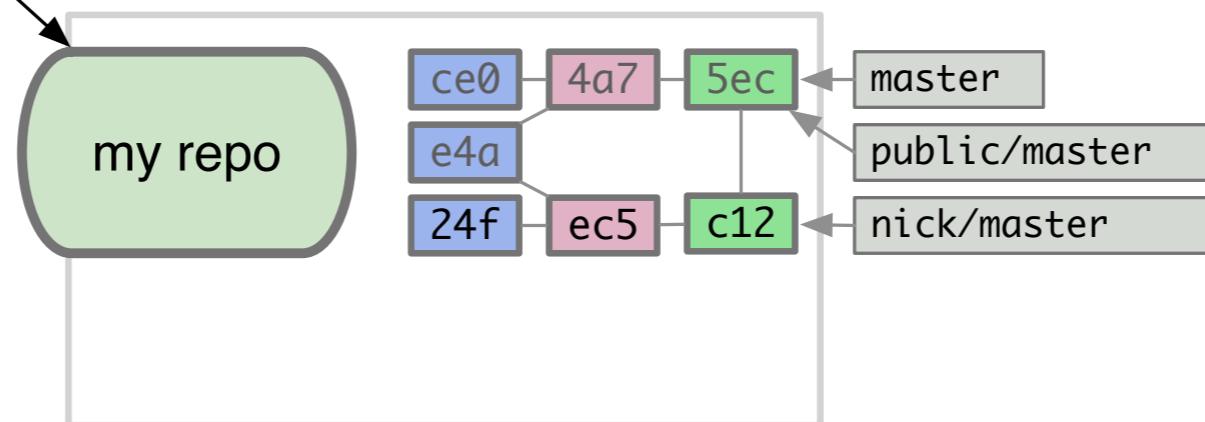
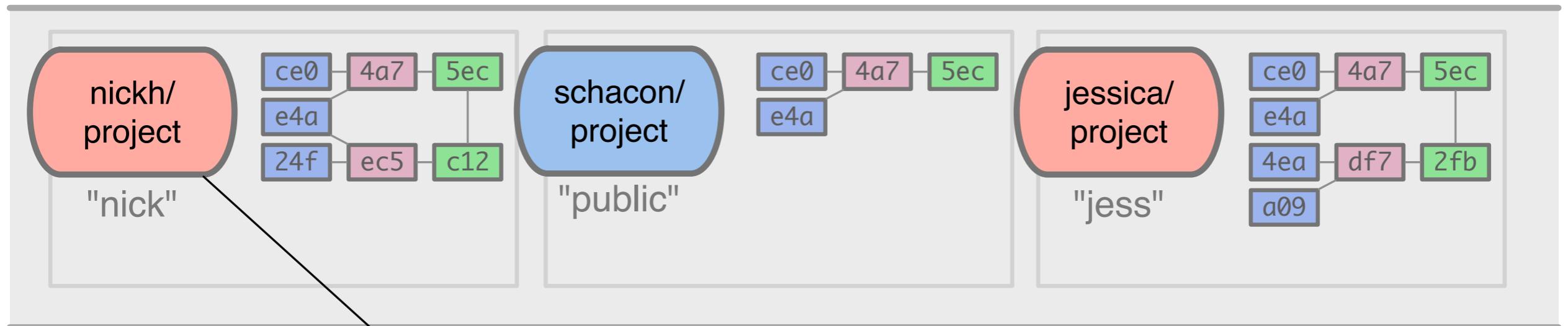
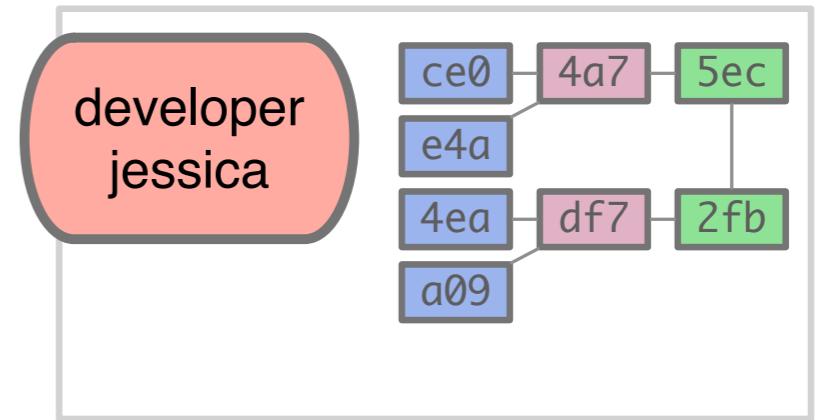
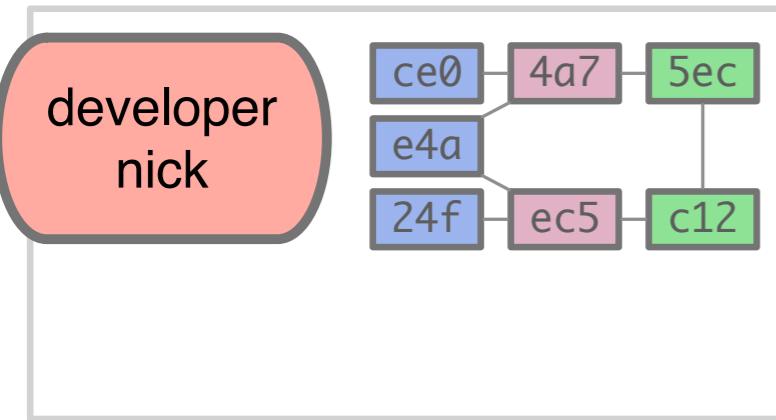
`git remote add jess git://github.com/jessica/project.git`

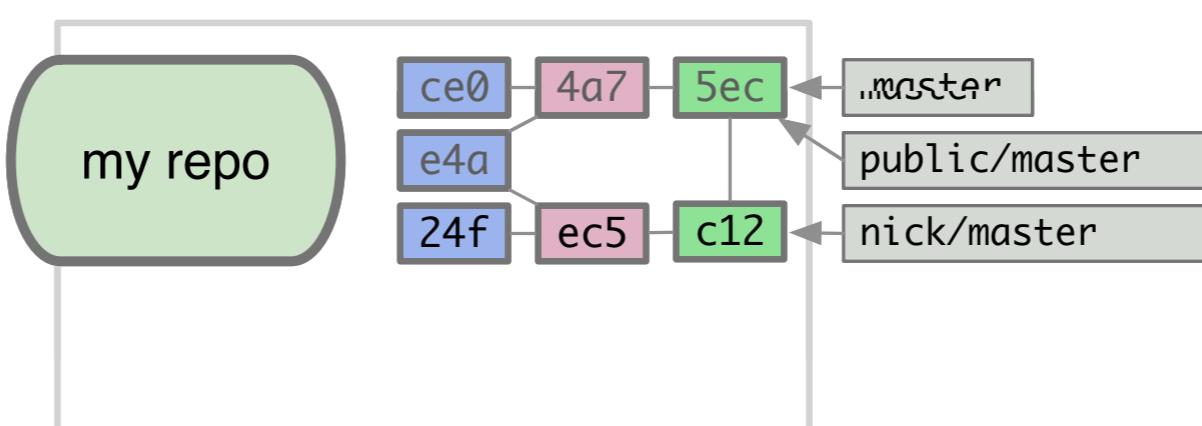
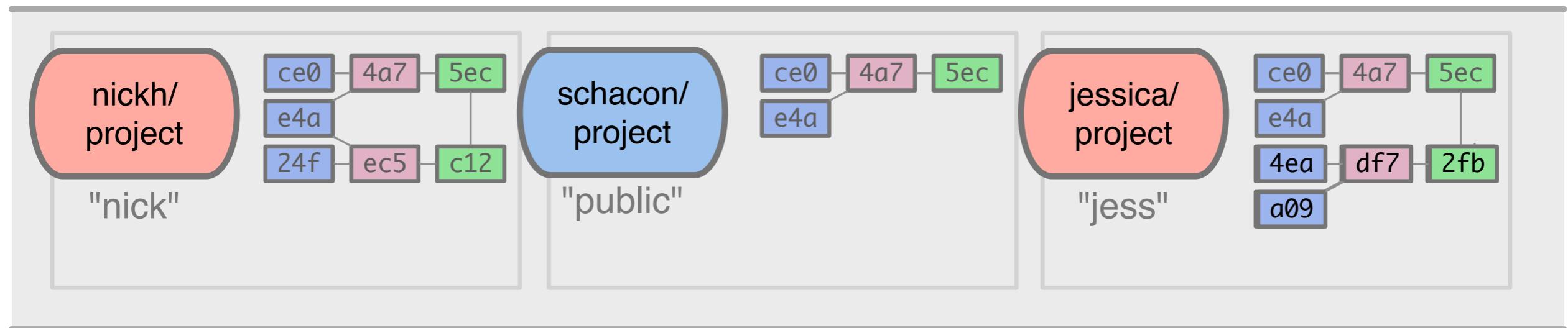
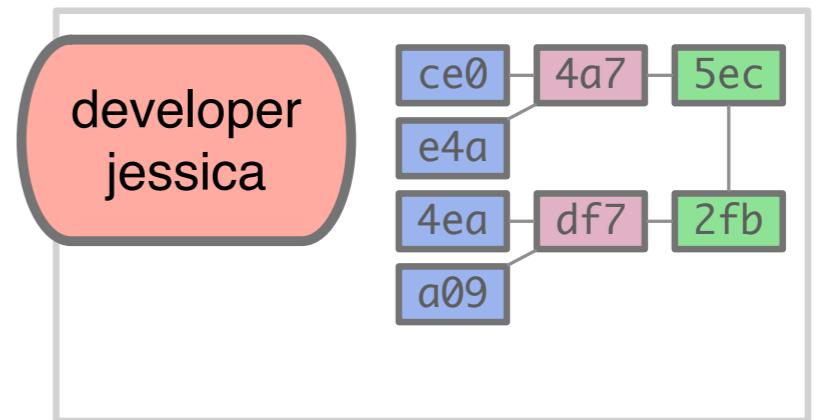
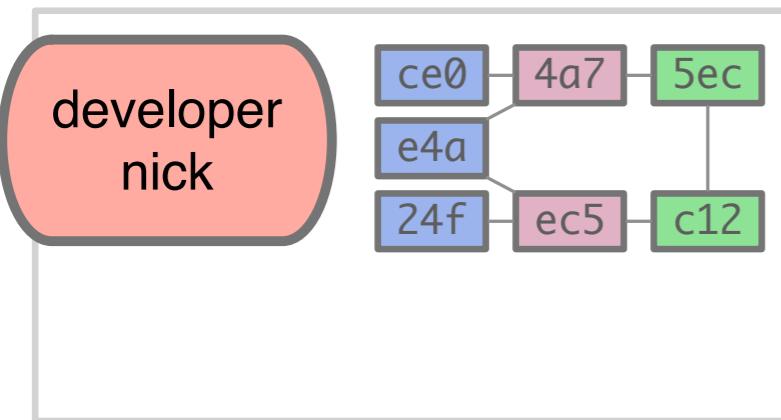


git fetch nick

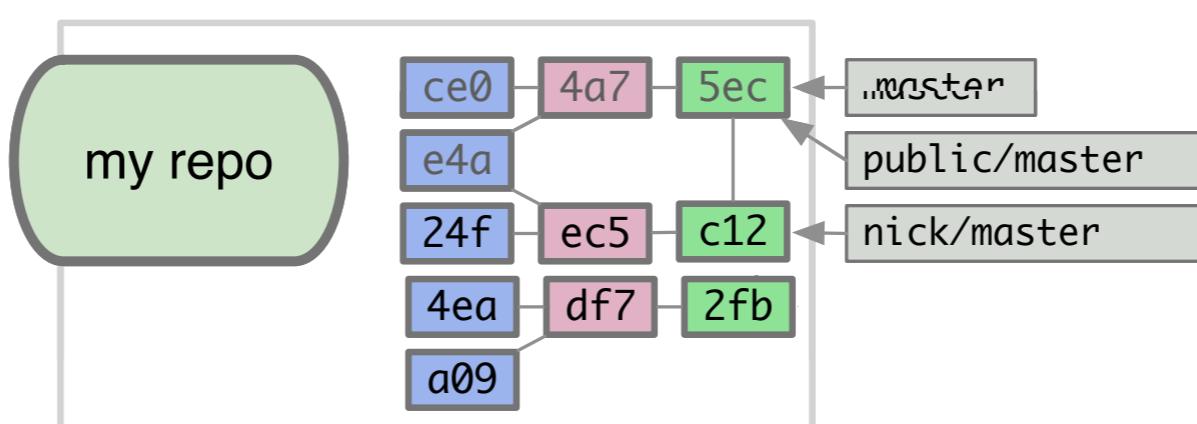
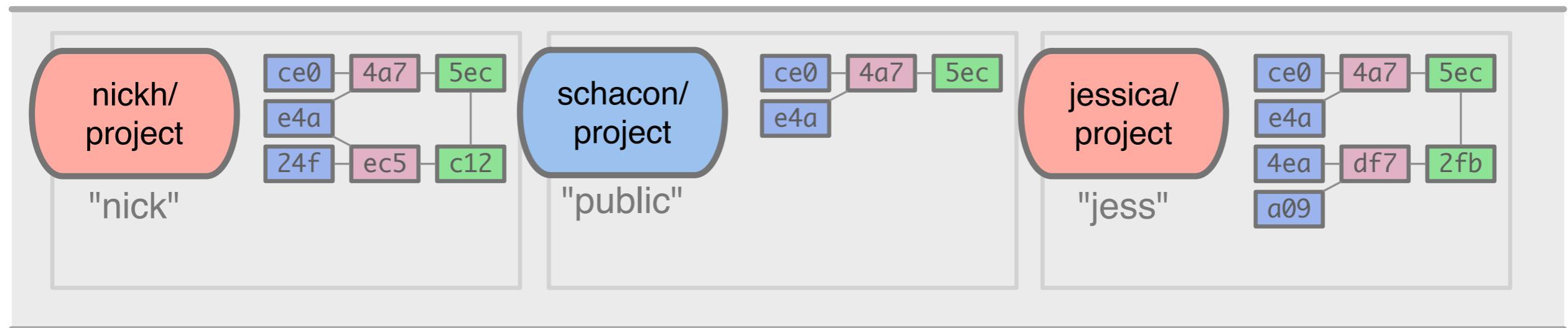
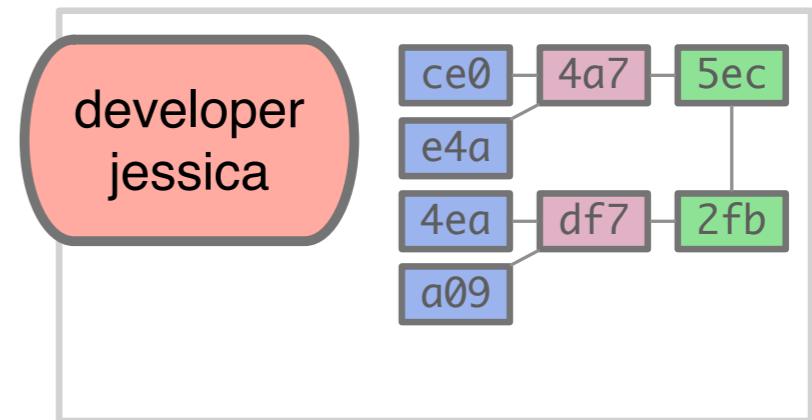
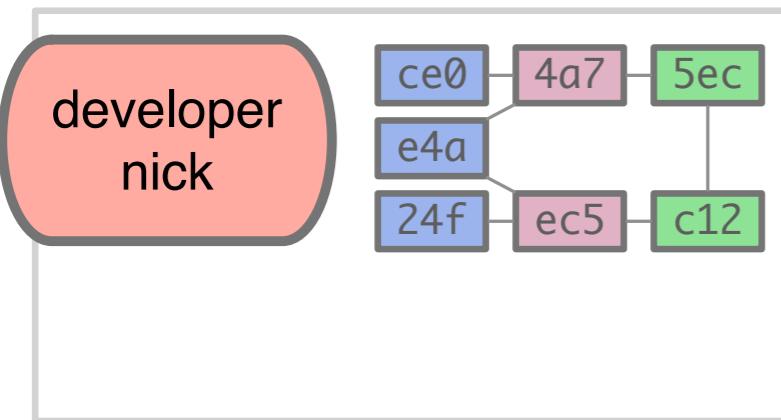


git fetch nick

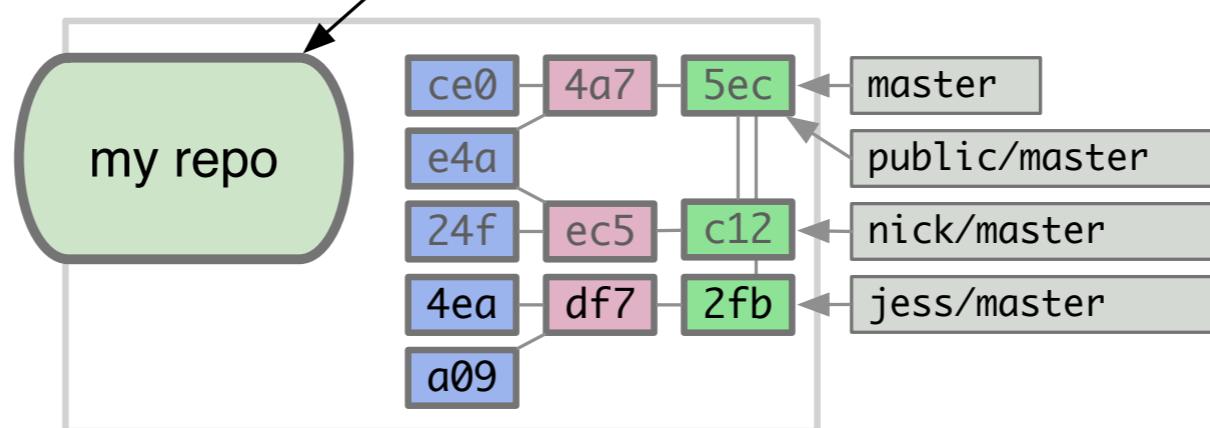
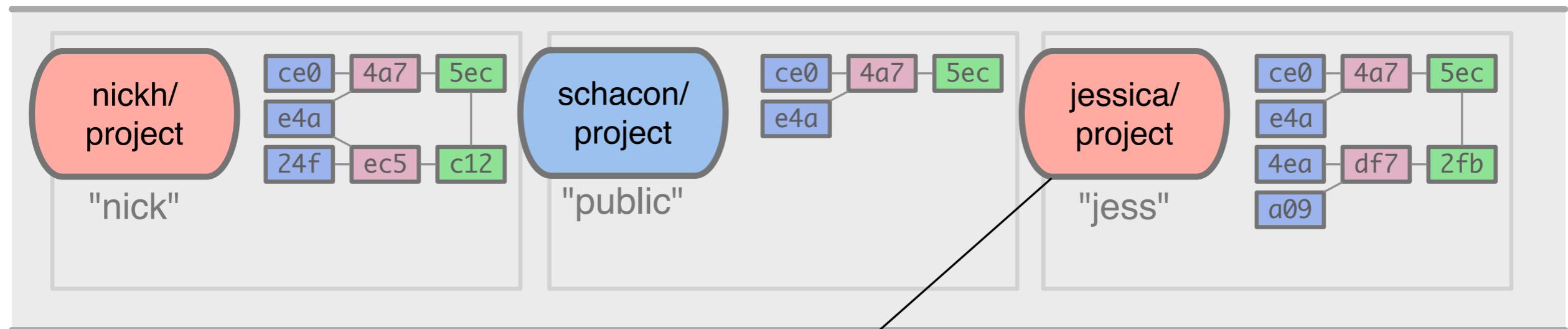
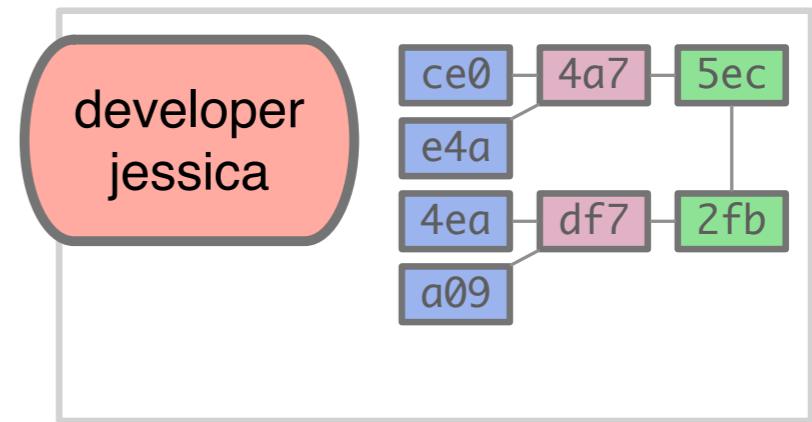
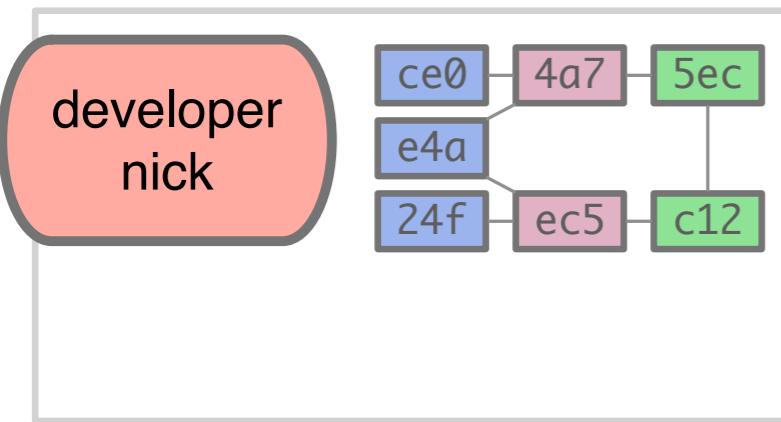


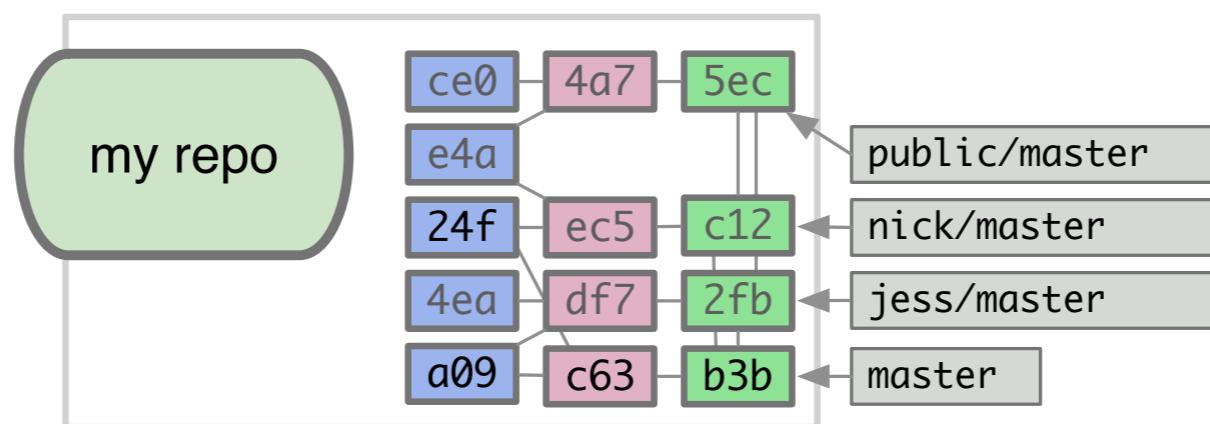
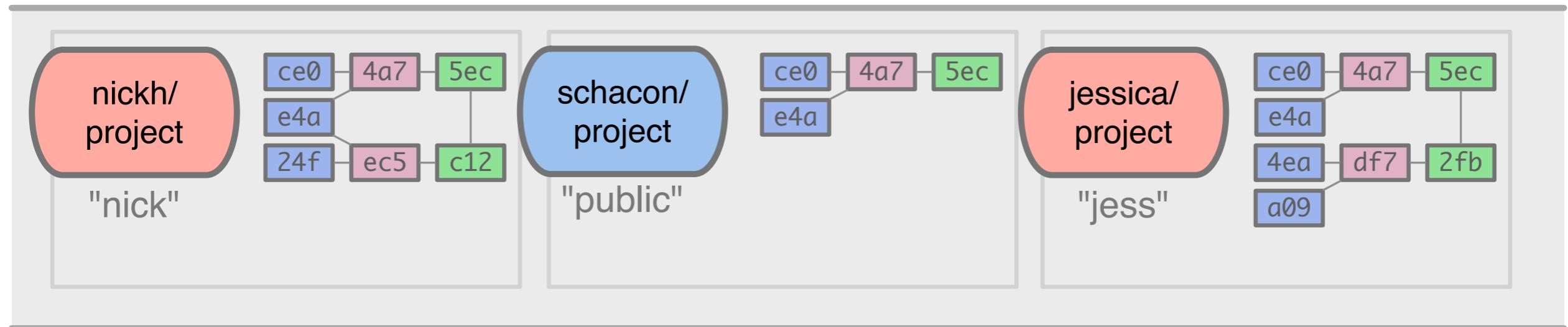
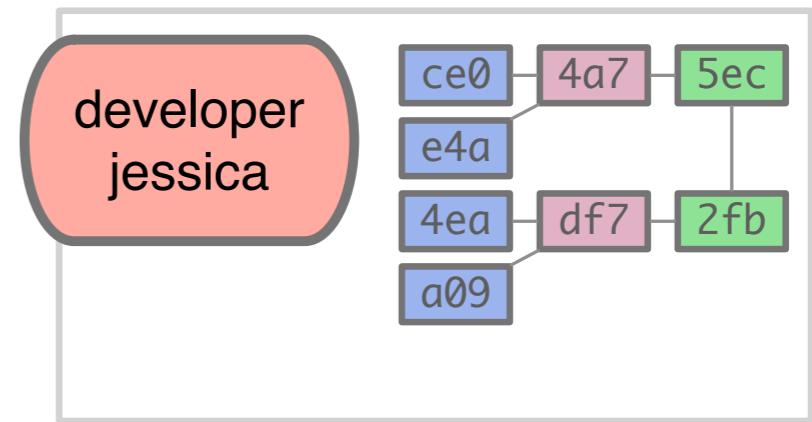
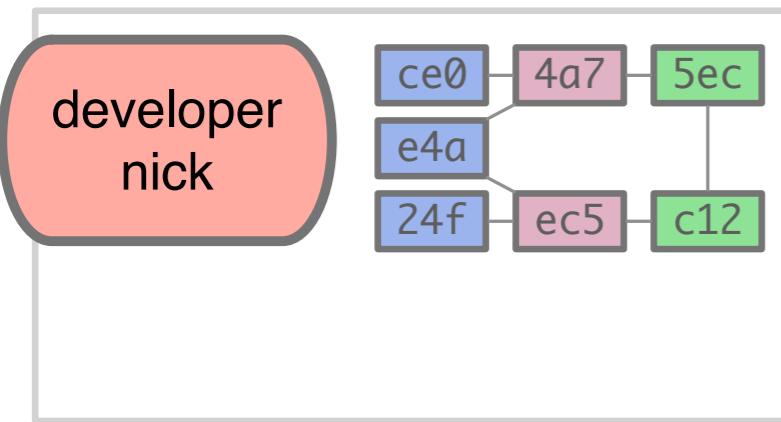


git fetch jess

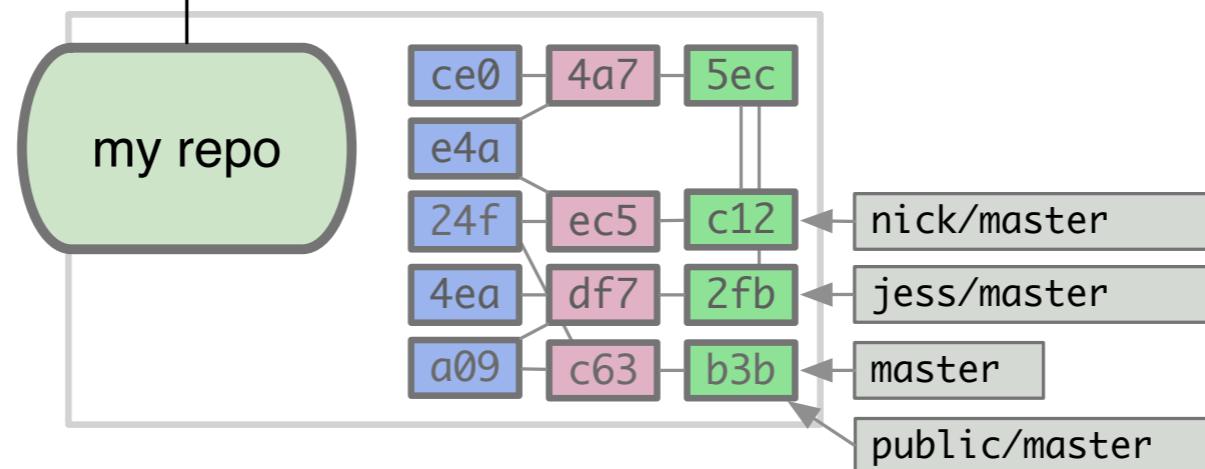
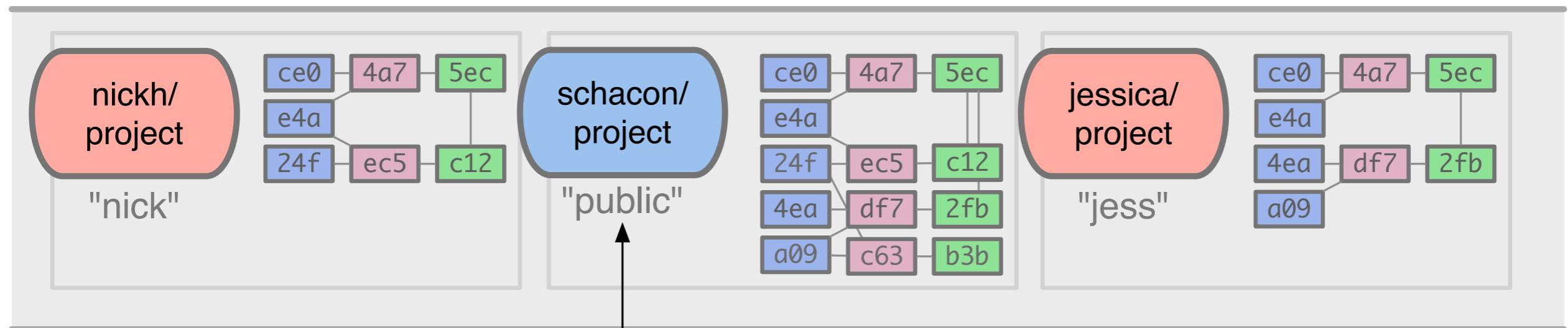
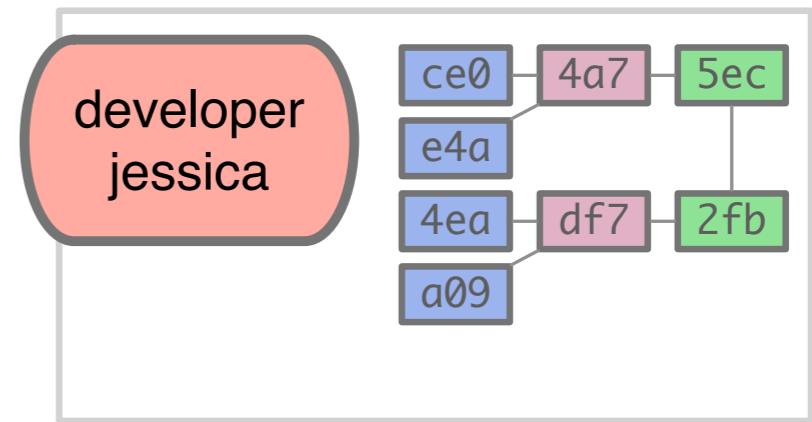
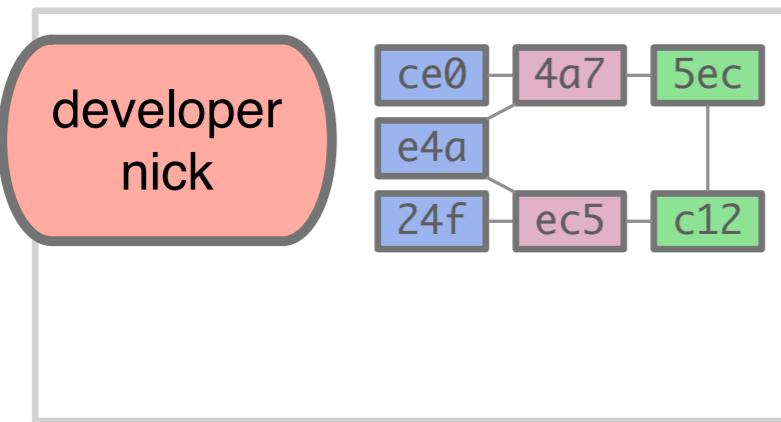


git fetch jess

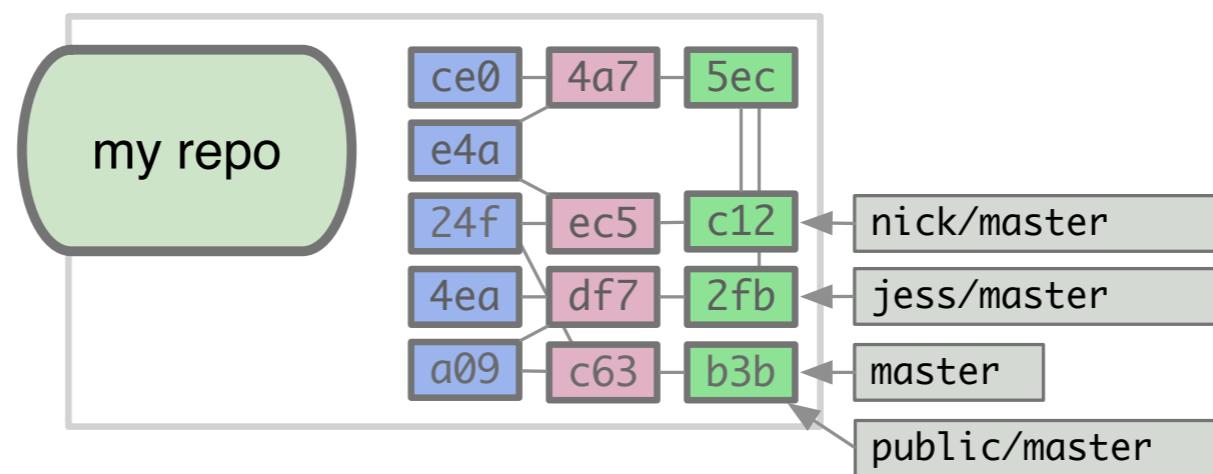
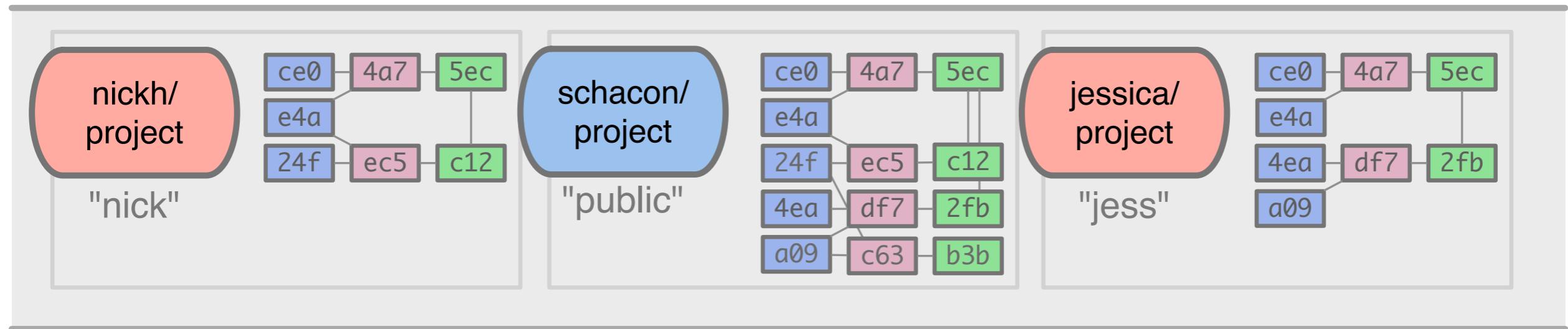
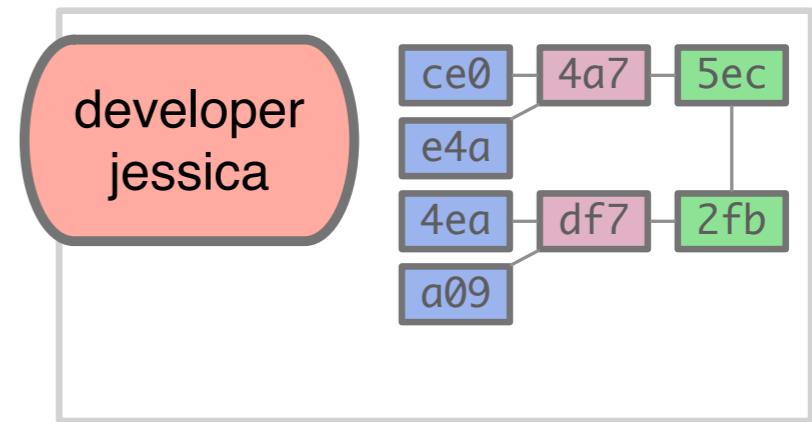
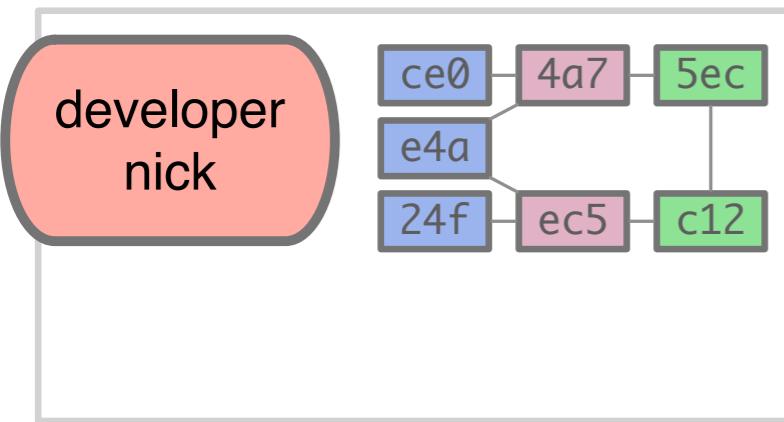


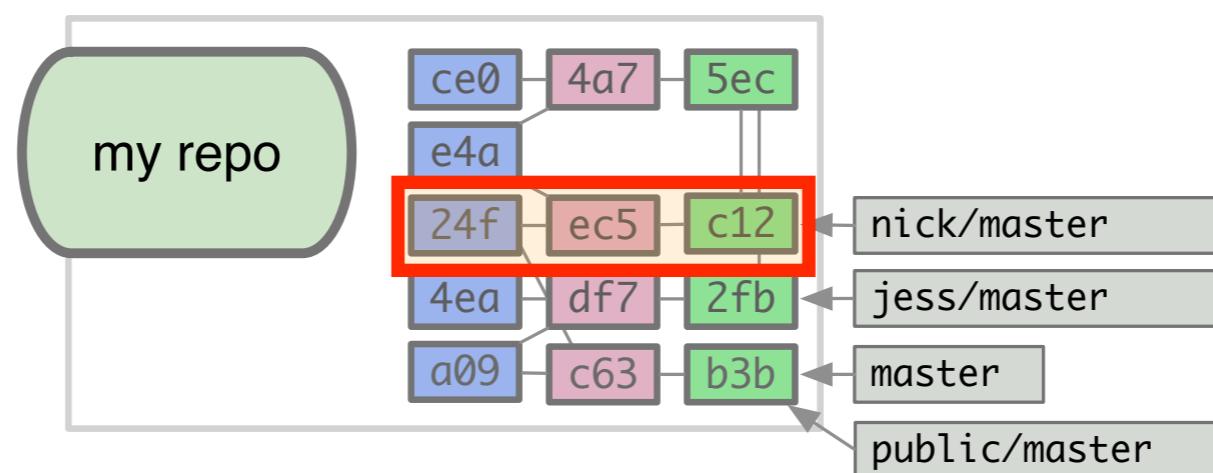
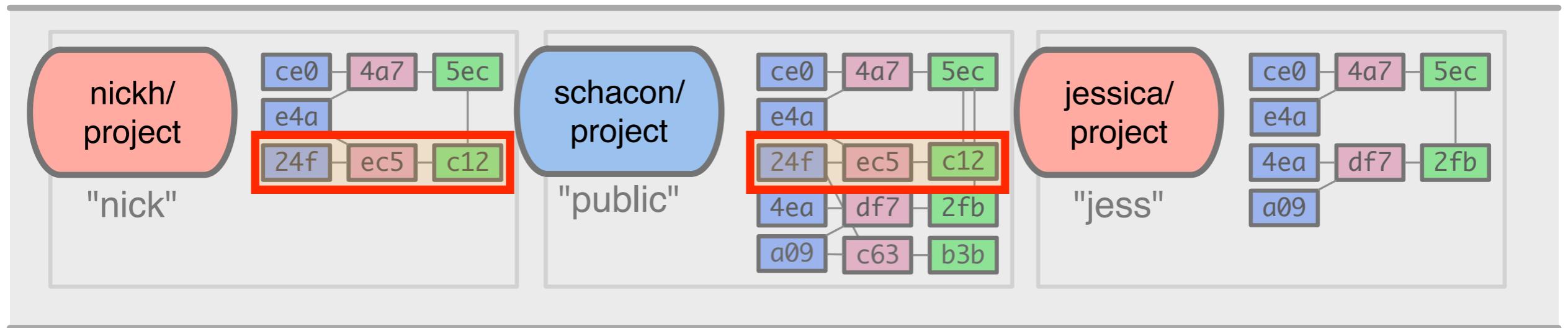
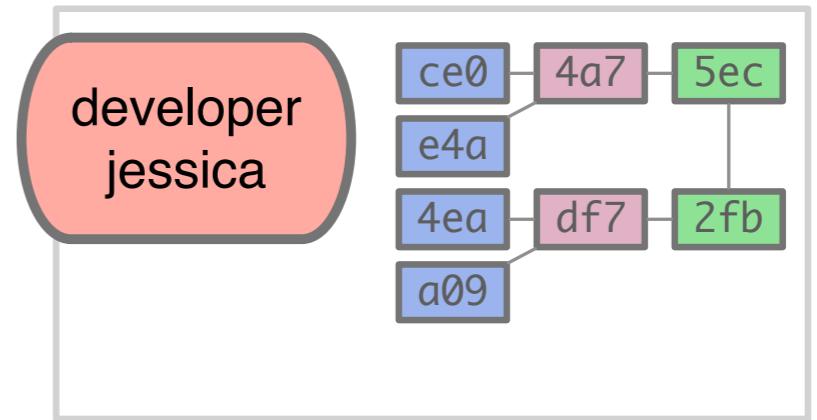
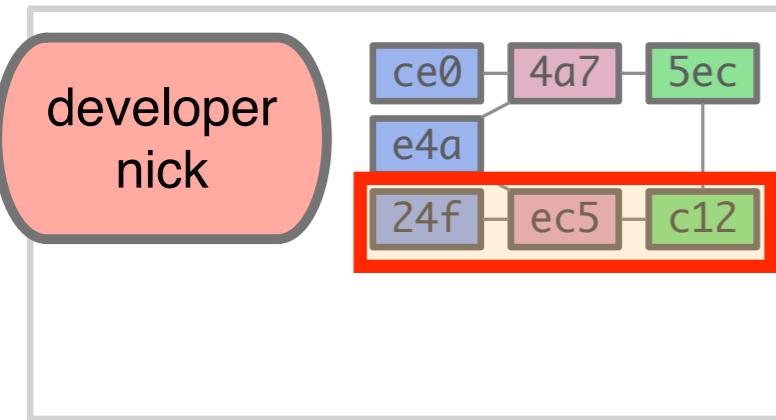


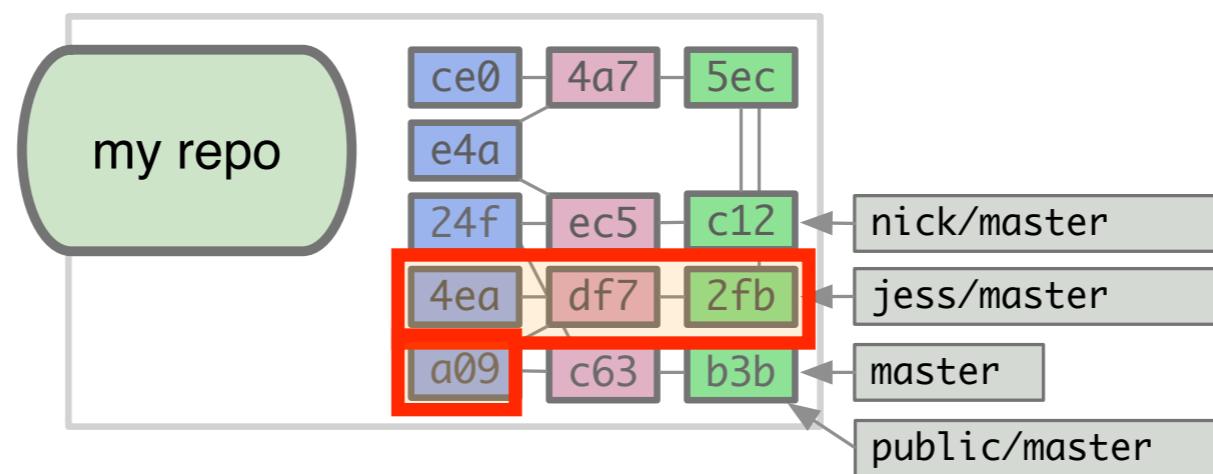
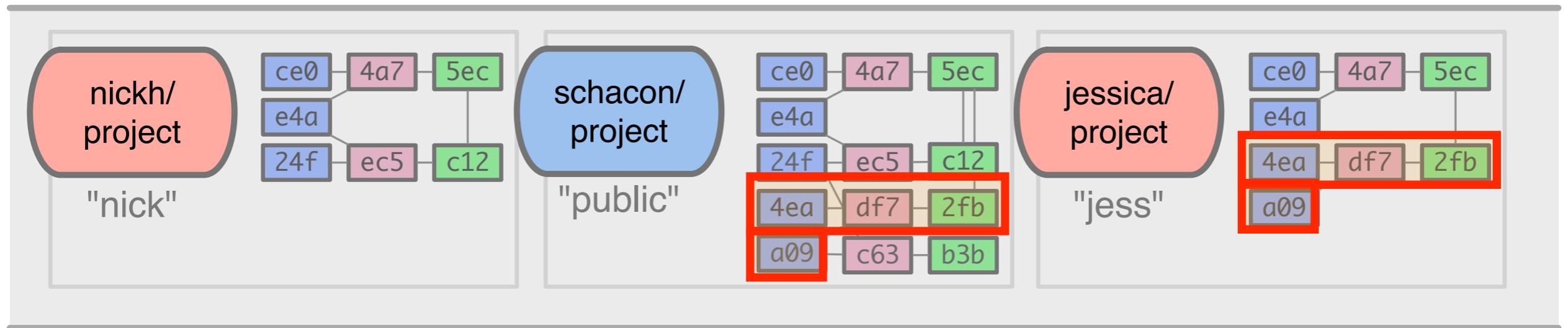
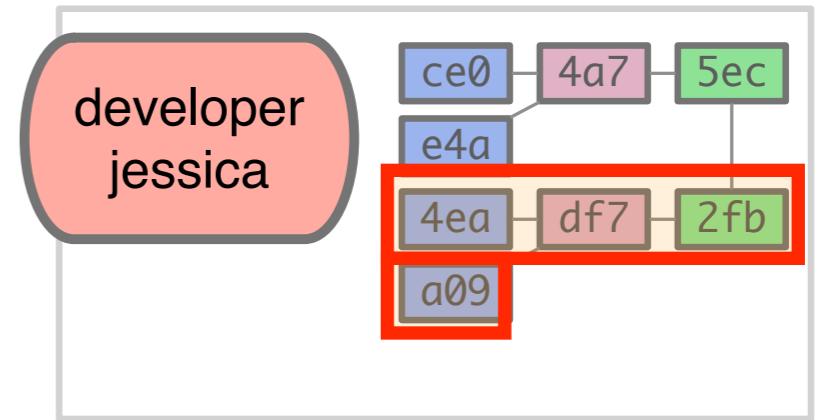
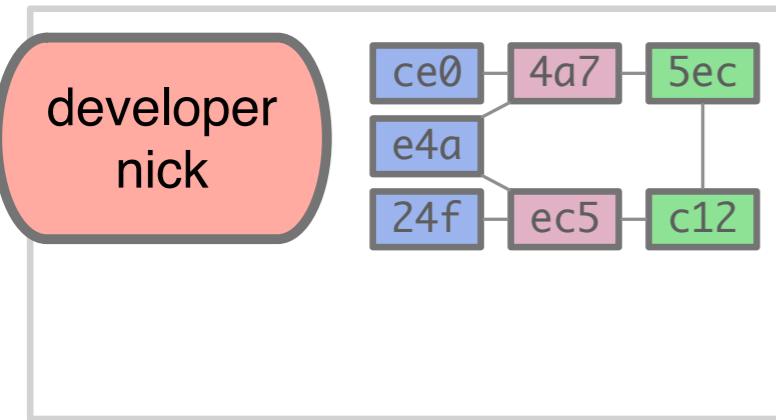
git merge nick jess

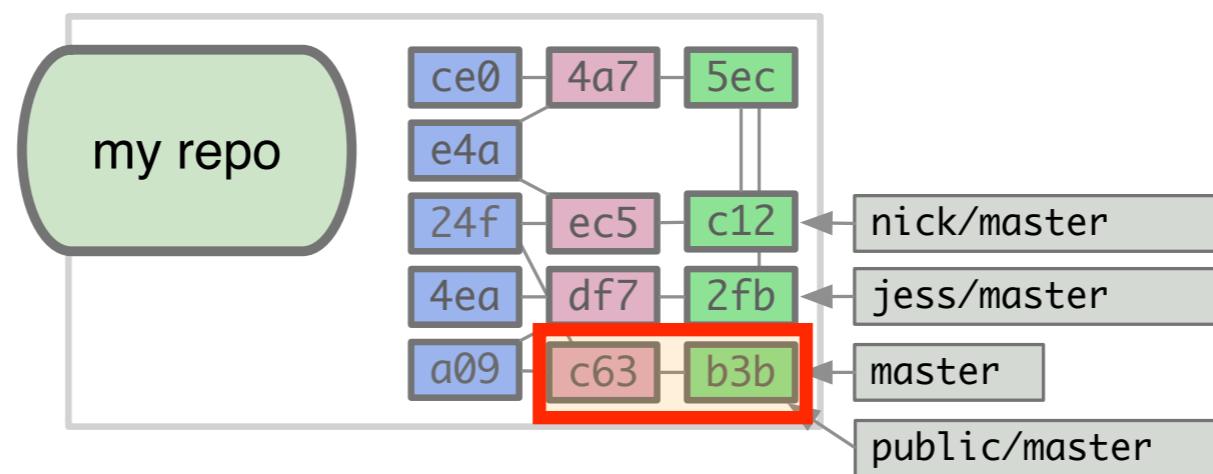
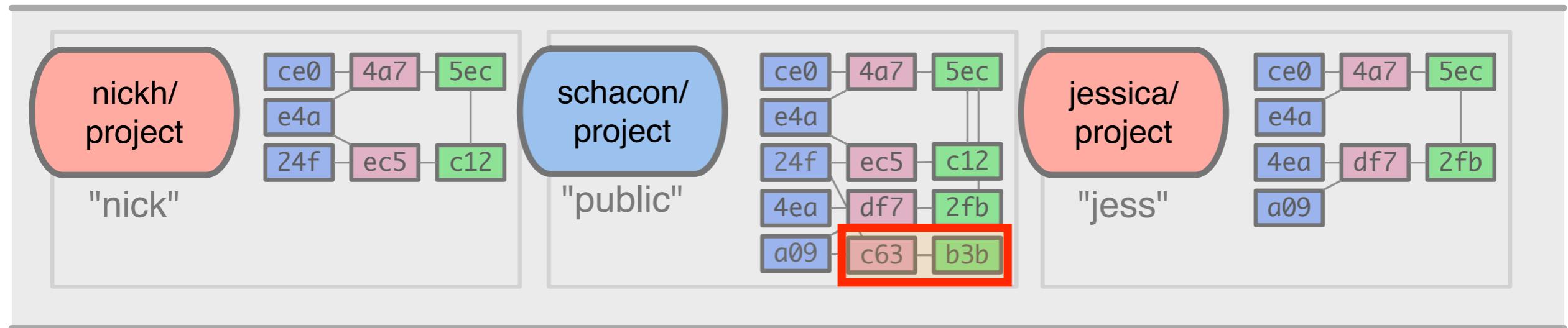
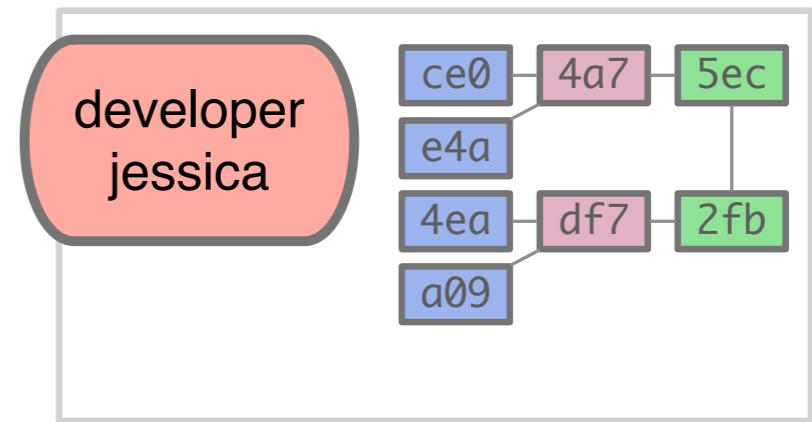
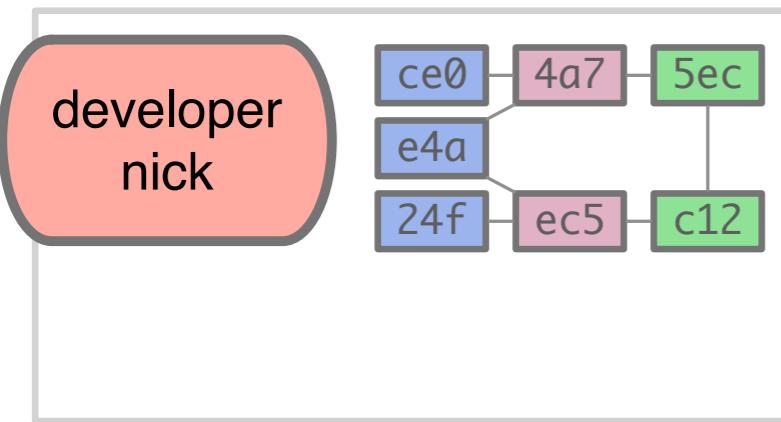


git push public





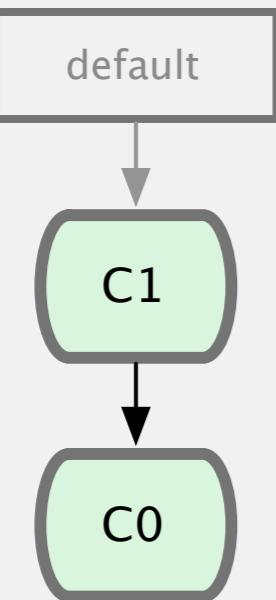




Remotes Are Branches

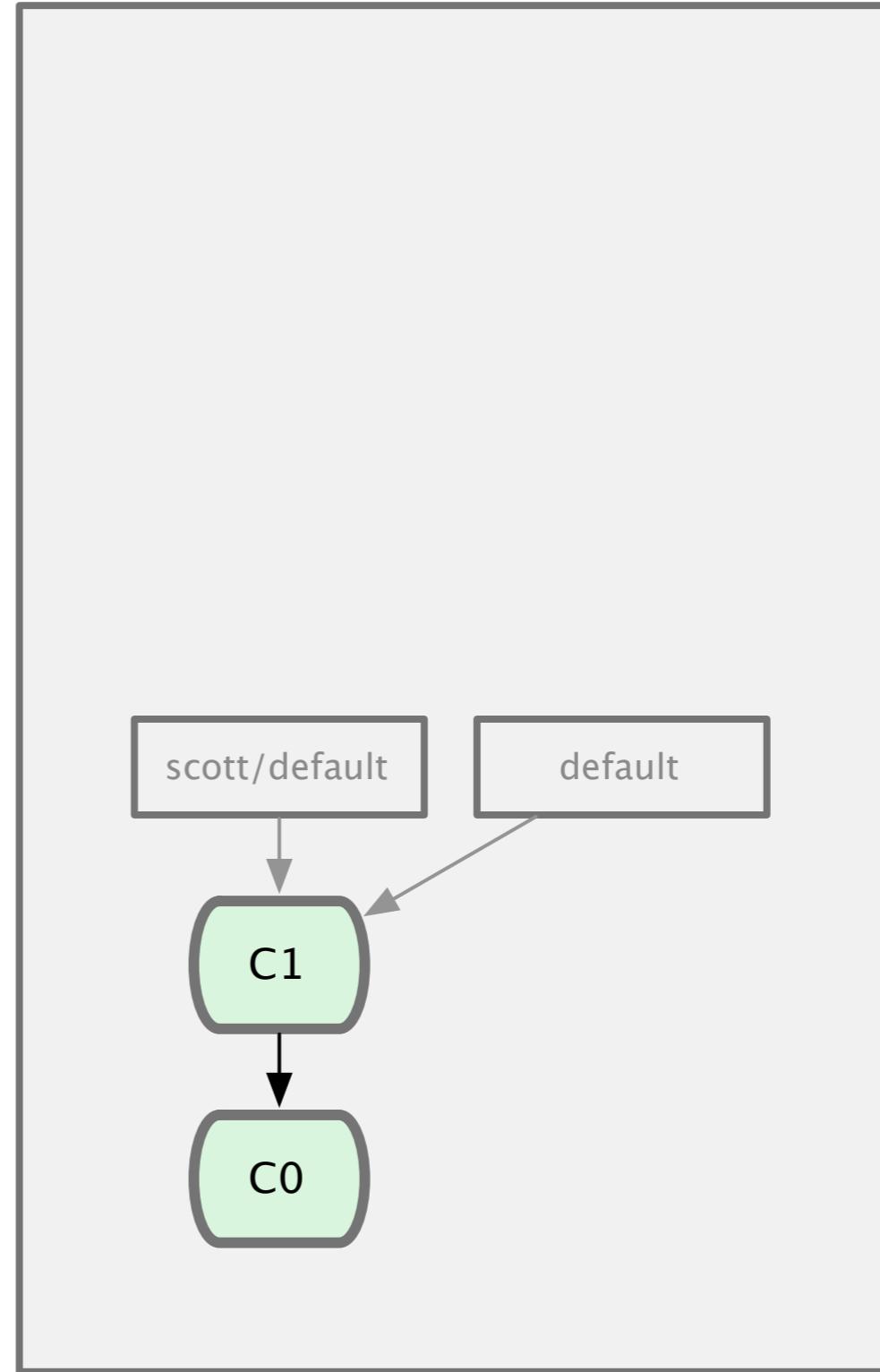
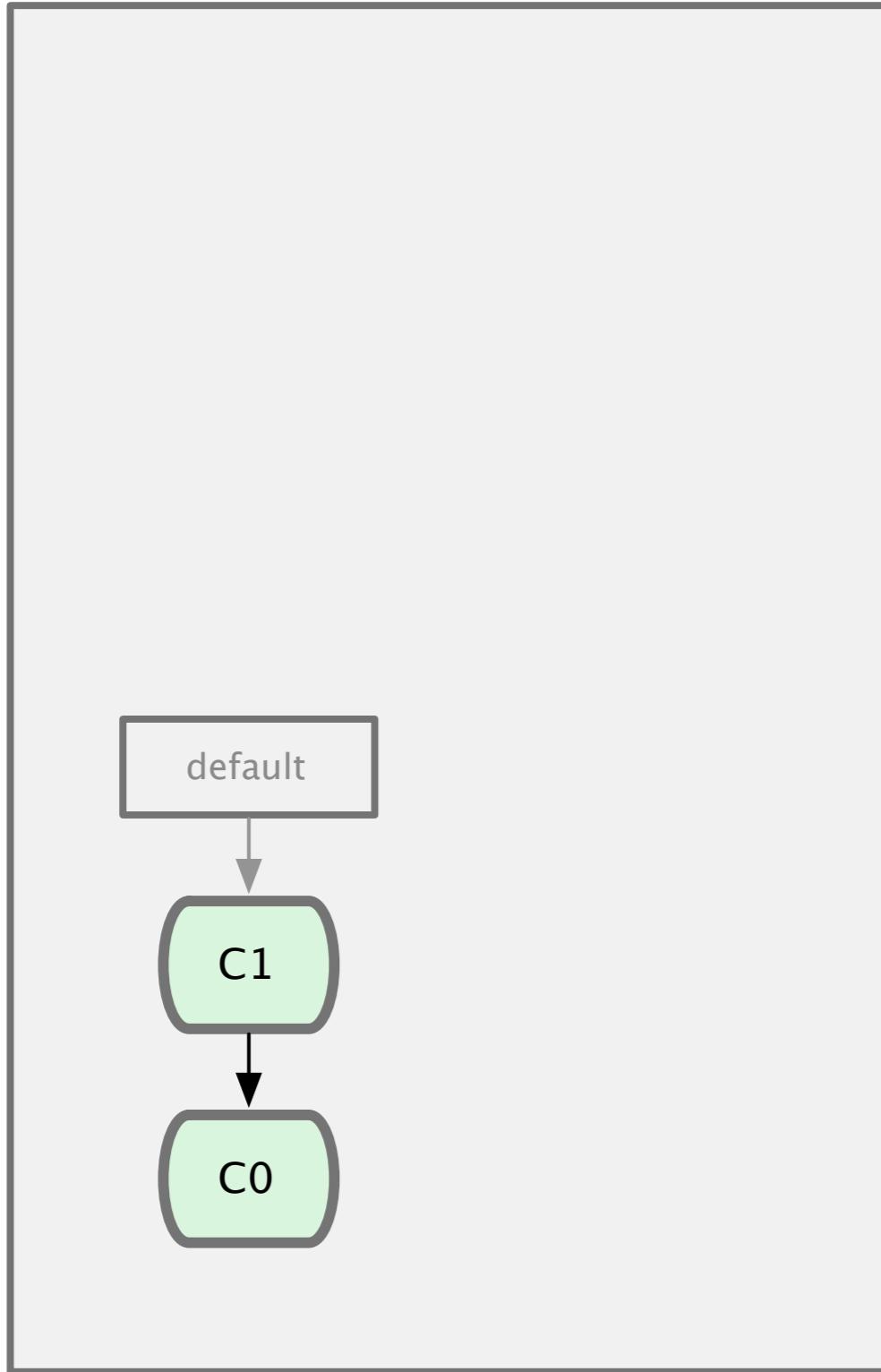
scott

jessica



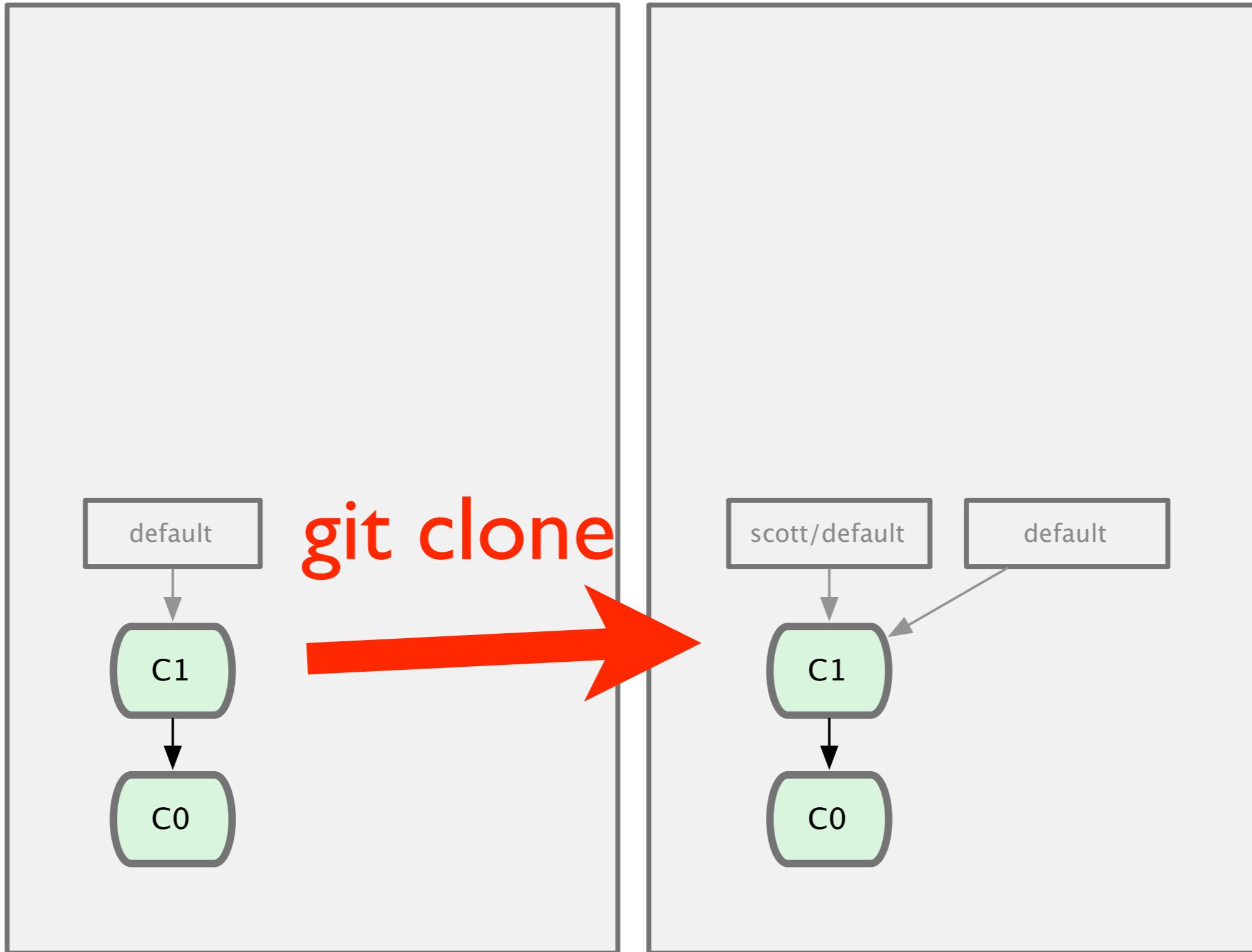
scott

jessica

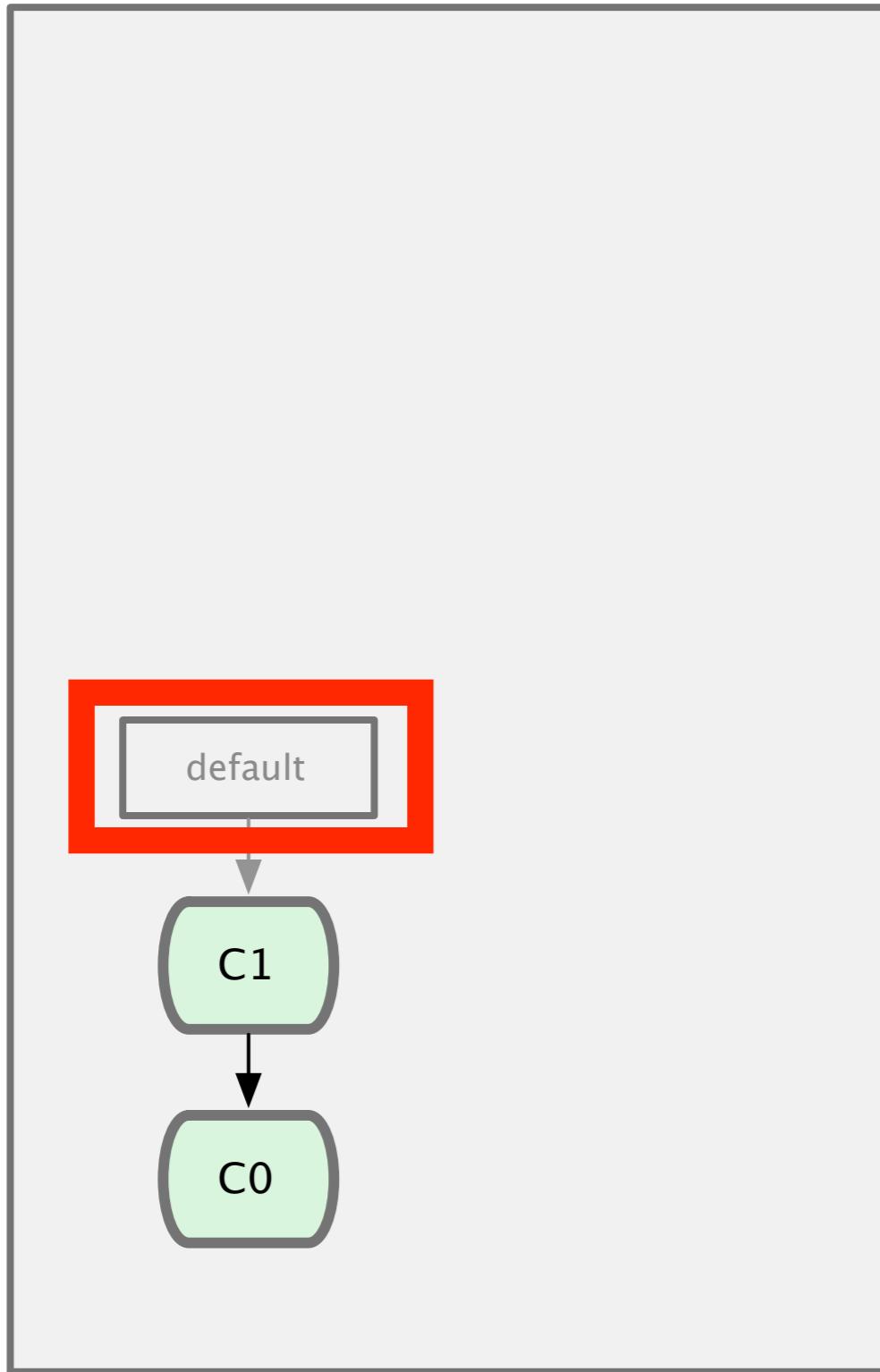


scott

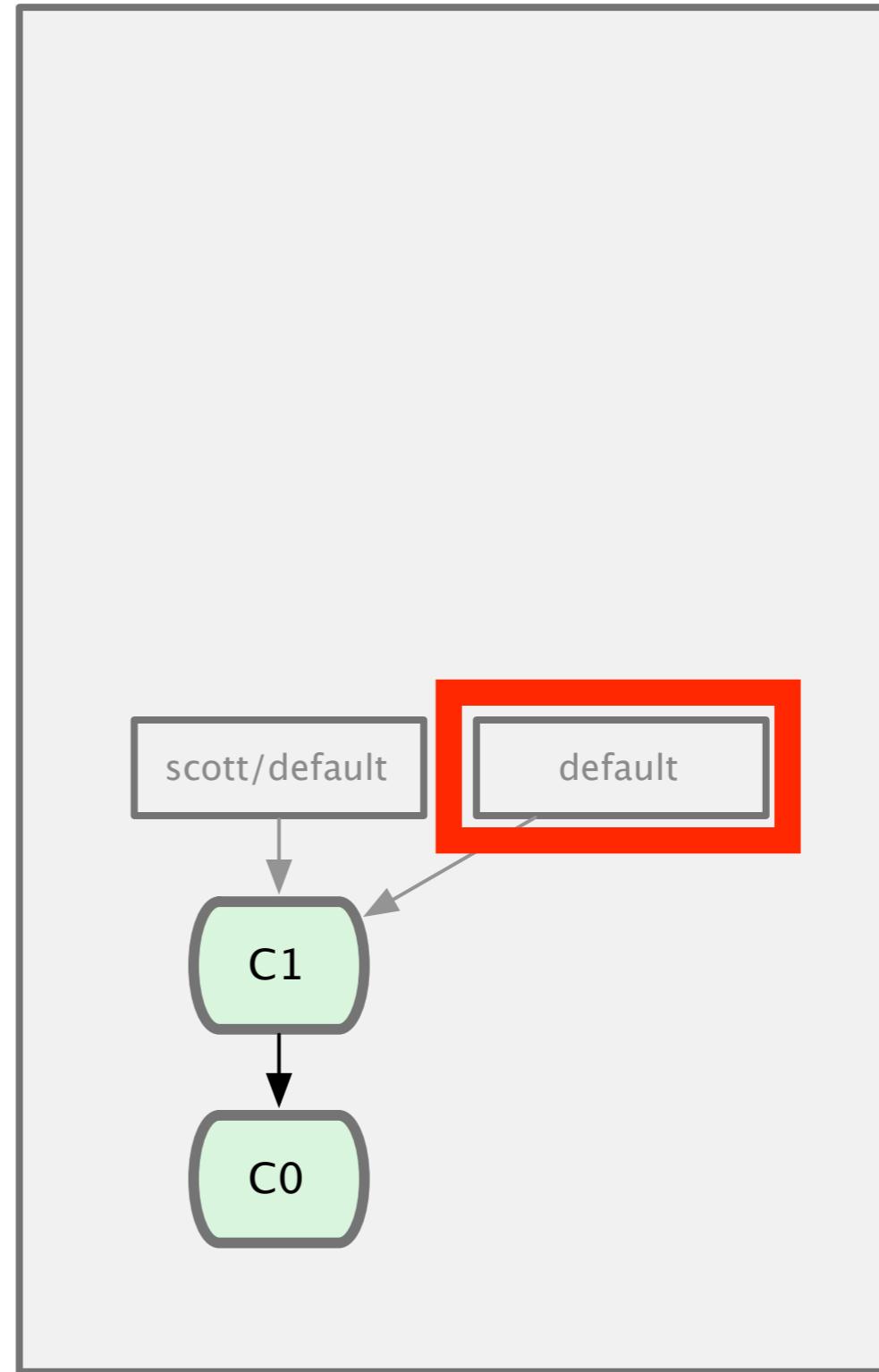
jessica



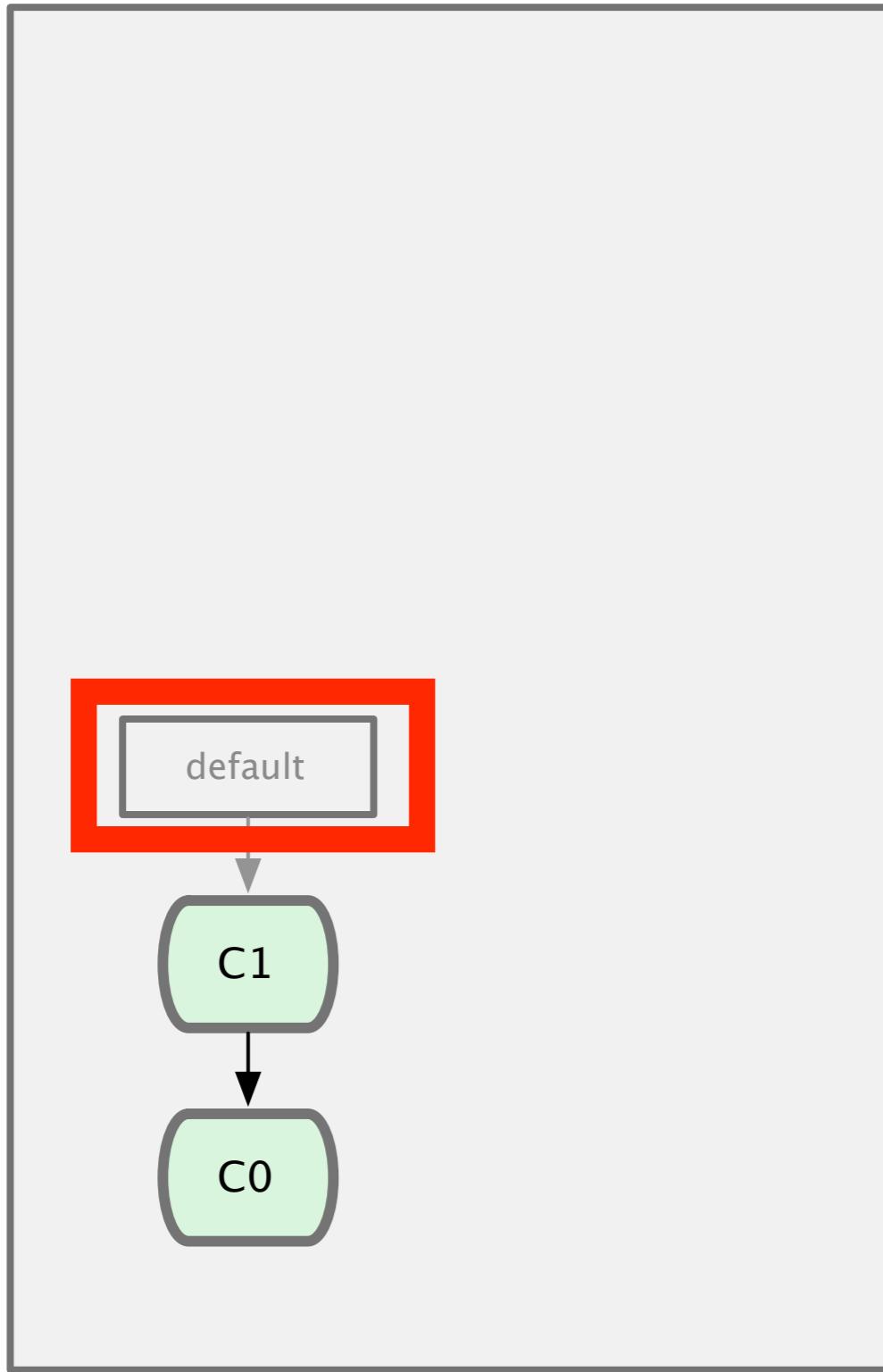
scott



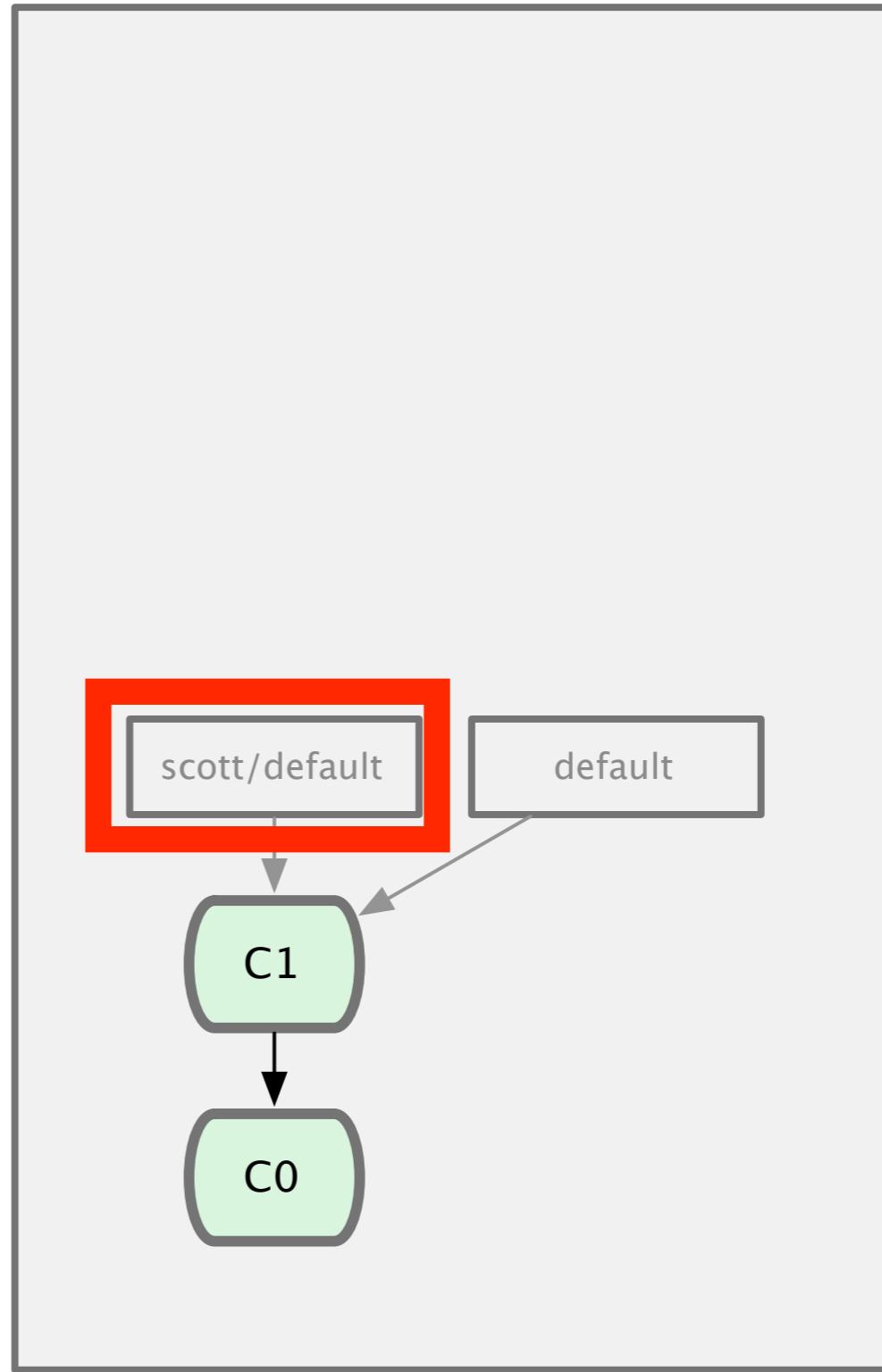
jessica



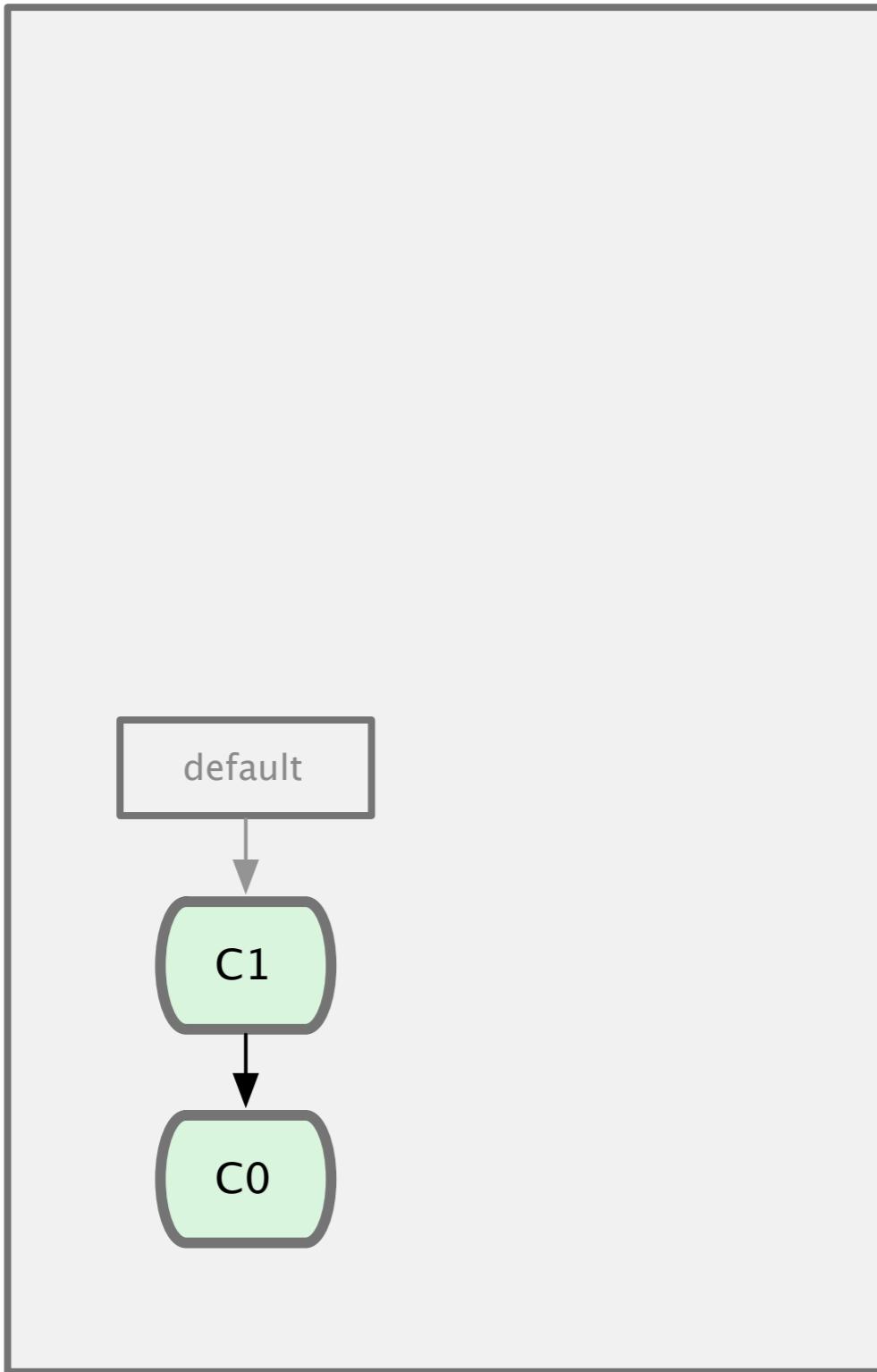
scott



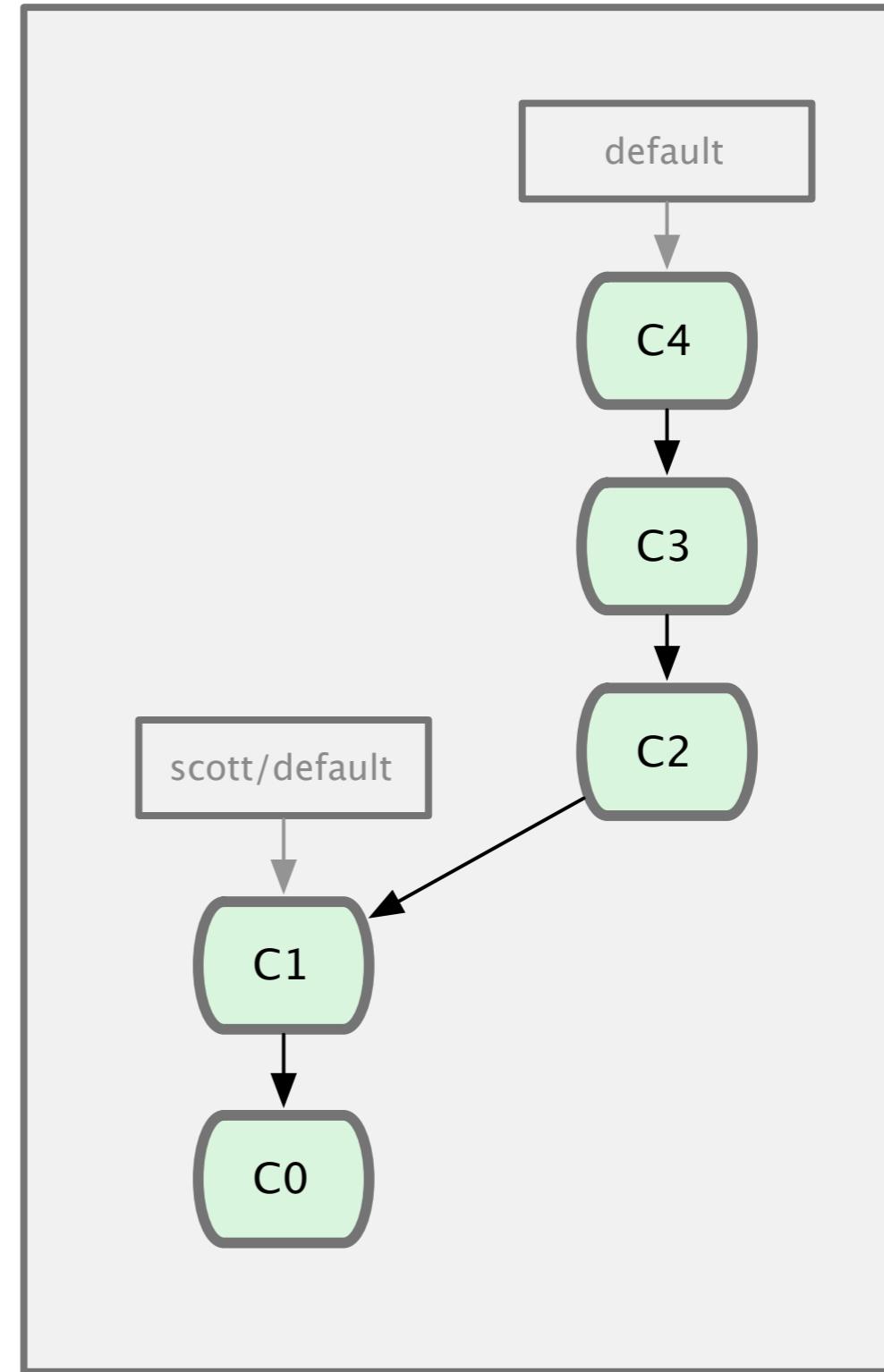
jessica



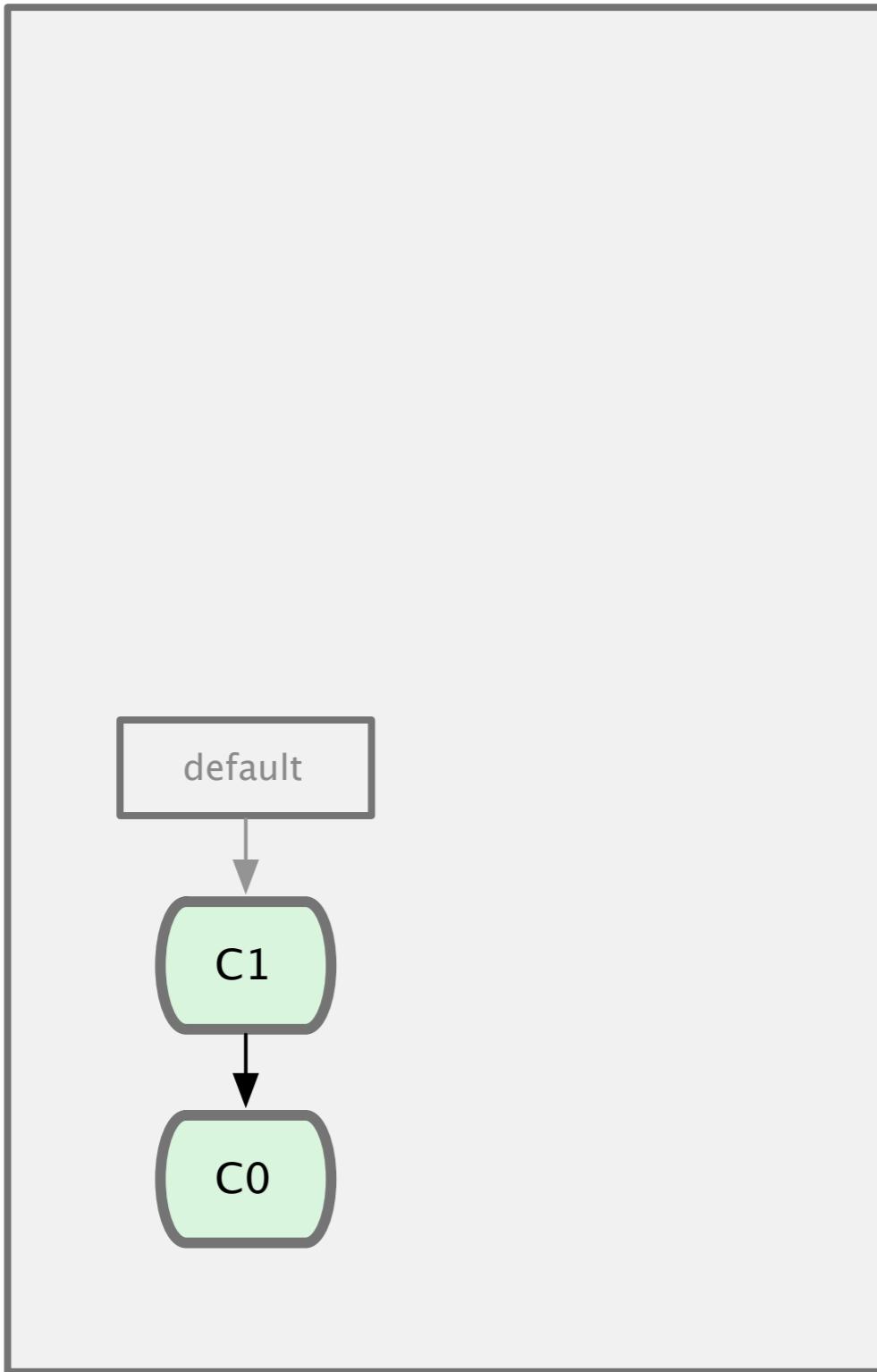
scott



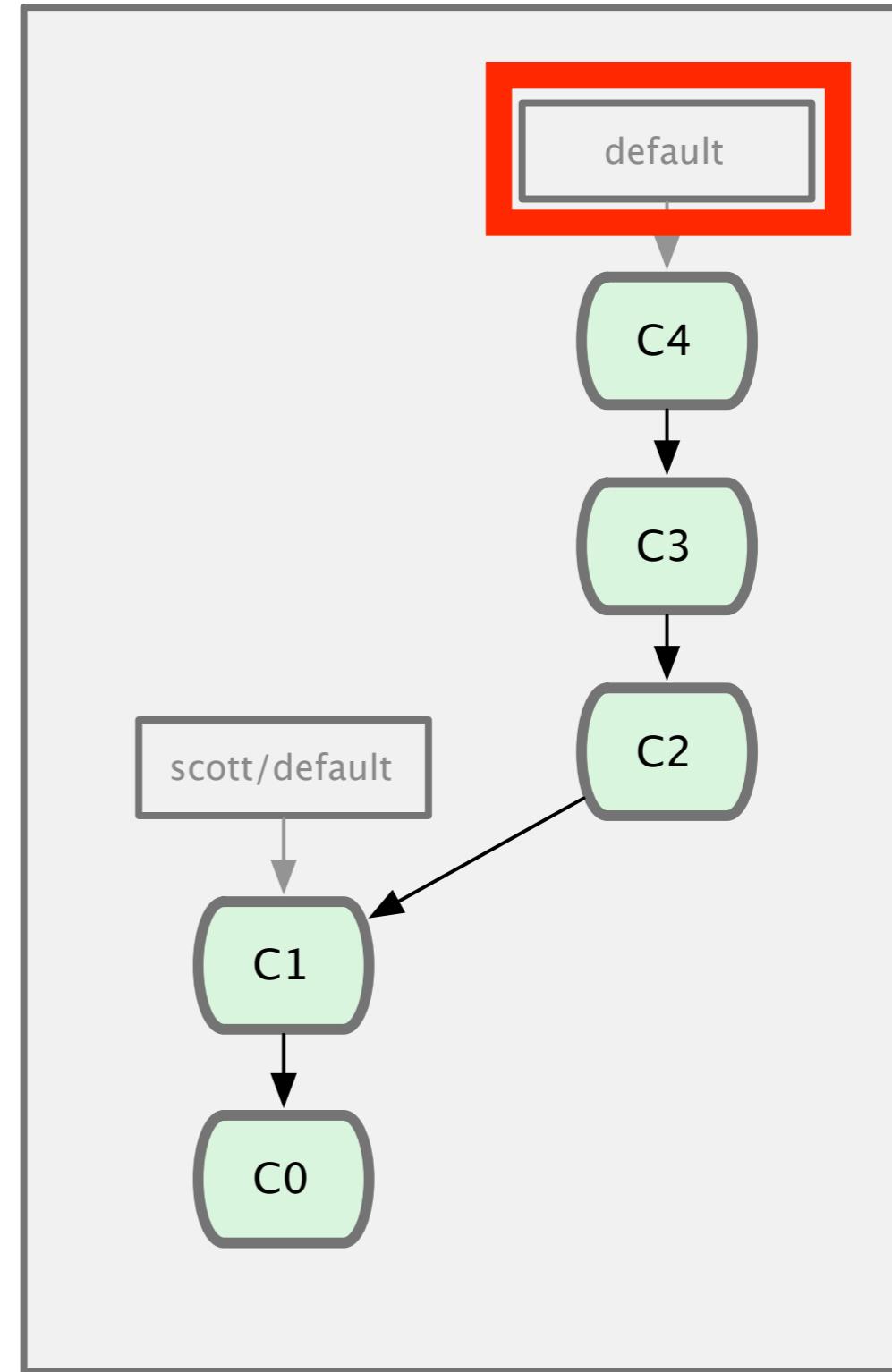
jessica



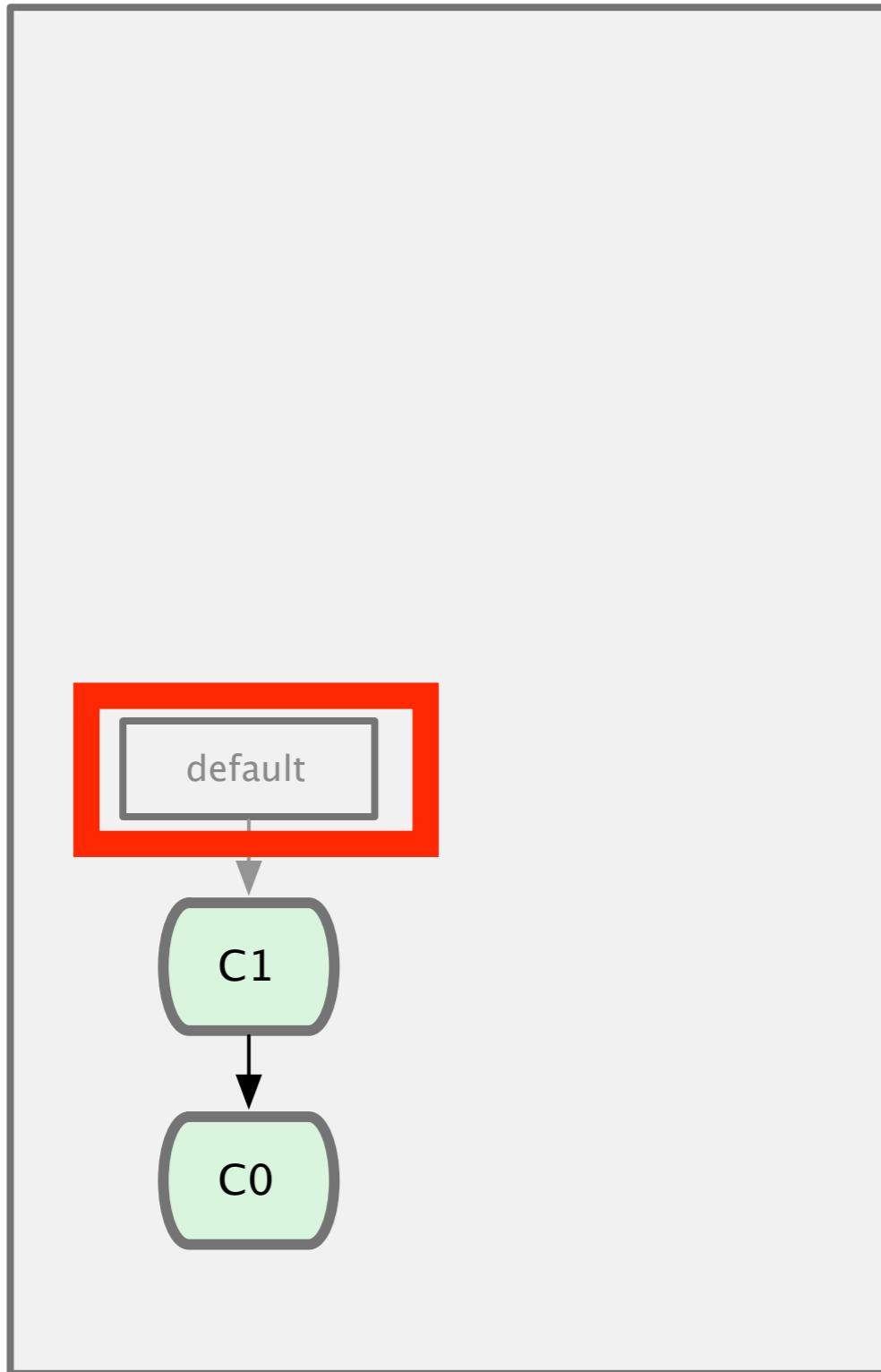
scott



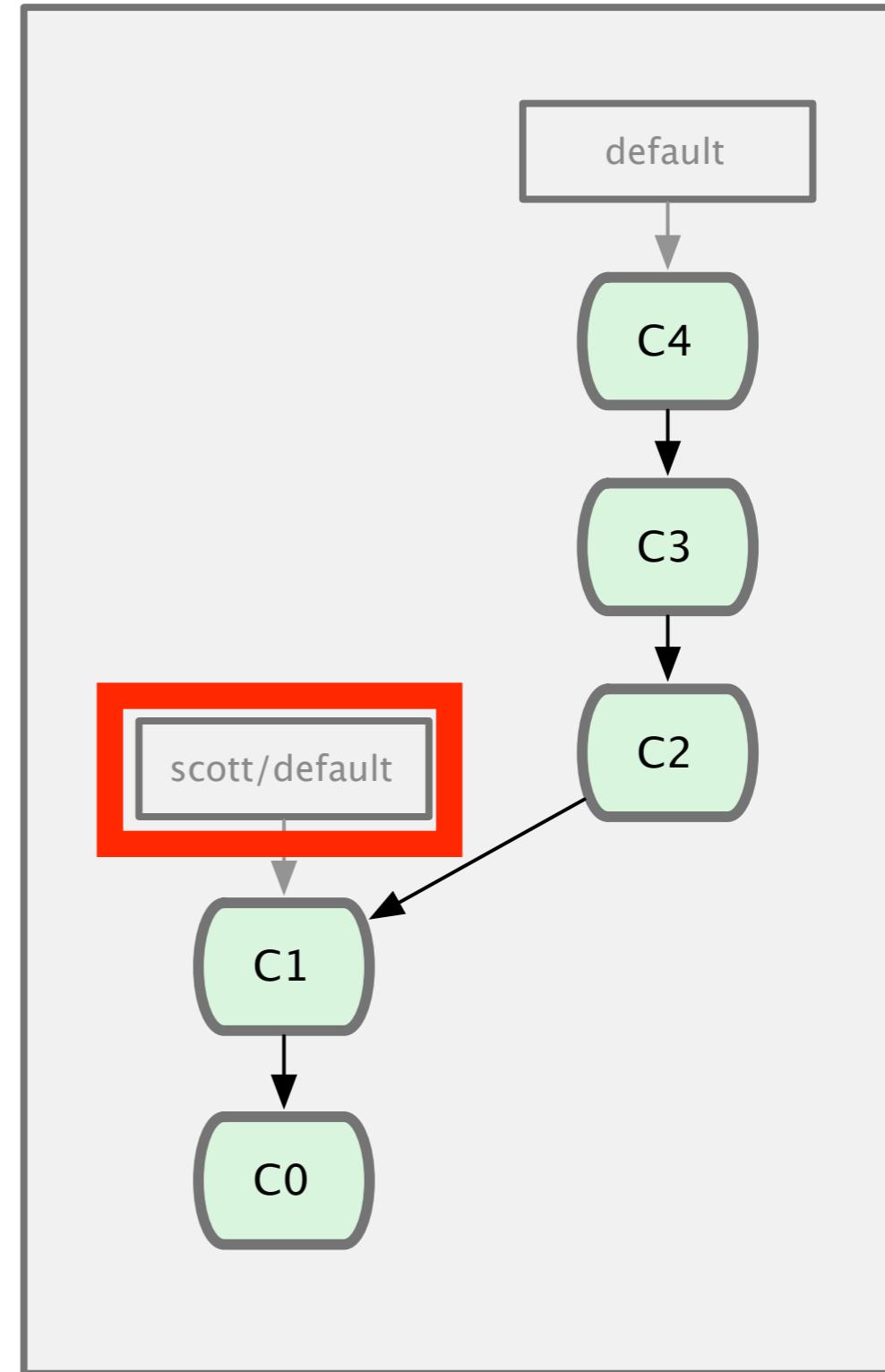
jessica



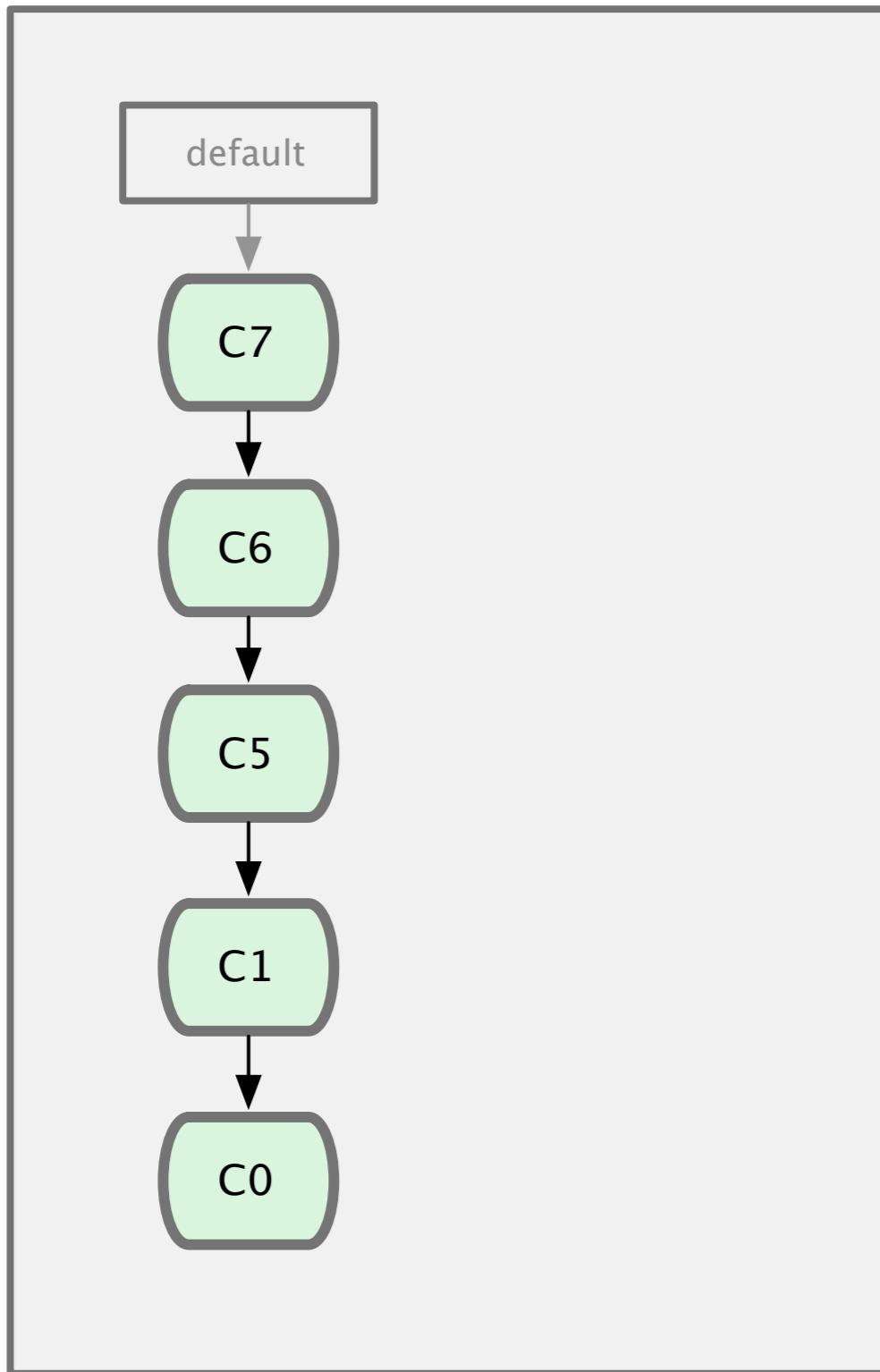
scott



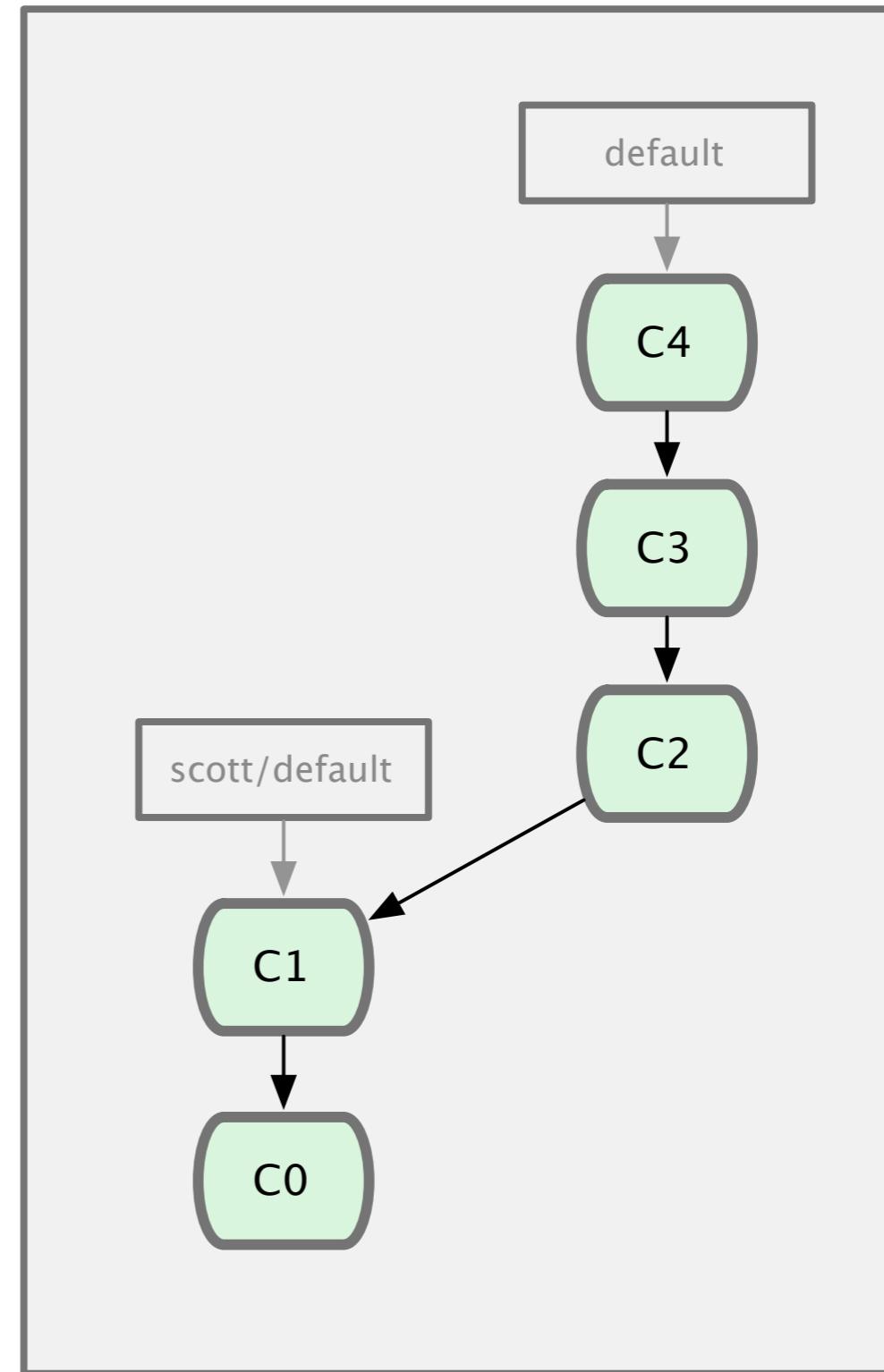
jessica



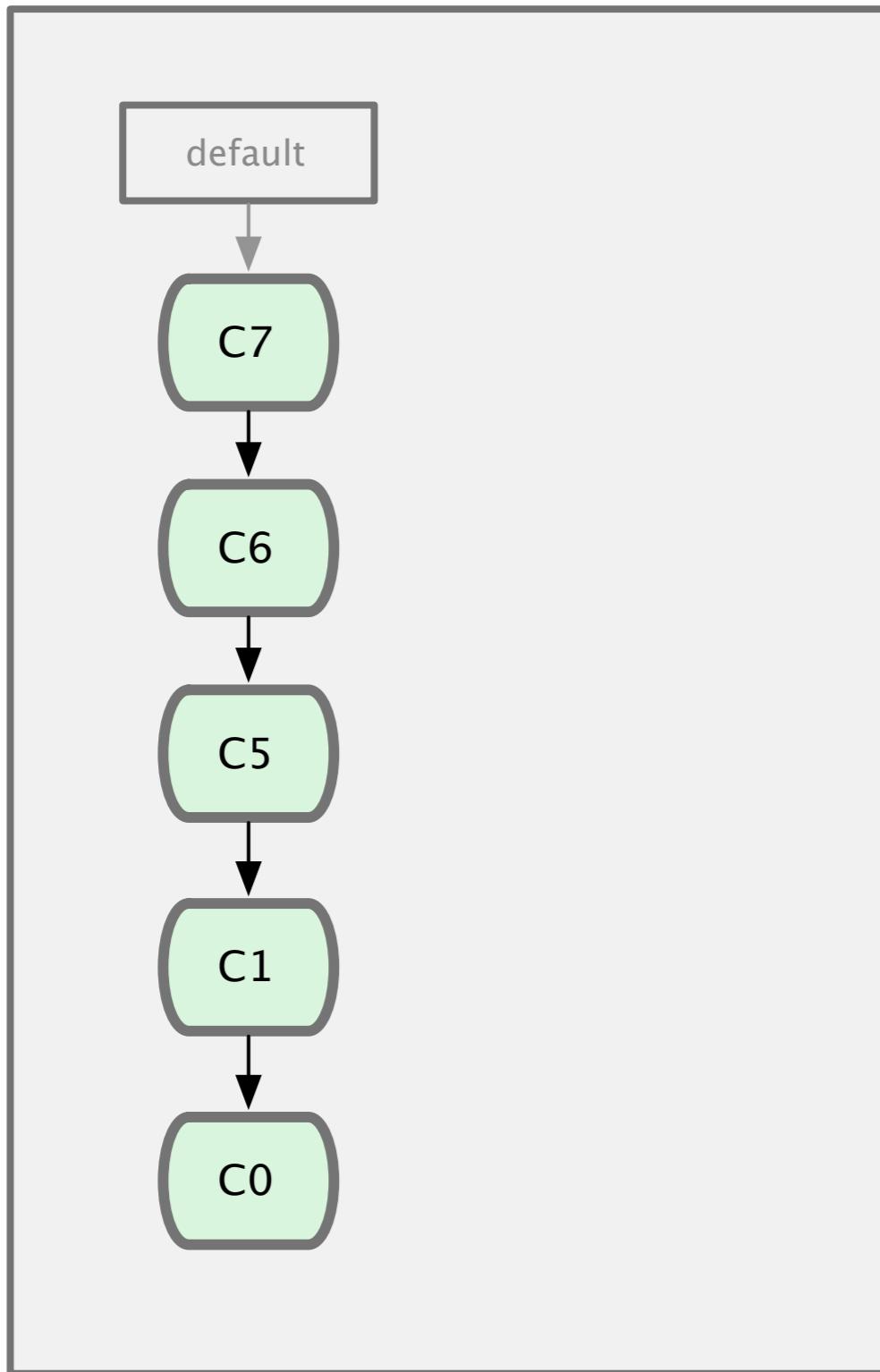
scott



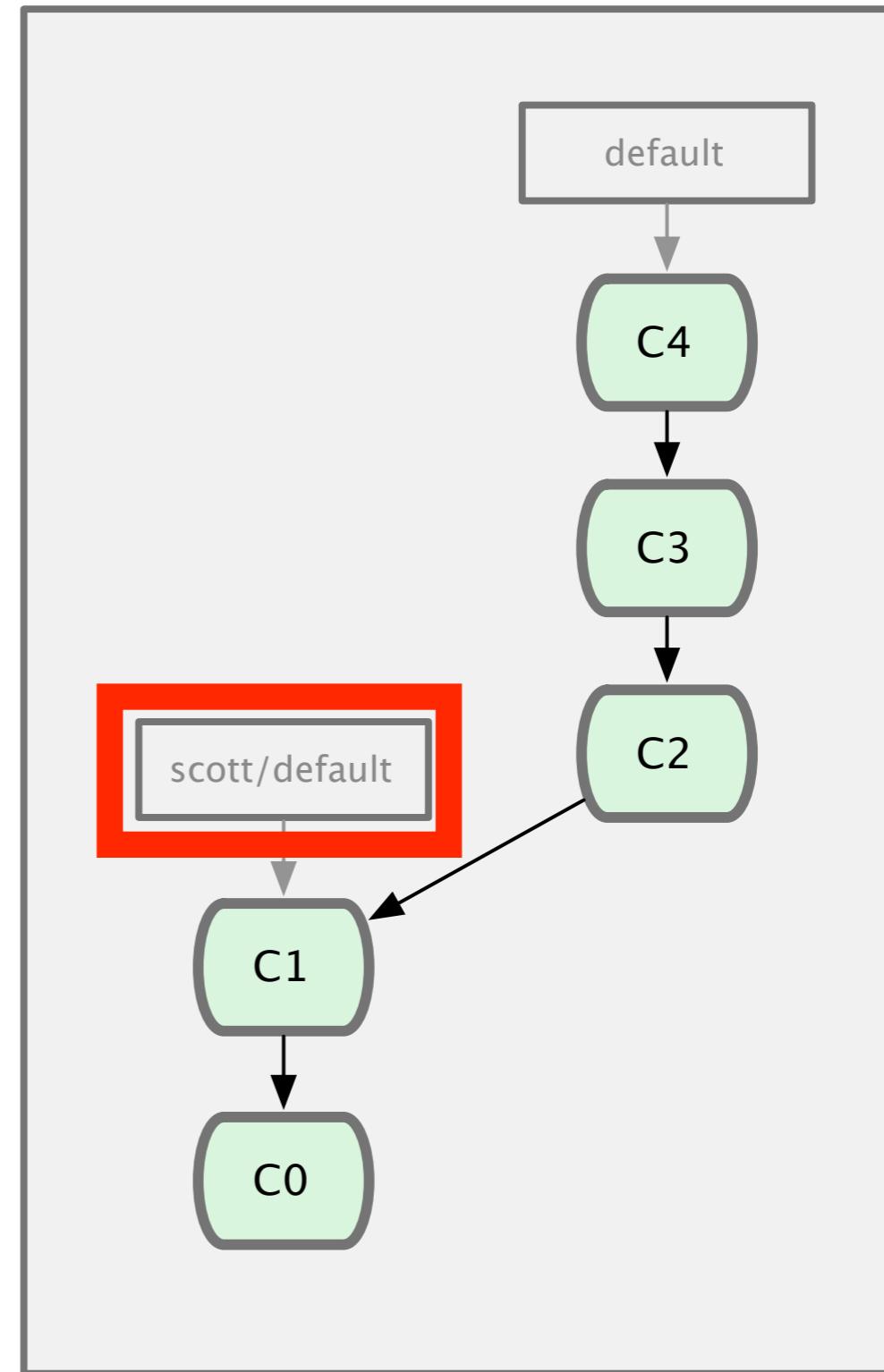
jessica



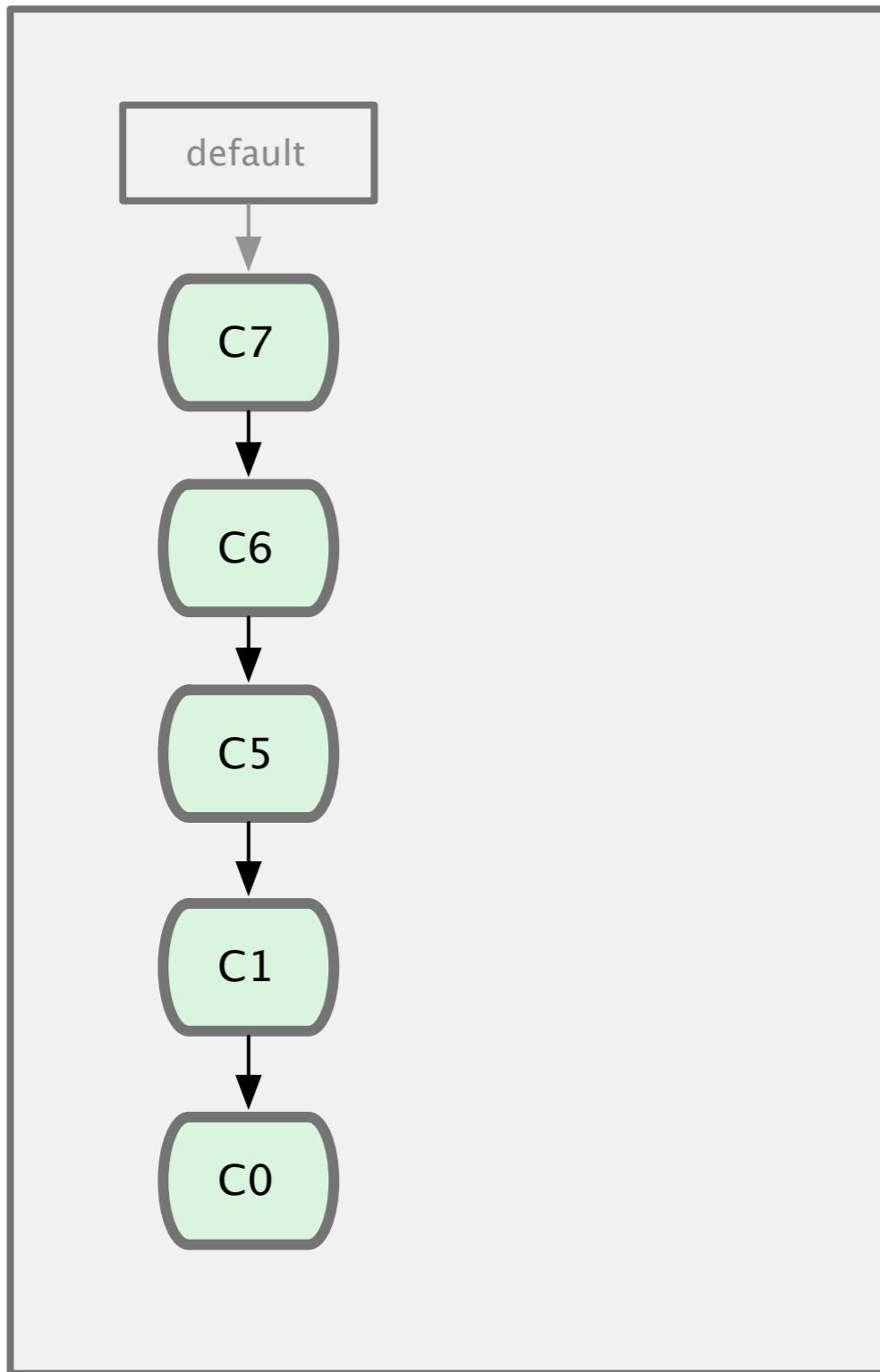
scott



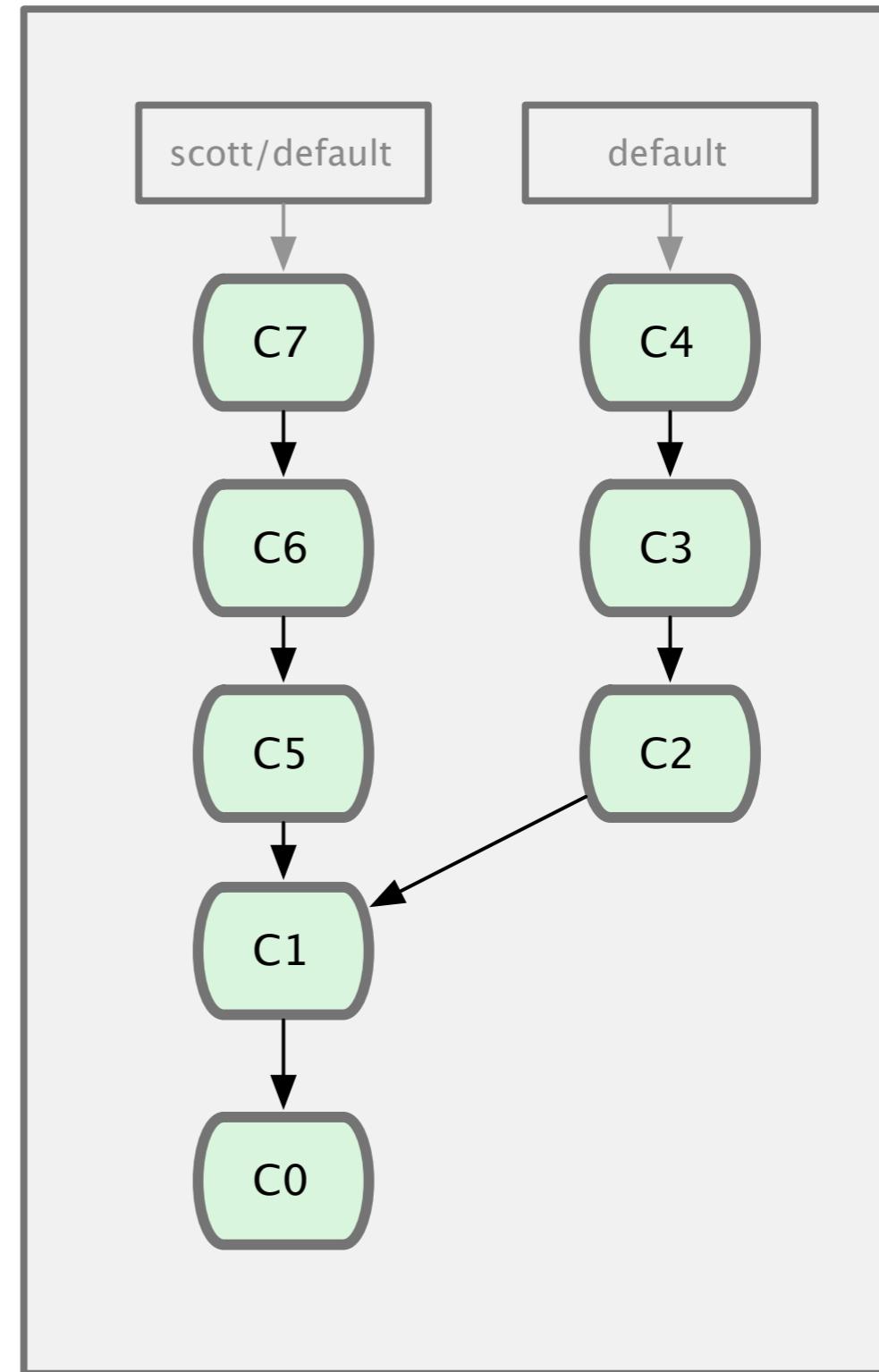
jessica



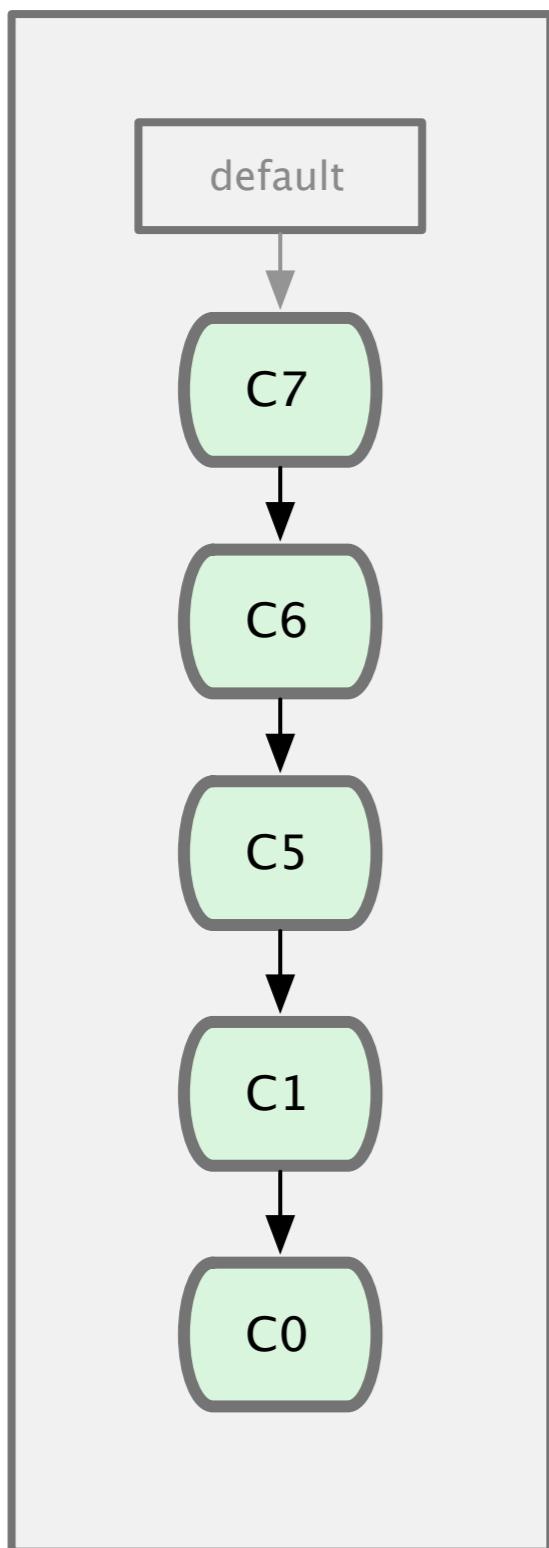
scott



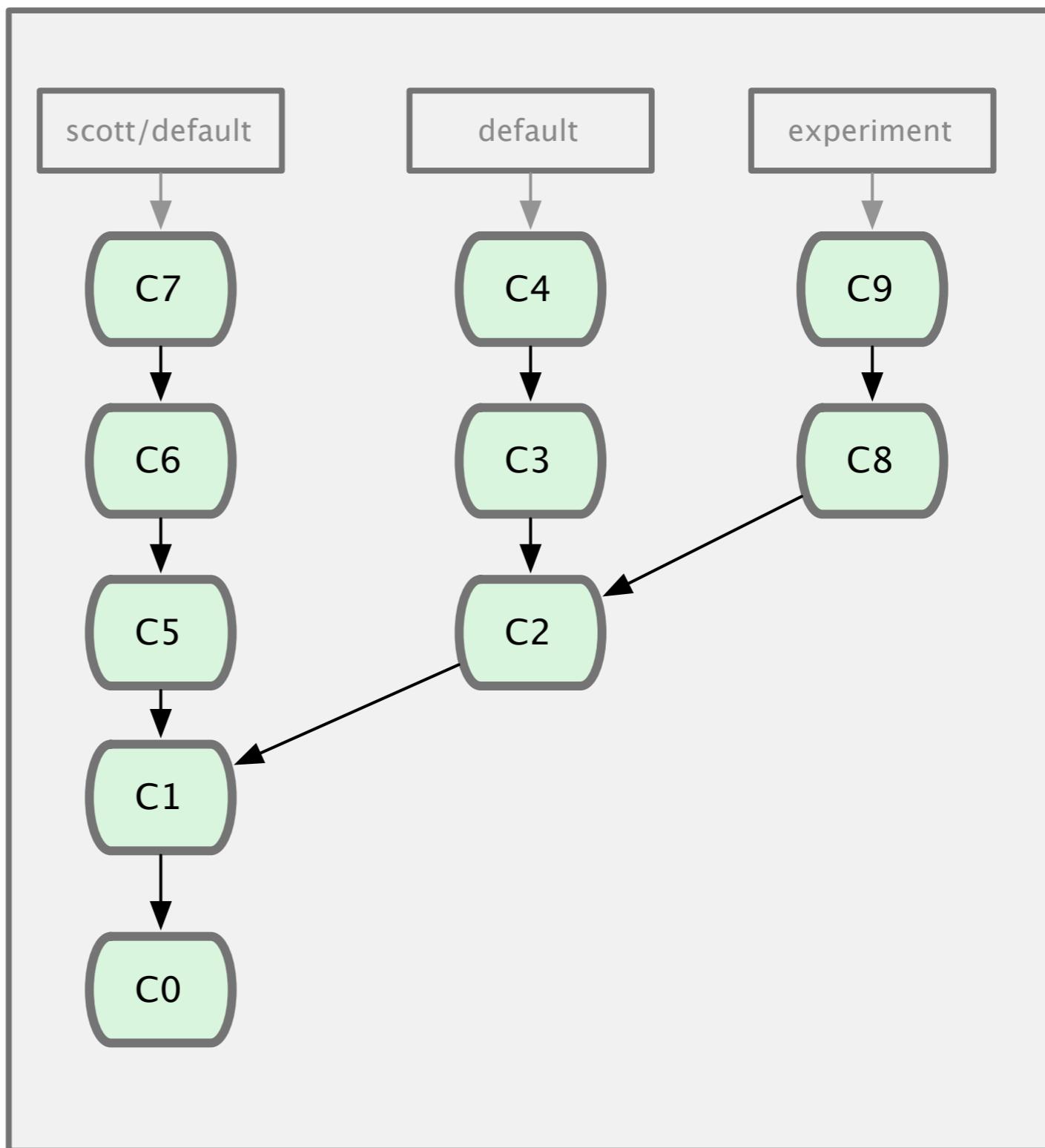
jessica



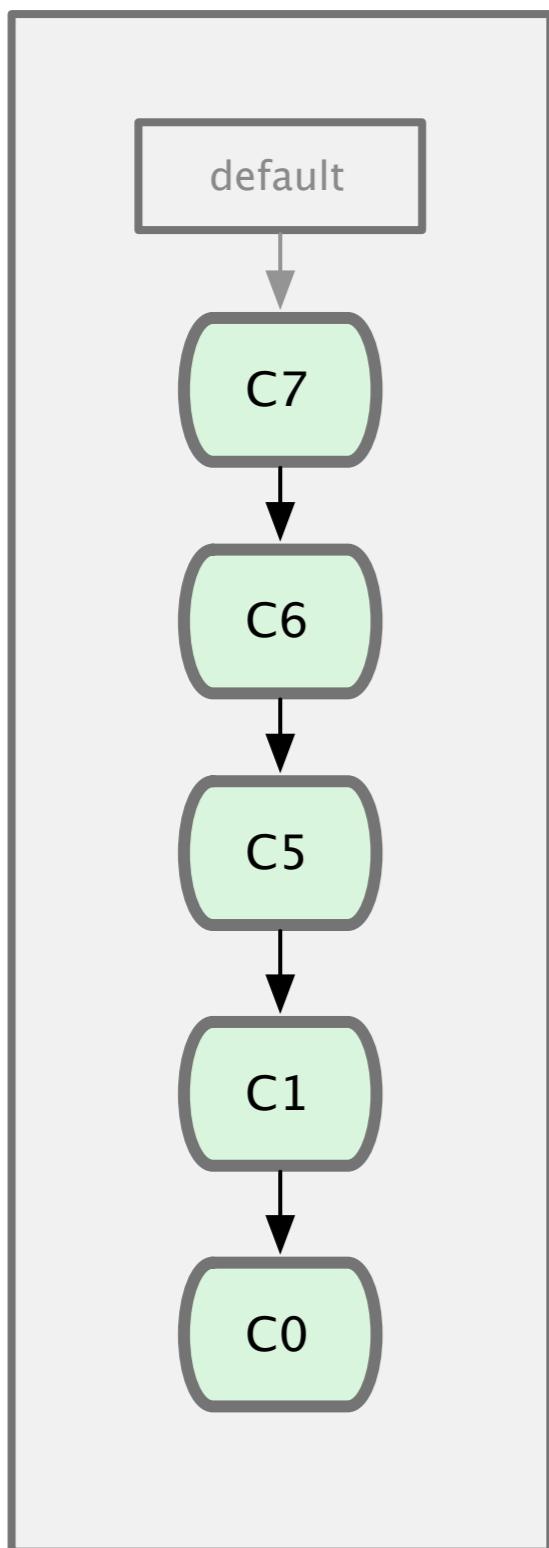
scott



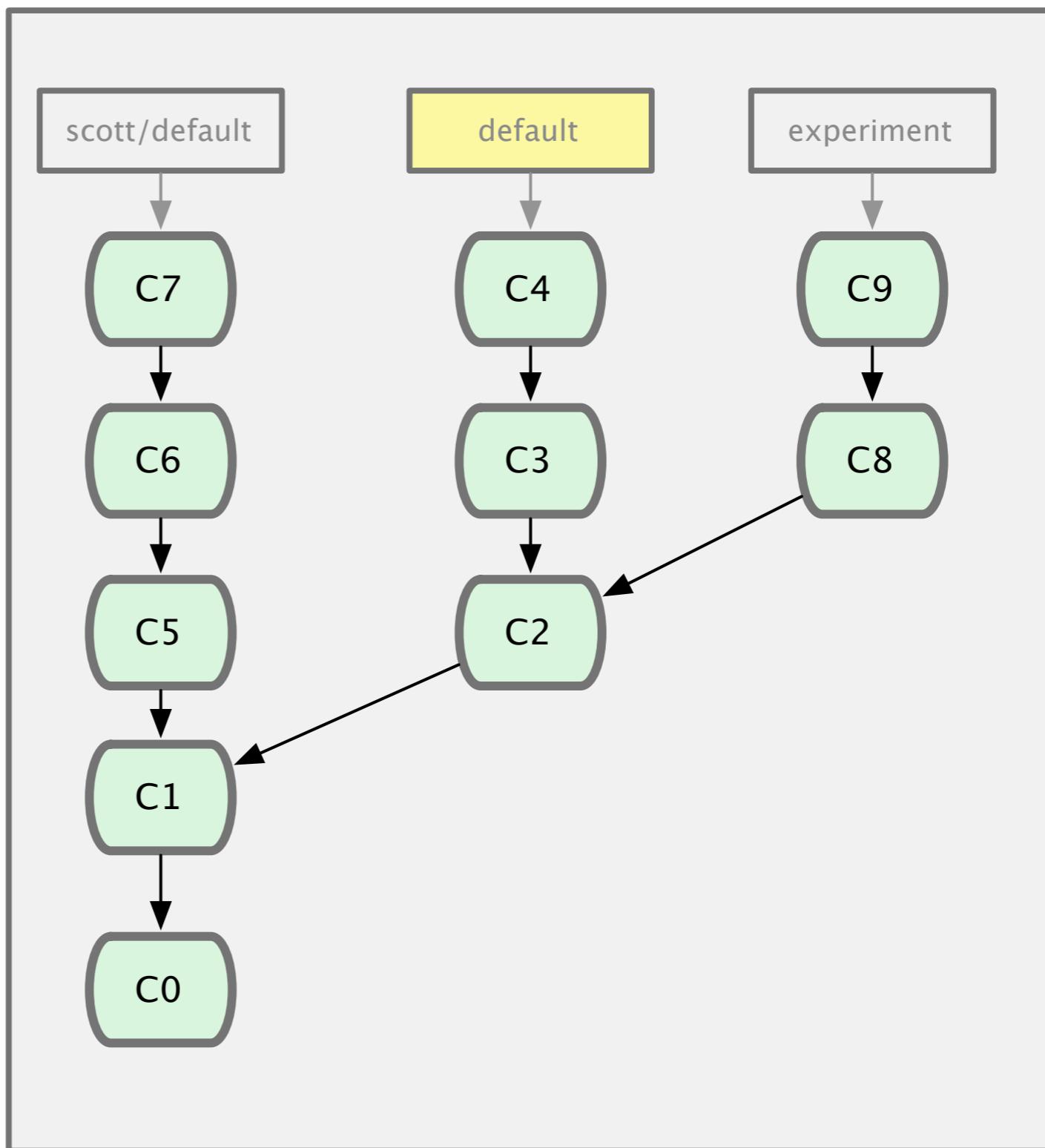
jessica



scott

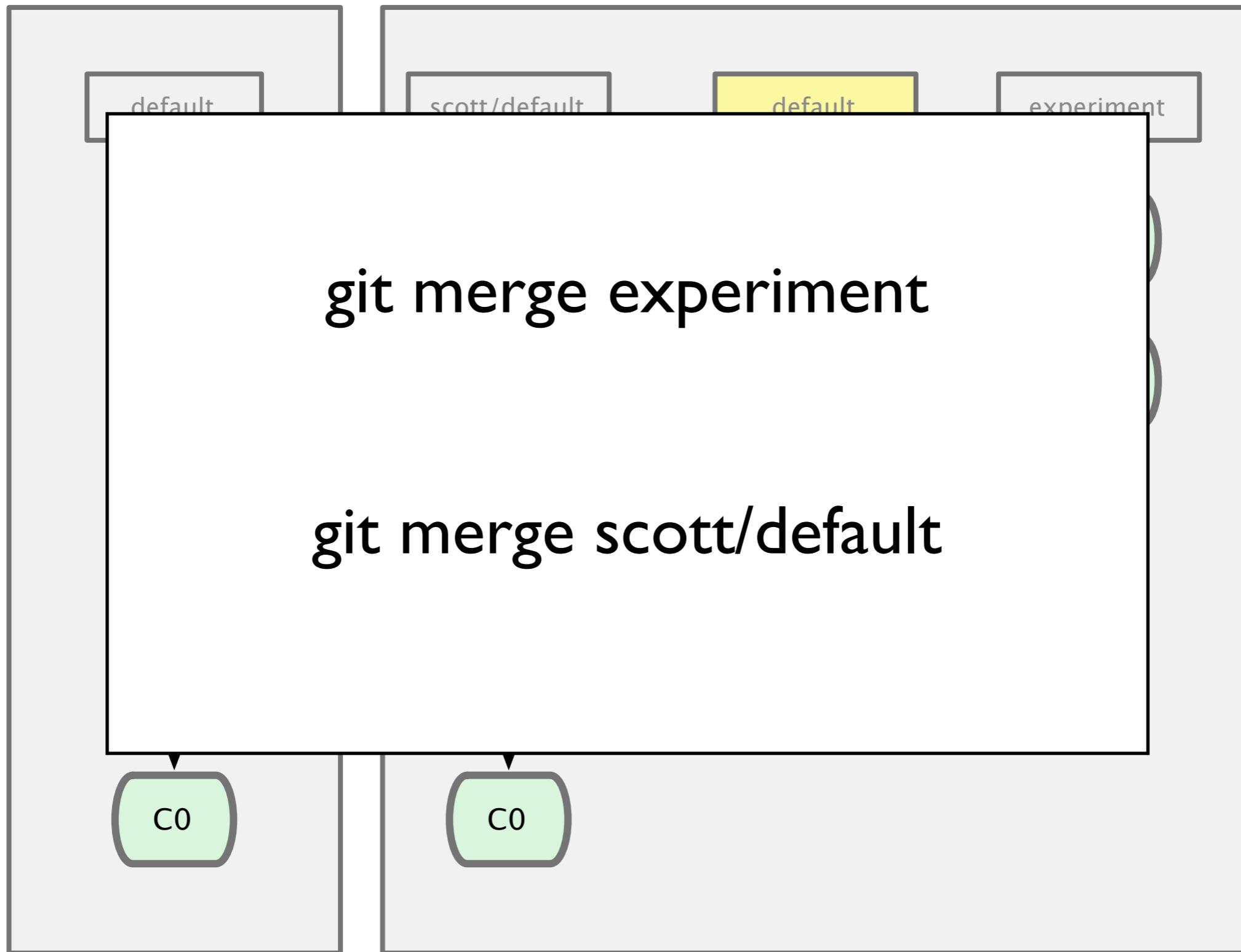


jessica

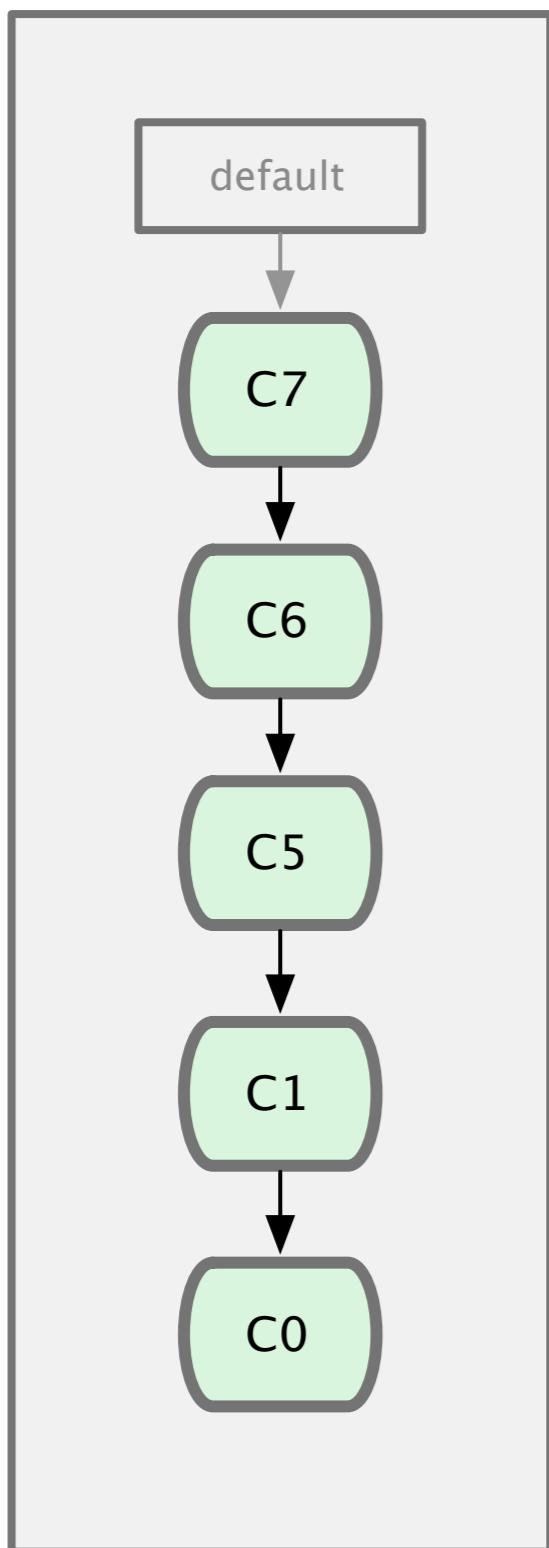


scott

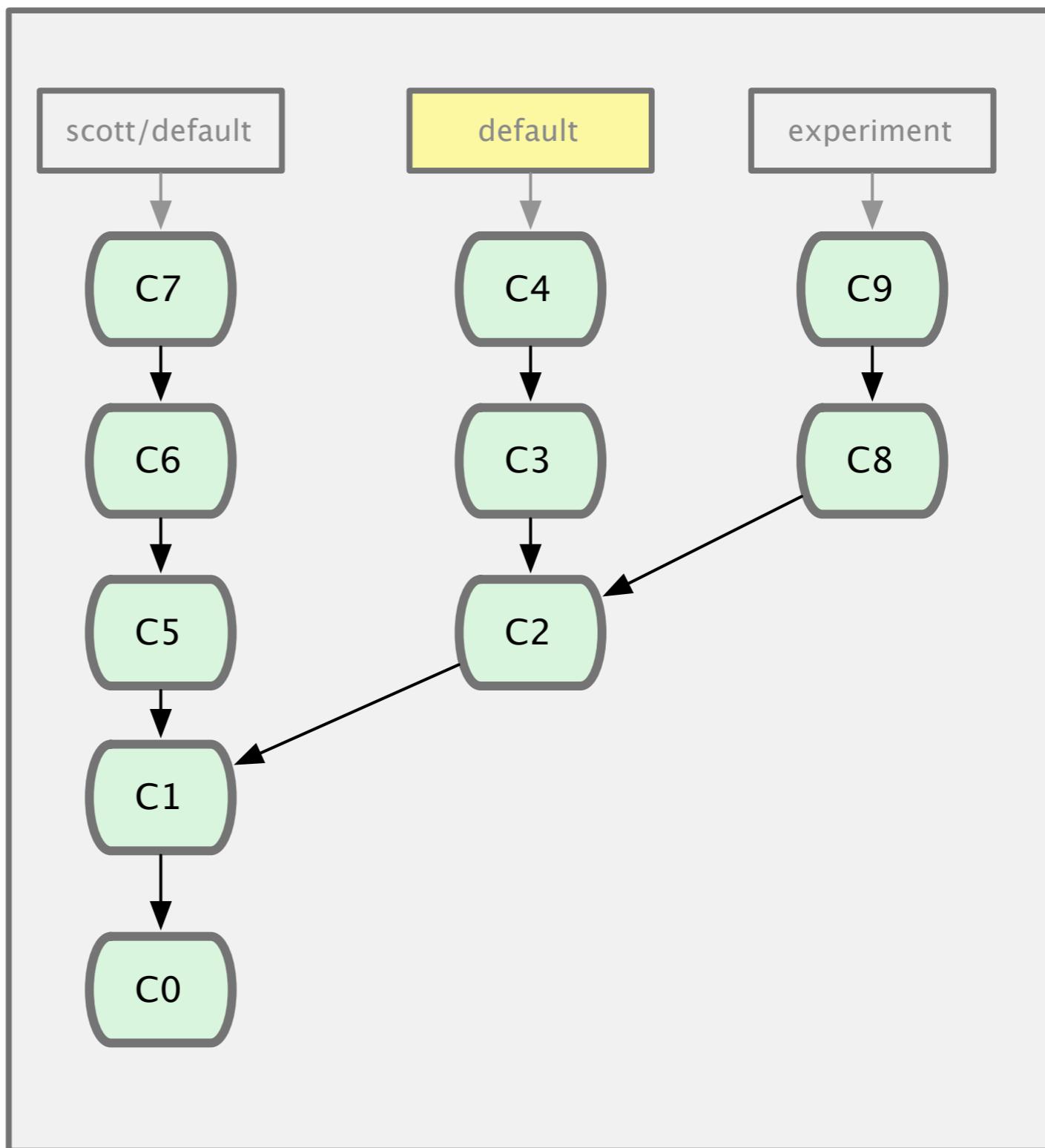
jessica



scott

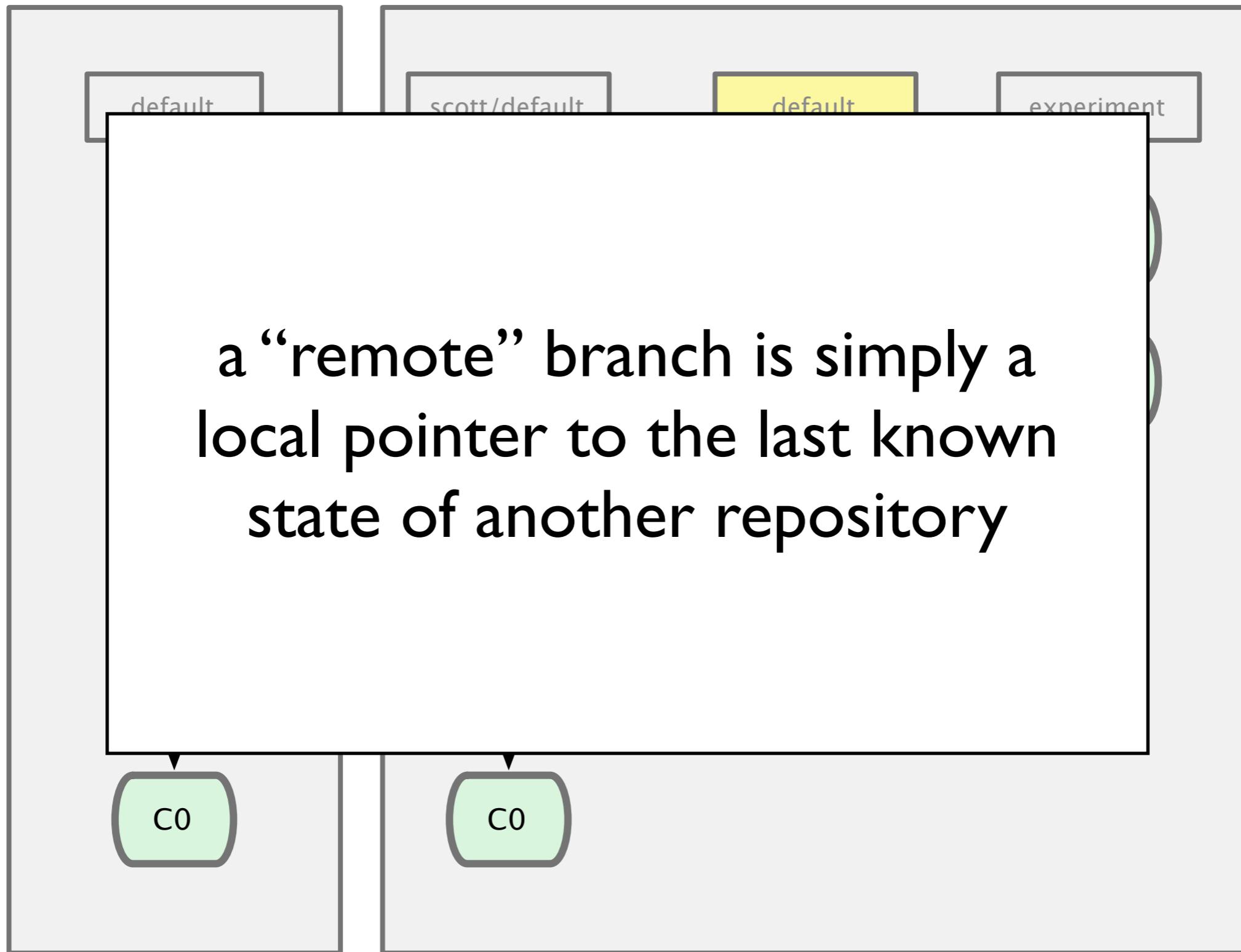


jessica



scott

jessica



Why is this cool?

non-linear development

- clone the code that is in production

- clone the code that is in production
- create a branch for issue #53 (iss53)

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’

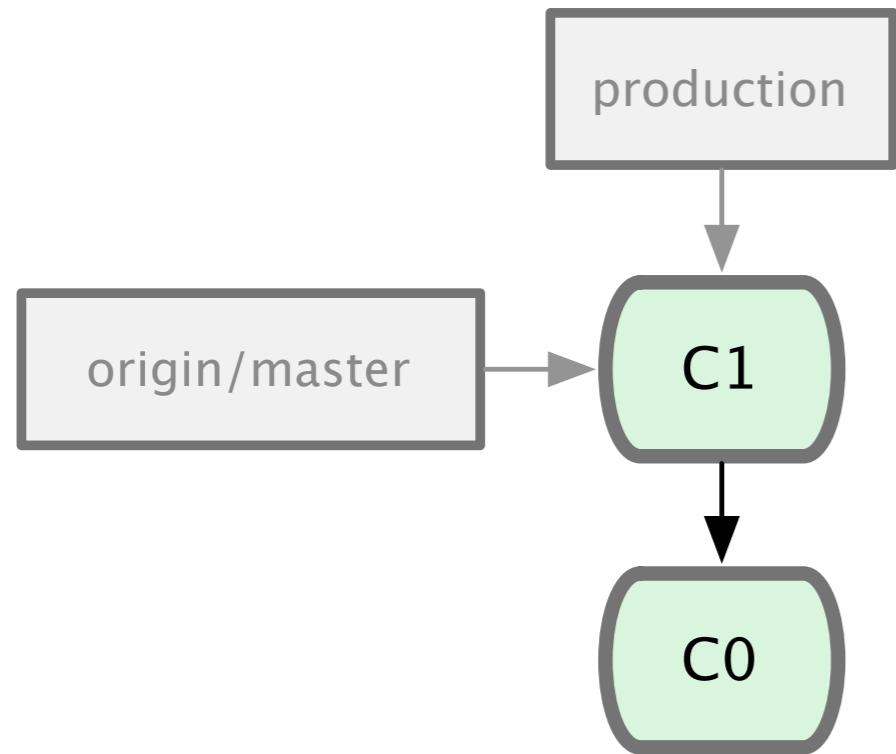
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’
- create a branch (iss102)

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’
- create a branch (iss102)
- fix the issue

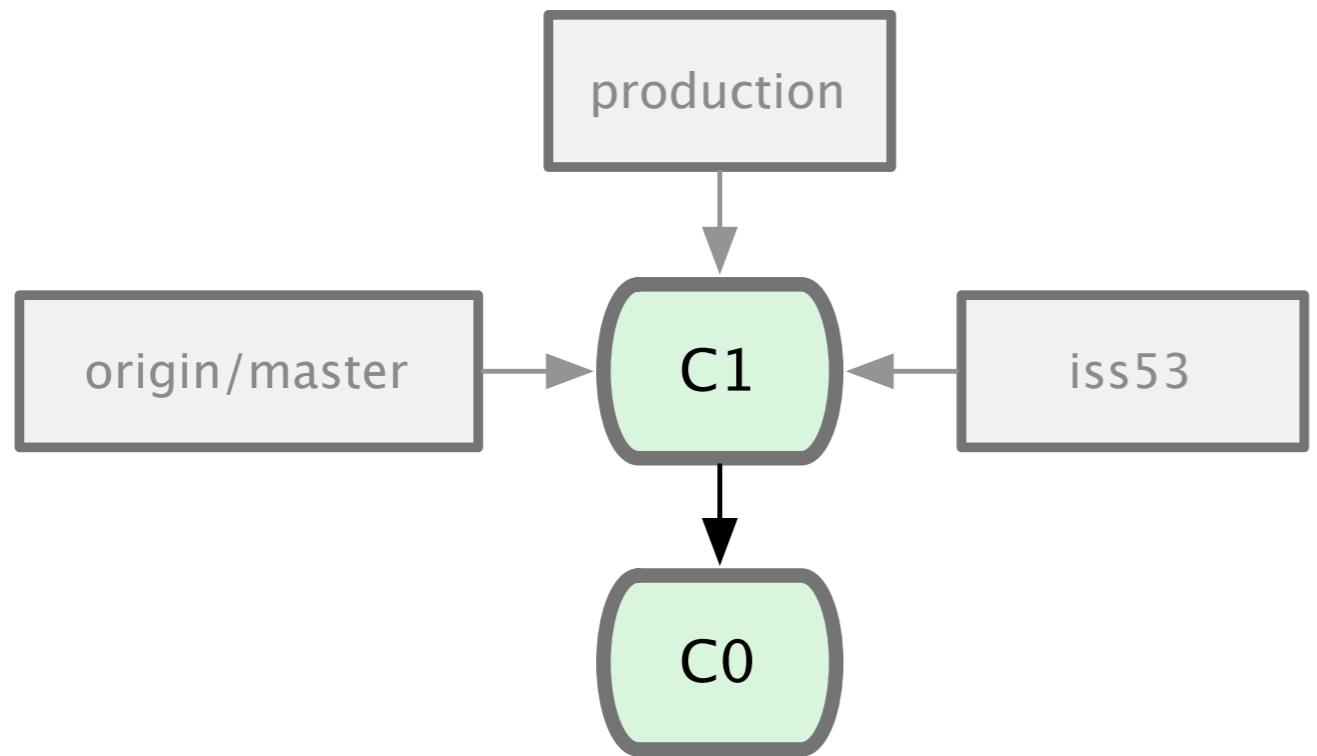
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’
- create a branch (iss102)
- fix the issue
- checkout ‘production’, merge ‘iss102’

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
 - checkout ‘production’
 - create a branch (iss102)
 - fix the issue
 - checkout ‘production’, merge ‘iss102’
 - push ‘production’

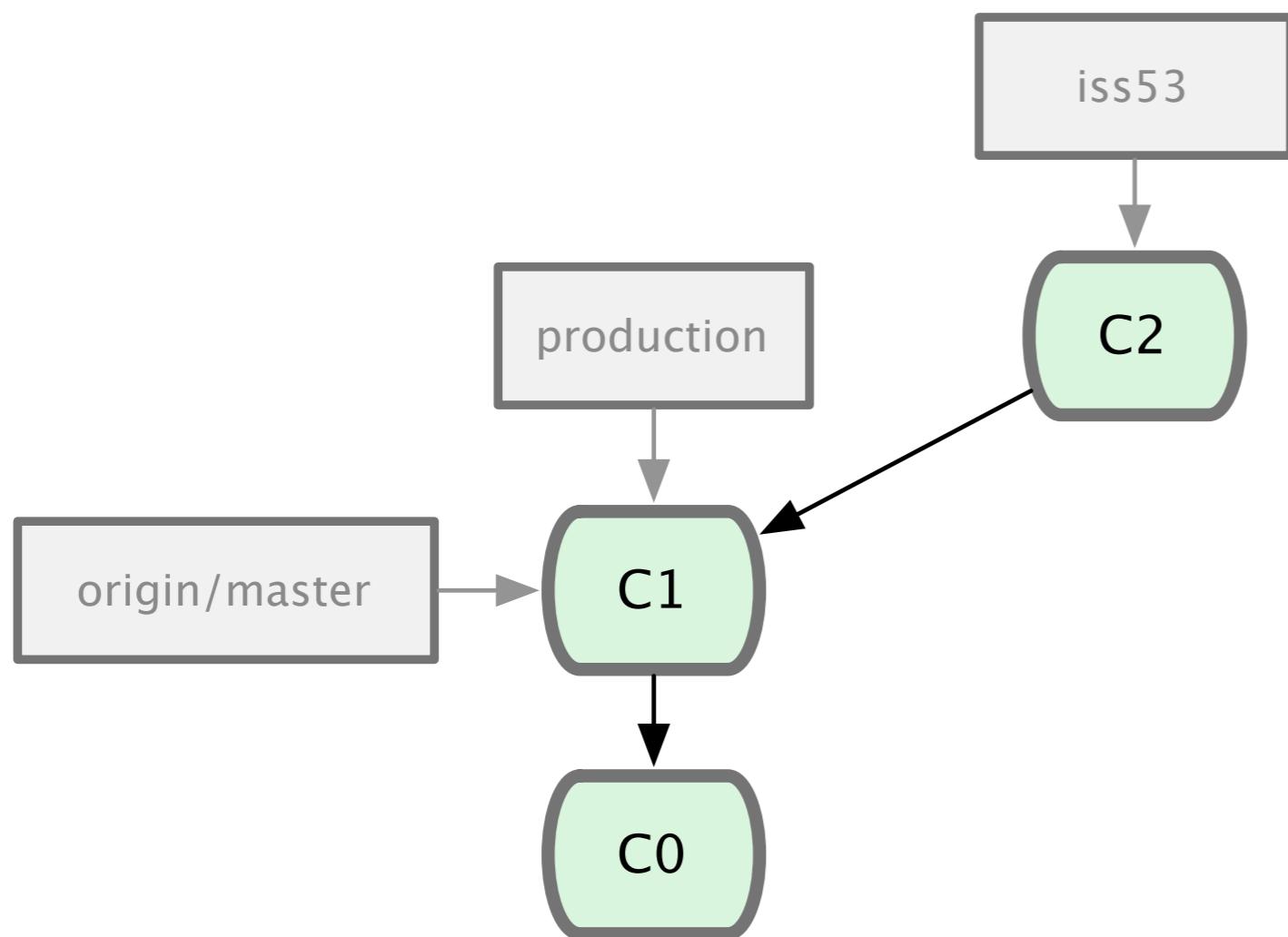
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
 - checkout ‘production’
 - create a branch (iss102)
 - fix the issue
 - checkout ‘production’, merge ‘iss102’
 - push ‘production’
 - etc



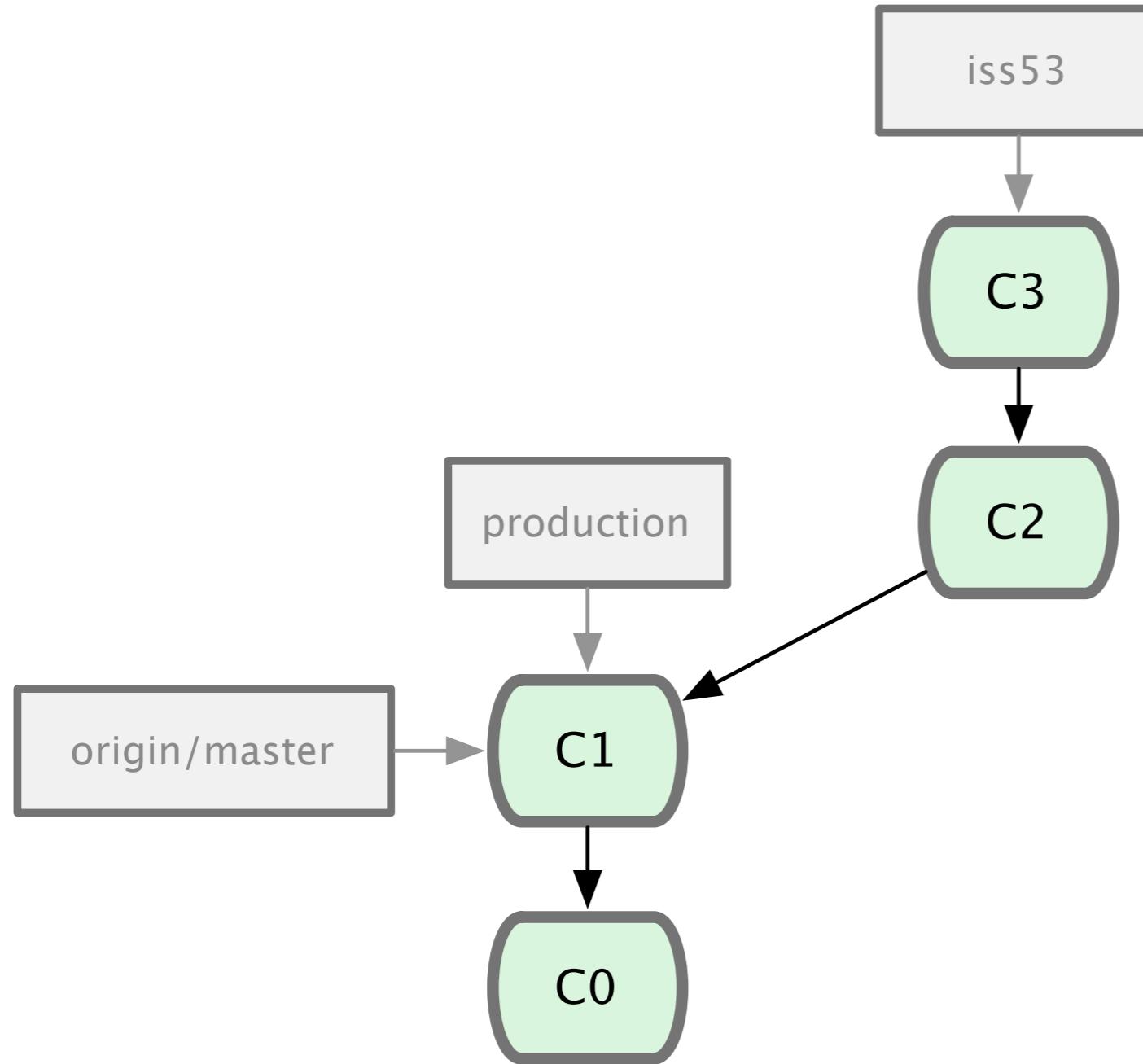
git clone



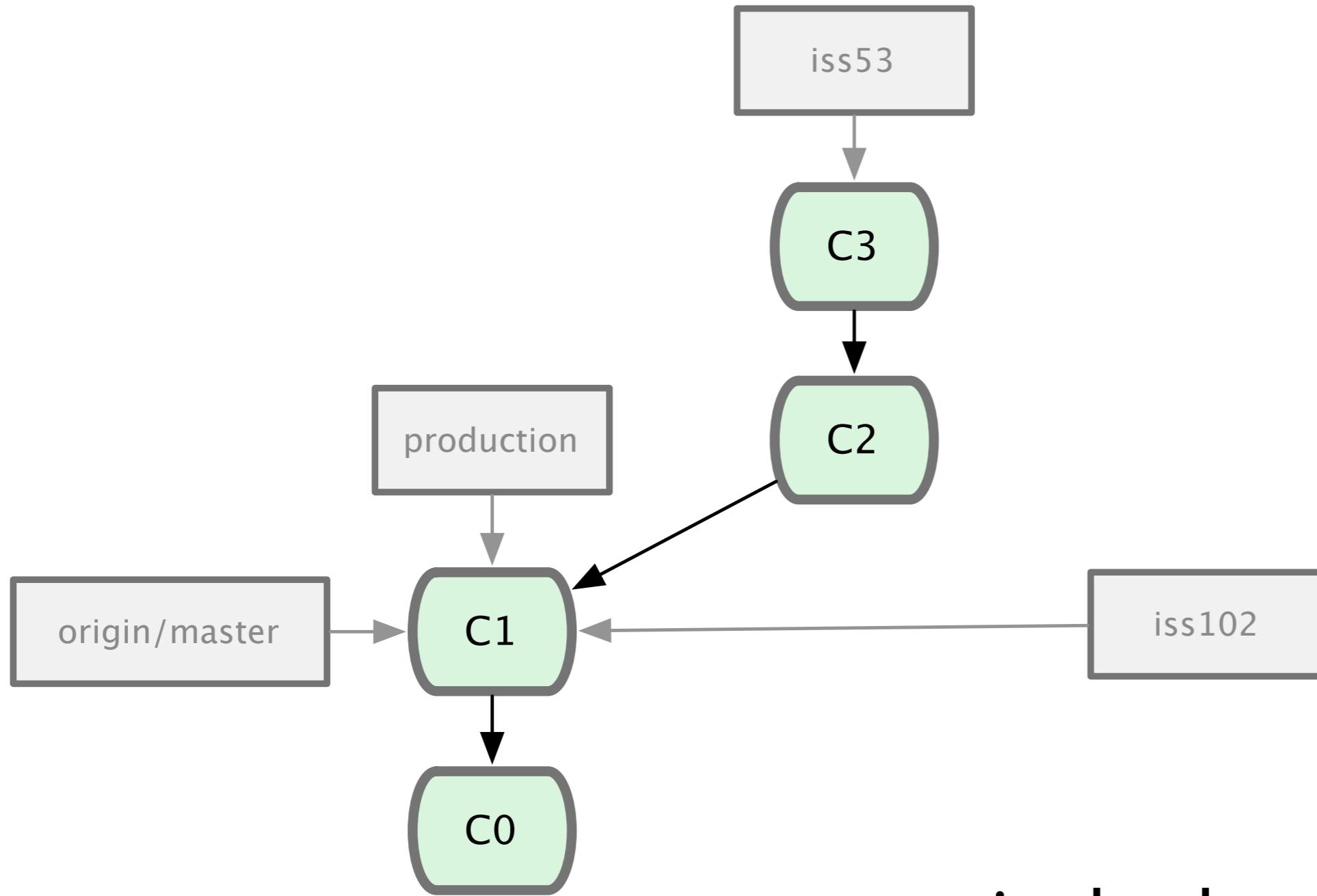
git checkout -b iss53



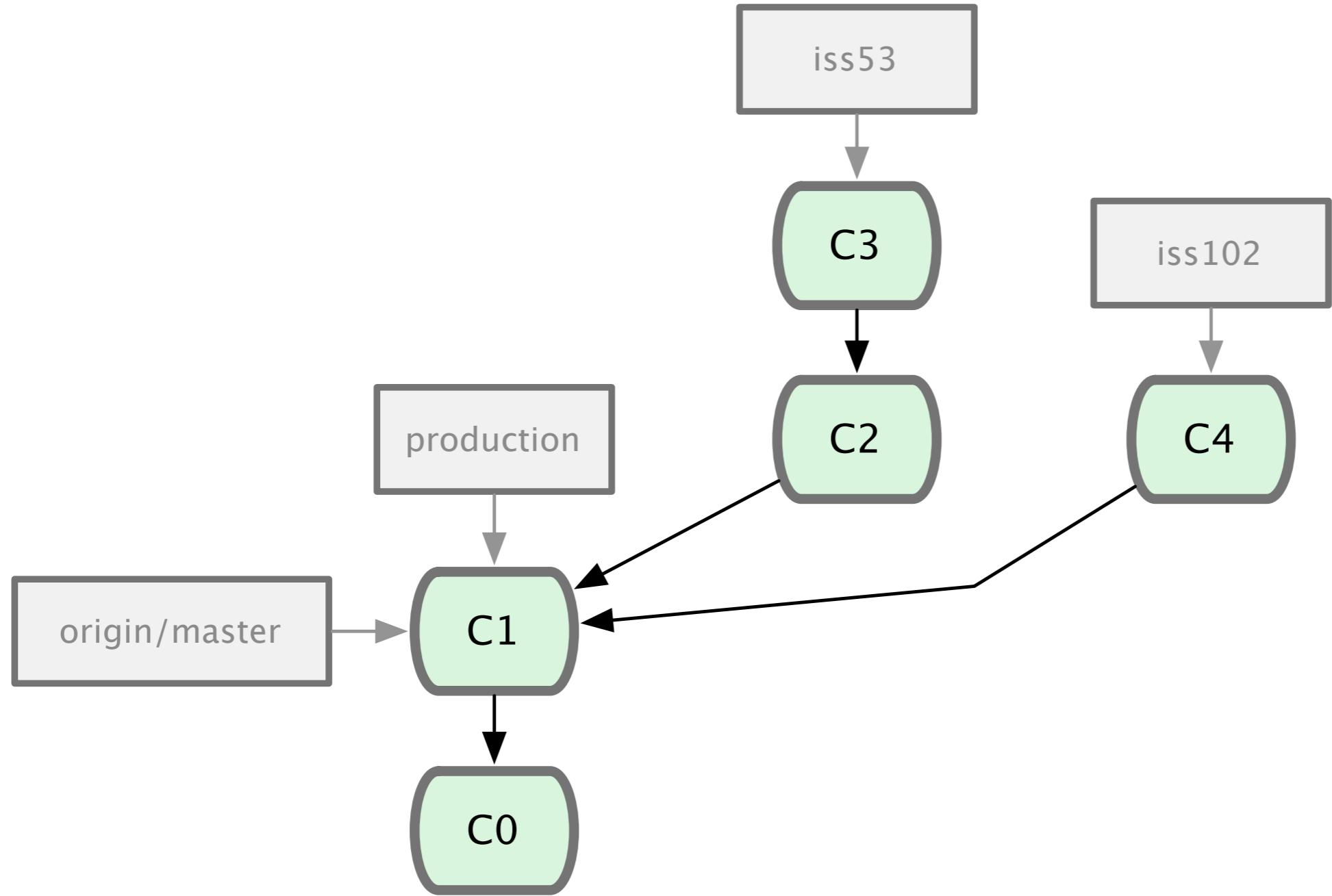
git commit



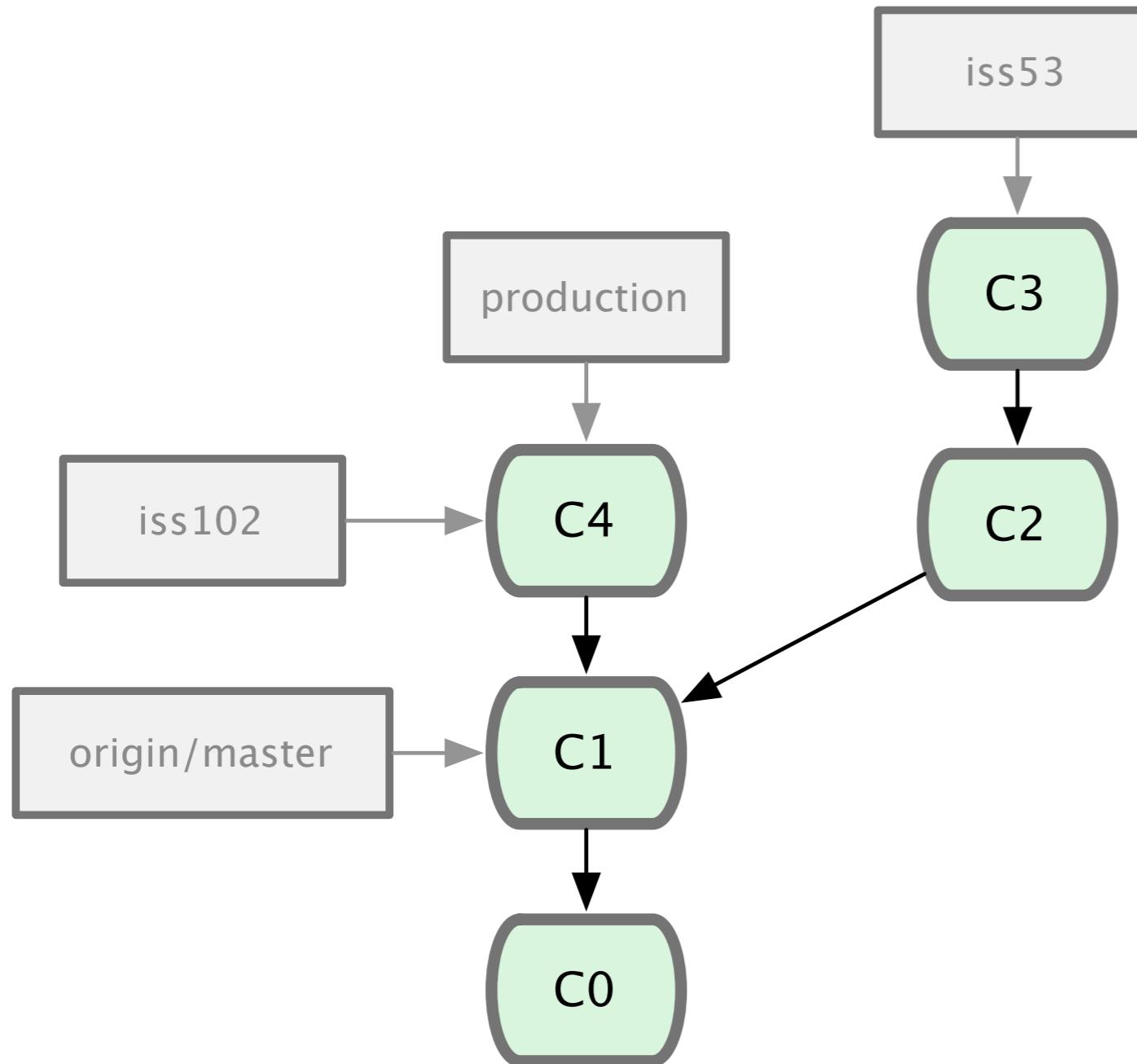
git commit



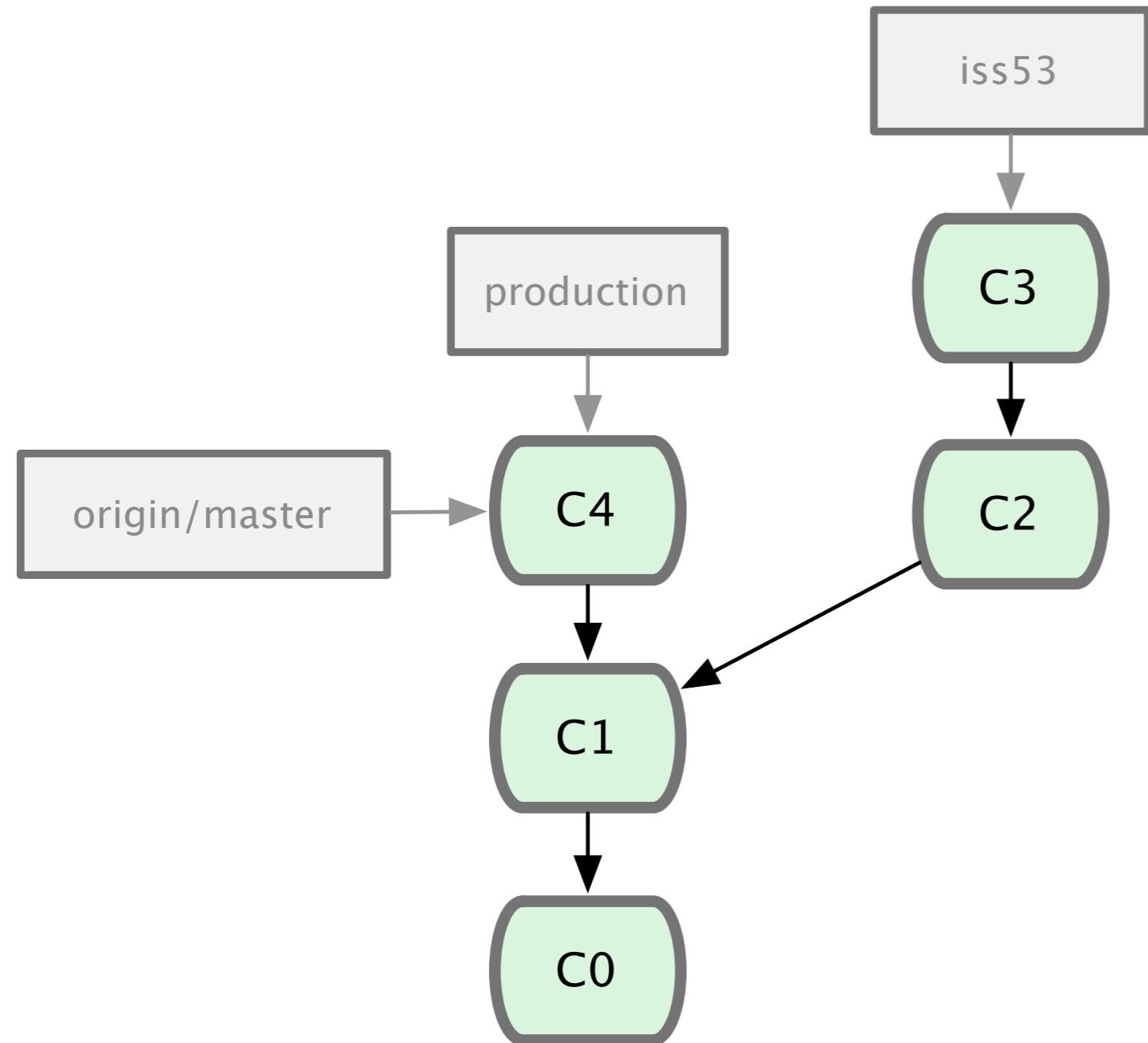
**git checkout production
git checkout -b iss102**



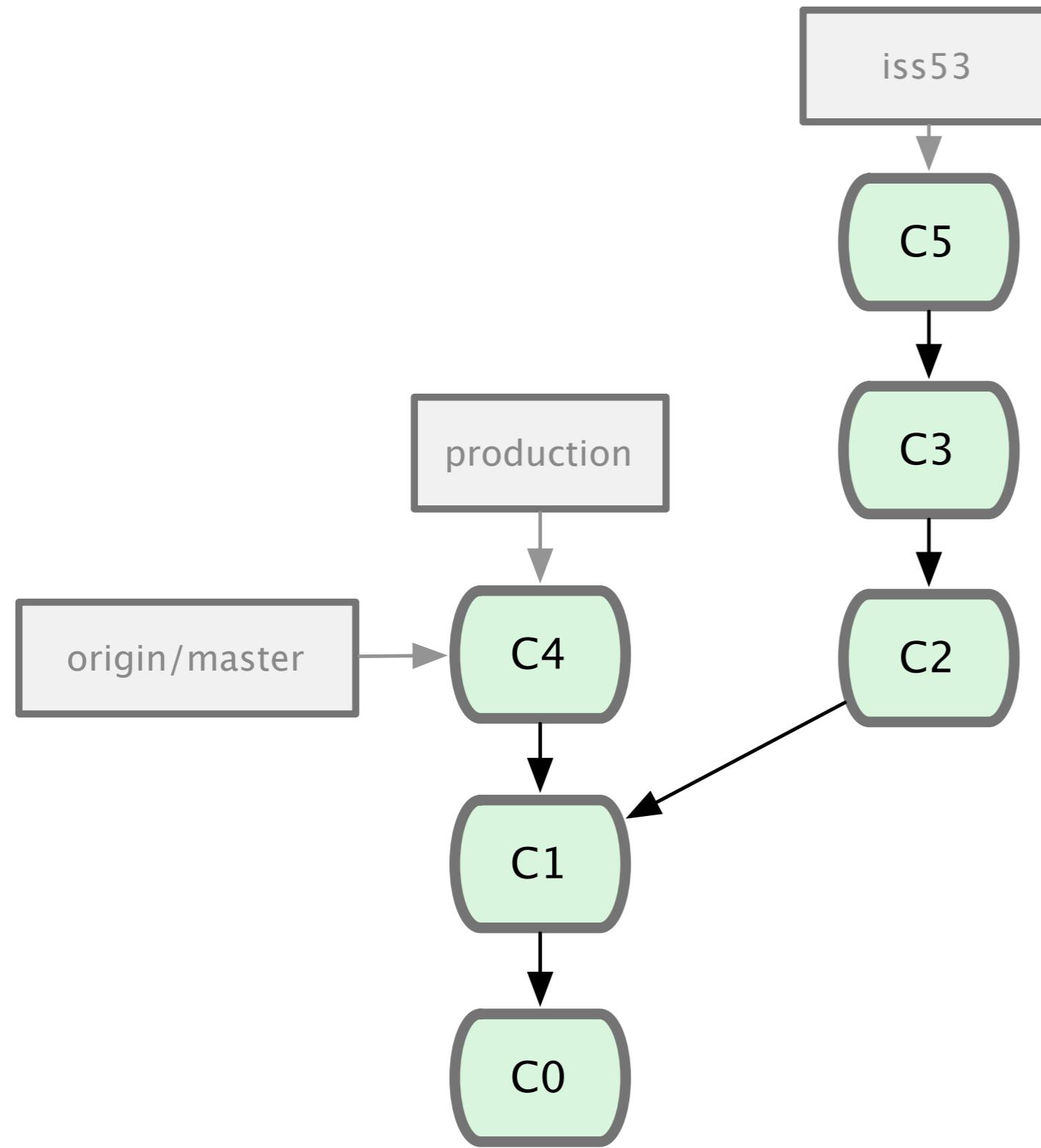
git commit



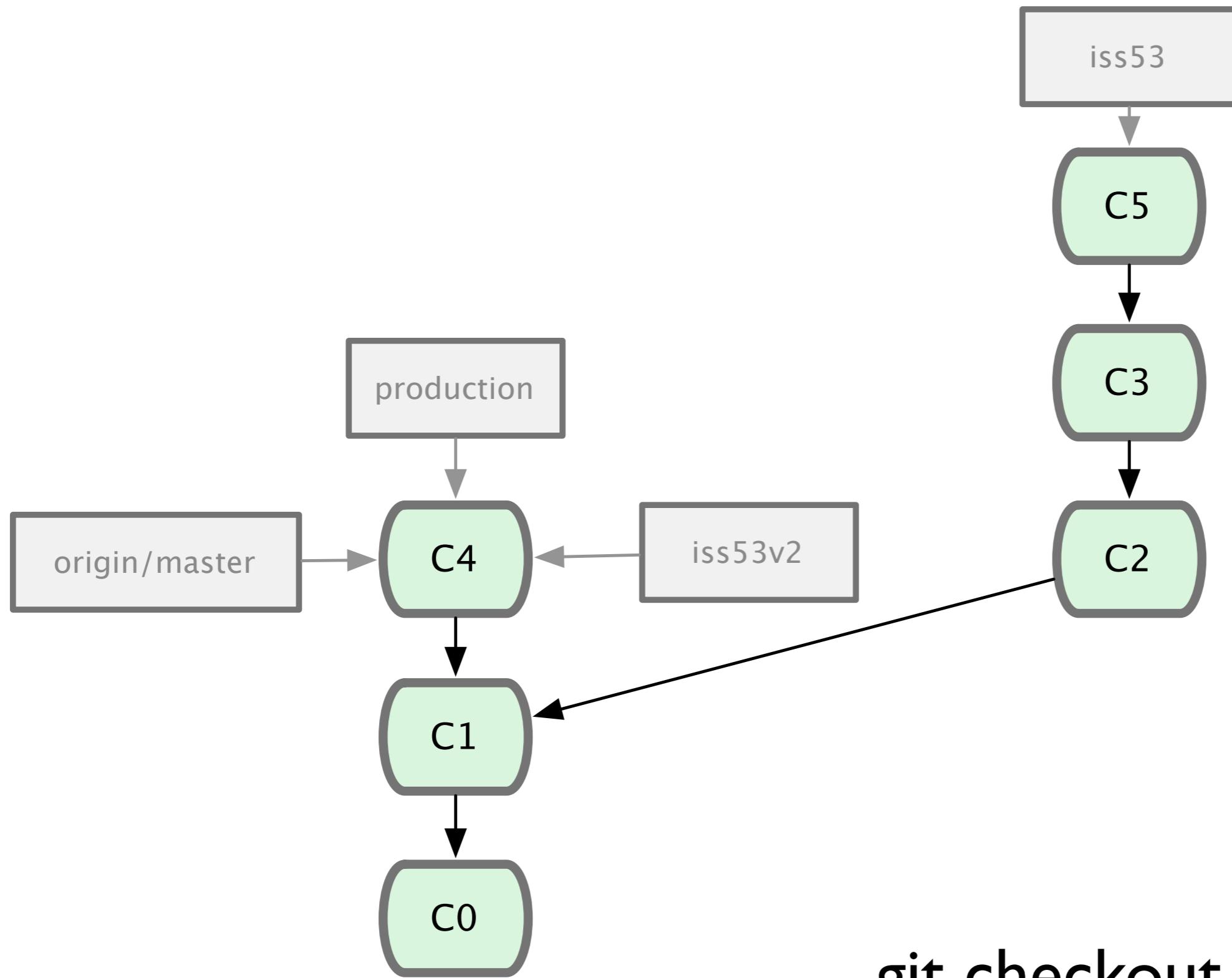
**git checkout production
git merge iss102**



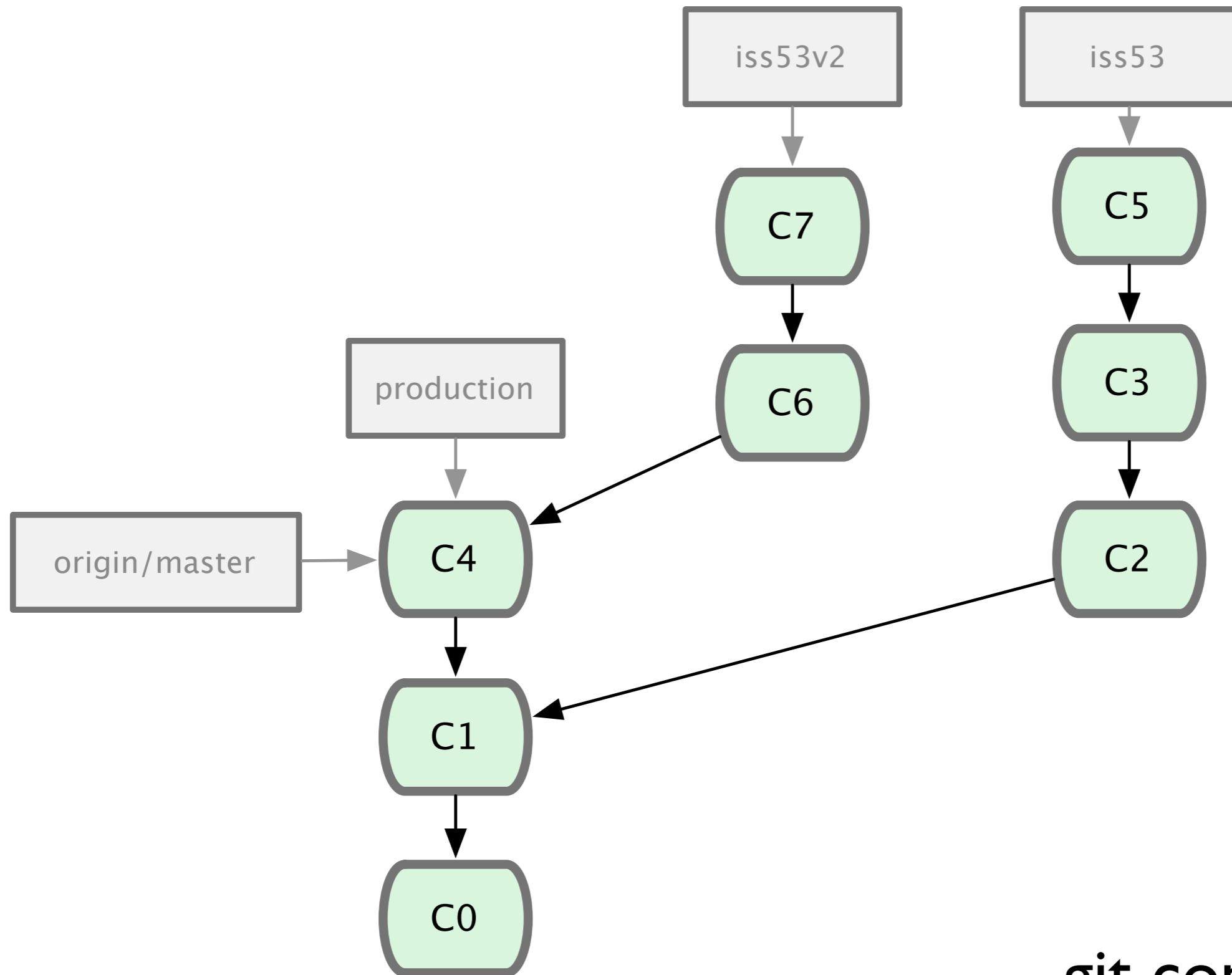
git push



**git checkout iss53
git commit**



git checkout production
git checkout -b iss53v2



git commit
git commit

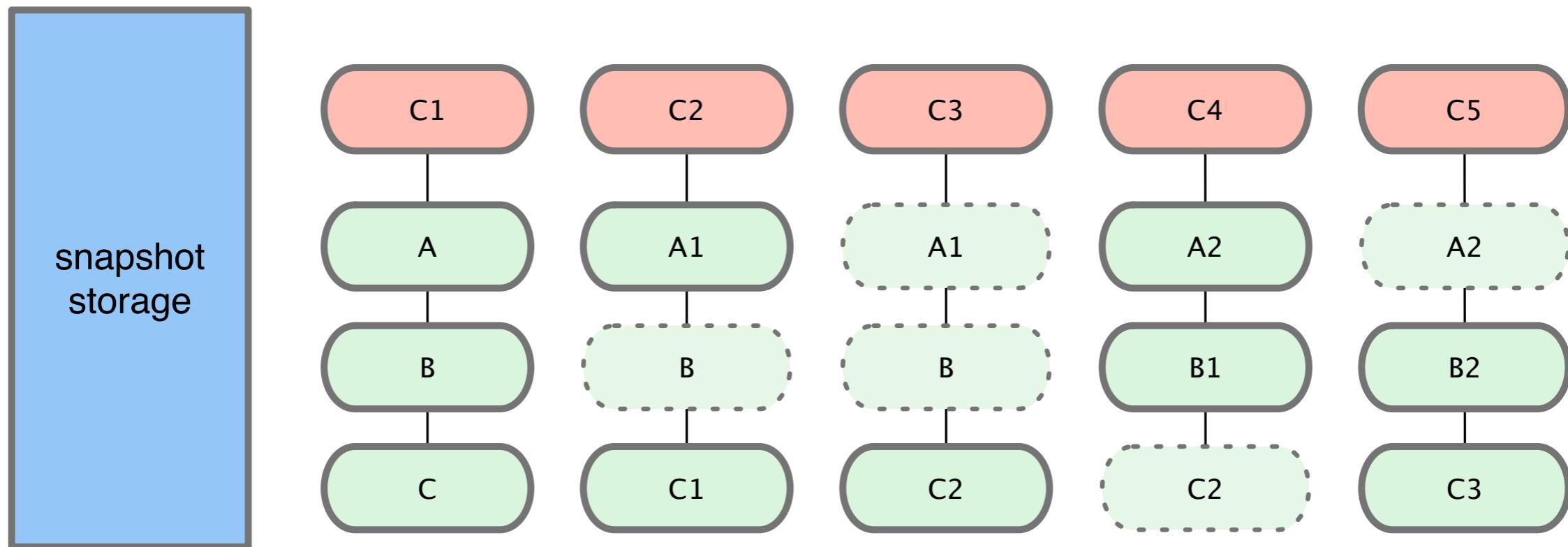
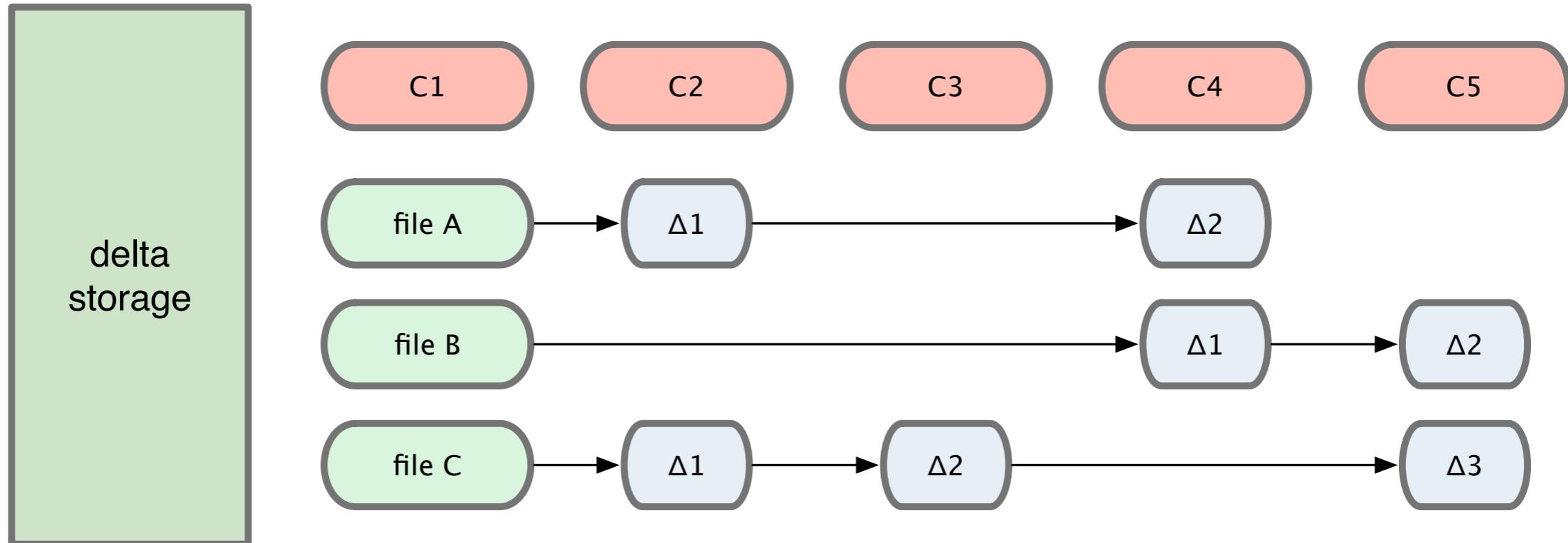
try out an idea

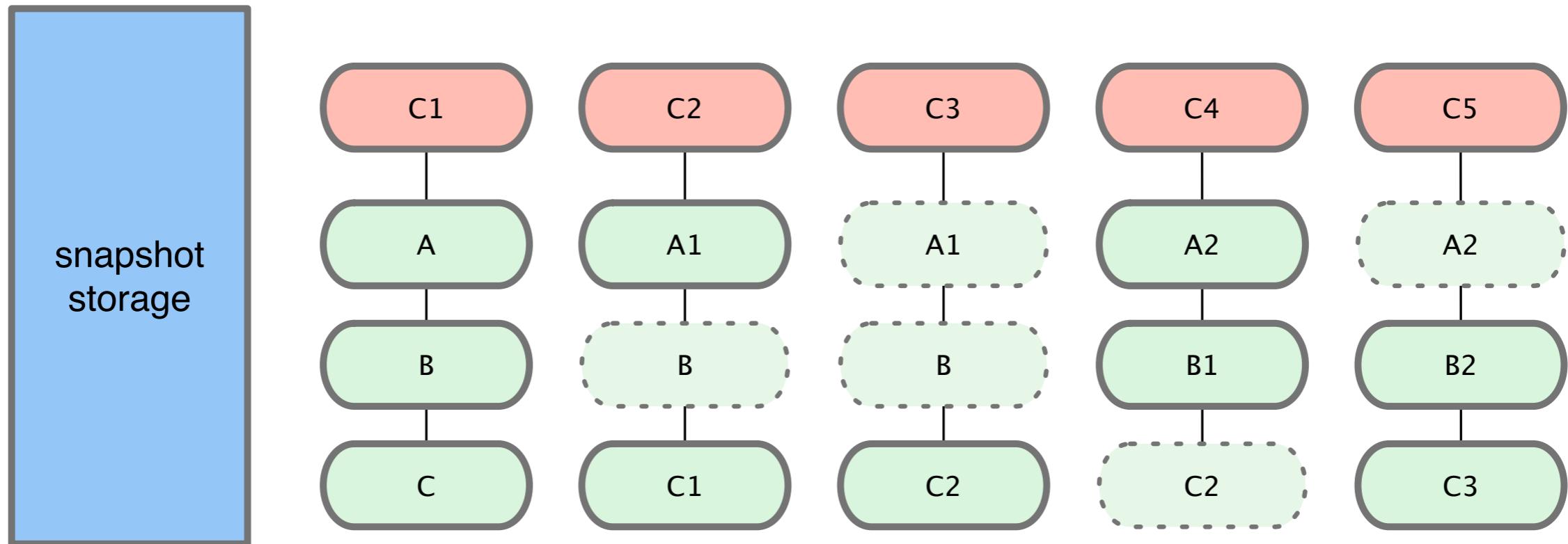
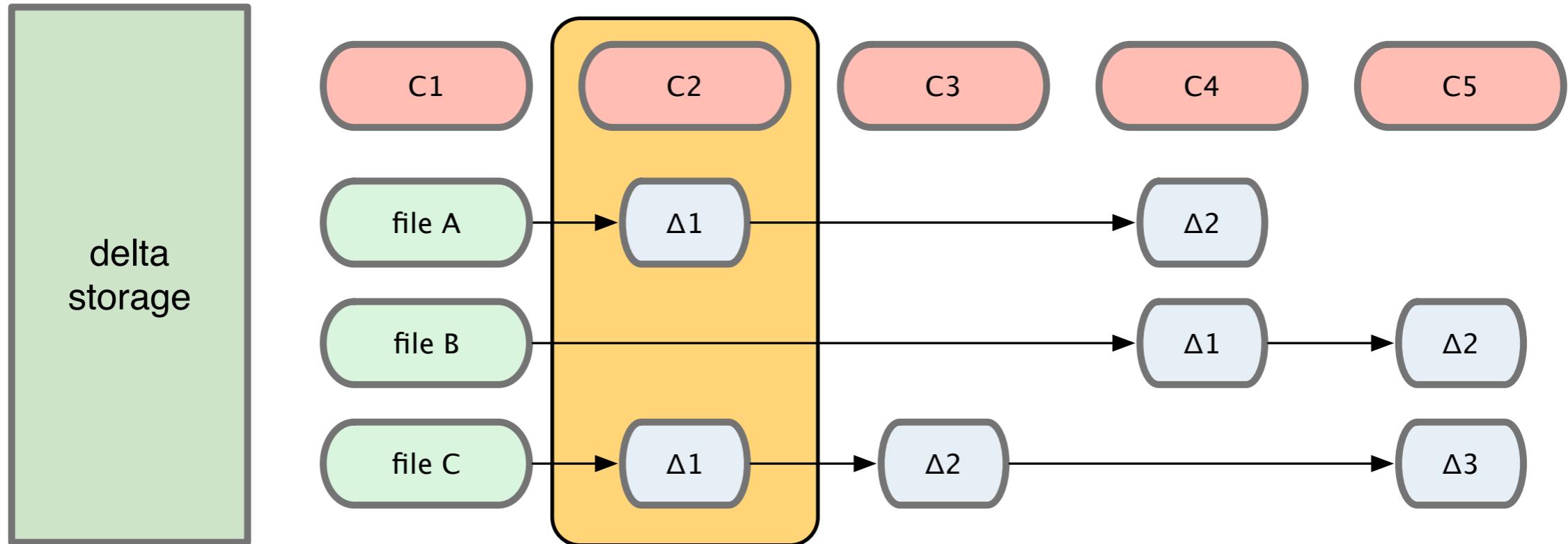
isolate work units

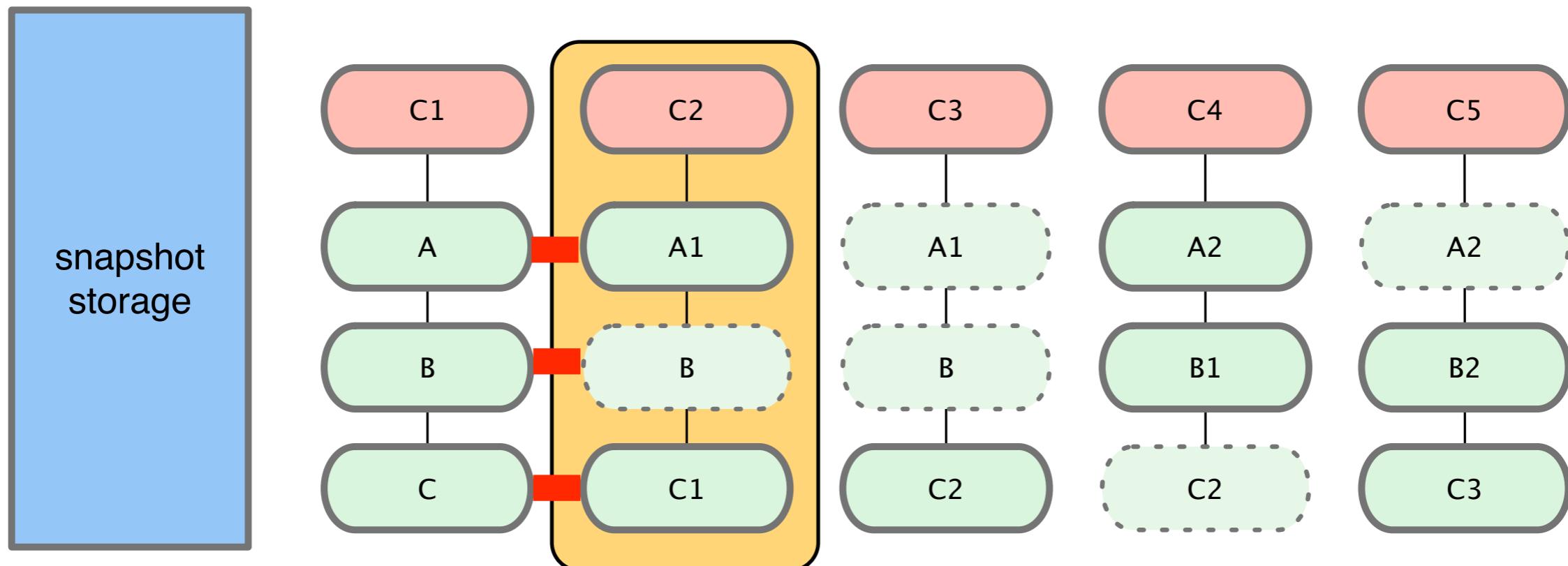
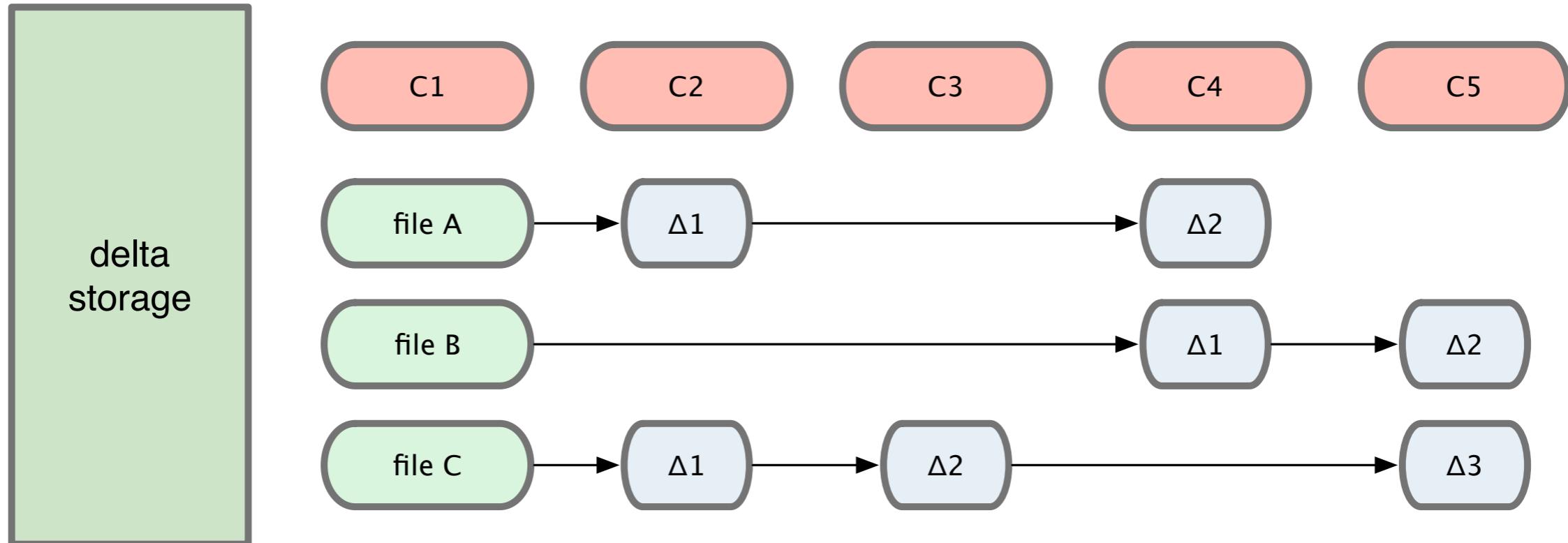
long running topics

pain free
context switching

Changes







git diff

```
diff --git a/main.py b/main.py
index 6db8b97..b9bcc62 100755
--- a/main.py
+++ b/main.py
@@ -2,10 +2,12 @@
 import wsgiref.handlers
 from google.appengine.ext import webapp

+# this program prints out 'hello world'
+
class MainHandler(webapp.RequestHandler):

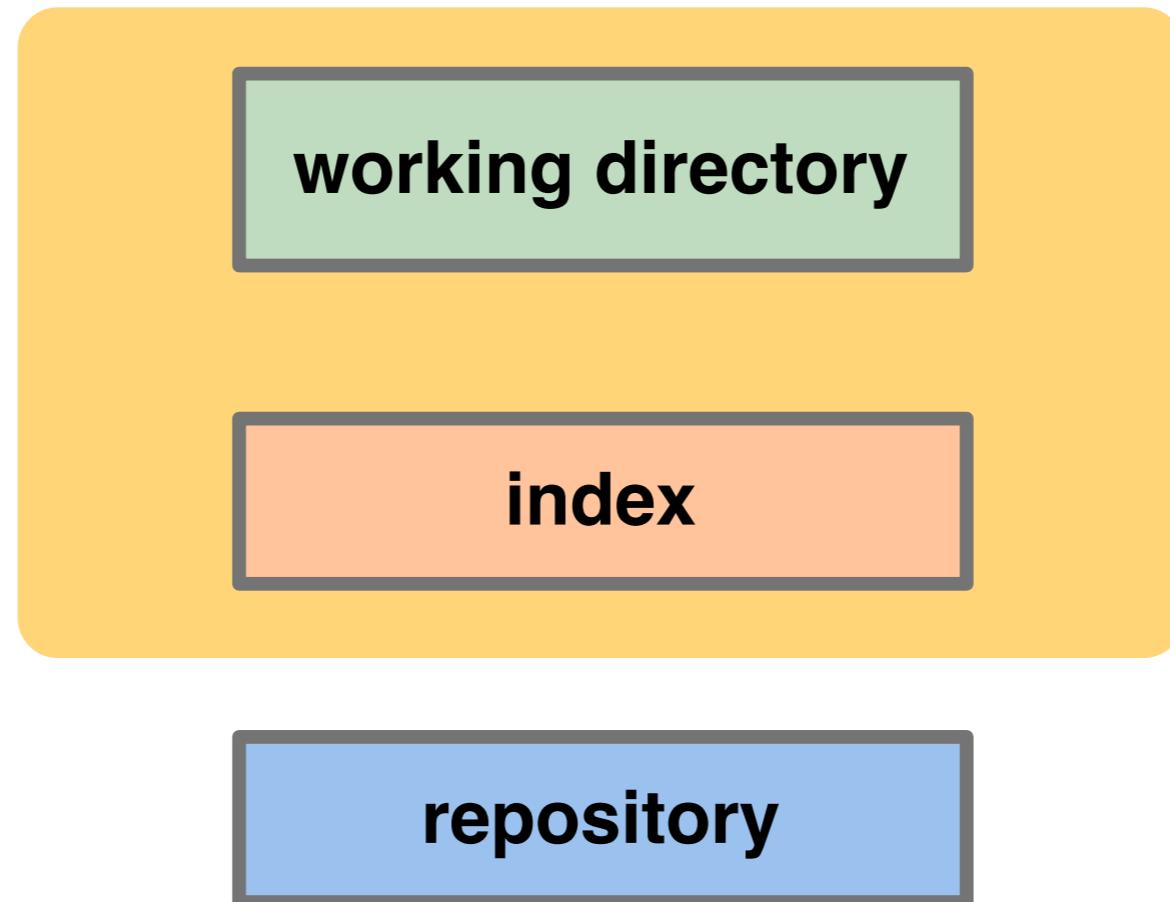
    def get(self):
-        self.response.out.write('Hello world!')
+        self.response.out.write('Hola mundo!')

def main():
    application = webapp.WSGIApplication([('/', MainHandler)],
```

What is not yet staged?

git diff

git diff



What is staged?

git diff --cached

`git diff --cached`

working directory

index

repository

Unified Diff

git diff > change.patch

`git diff > change.patch`

`patch -p1 < change.patch`

`git diff > change.patch`

`patch -p1 < change.patch`

or

`git apply change.patch`

History

git log

```
$>git log
```

```
$>git log
```

```
commit 310154e3c7db47d8bac935c2c43aee6afac11aae
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 13 10:49:15 2008 -0700
```

```
updated README formatting and added blame
```

```
commit f7f3f6dd8fd3fa40f052427c32785a0fa01aaa5f
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:45:01 2008 -0700
```

```
changed my name a bit
```

```
commit 710f0f8d2cdf5af87033b9ec08859a505f9a6af5
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:34:16 2008 -0700
```

```
added ls-files
```

```
commit c110d7ff8cfb86fd5cce9a8aee462678dbb4ef9b
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 6 12:13:36 2008 -0700
```

```
made the ls-tree function recursive and list trees
```

```
$>git log
```

```
commit 310154e3c7db47d8bac935c2c43aee6afac11aae
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 13 10:49:15 2008 -0700
```

```
updated README formatting and added blame
```

```
commit f7f3f6dd8fd3fa40f052427c32785a0fa01aaa5f
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:45:01 2008 -0700
```

```
changed my name a bit
```

```
commit 710f0f8d2cdf5af87033b9ec08859a505f9a6af5
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:34:16 2008 -0700
```

```
added ls-files
```

```
commit c110d7ff8cfb86fd5cce9a8aee462678dbb4ef9b
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 6 12:13:36 2008 -0700
```

```
made the ls-tree function recursive and list trees
```

```
$>git log
```

```
commit 310154e3c7db47d8bac935c2c43aee6afac11aae
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 13 10:49:15 2008 -0700
```

```
updated README formatting and added blame
```

```
commit f7f3f6dd8fd3fa40f052427c32785a0fa01aaa5f
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:45:01 2008 -0700
```

```
changed my name a bit
```

```
commit 710f0f8d2cdf5af87033b9ec08859a505f9a6af5
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:34:16 2008 -0700
```

```
added ls-files
```

```
commit c110d7ff8cfb86fd5cce9a8aee462678dbb4ef9b
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 6 12:13:36 2008 -0700
```

```
made the ls-tree function recursive and list trees
```

```
$>git log
```

```
commit 310154e3c7db47d8bac935c2c43aee6afac11aae
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 13 10:49:15 2008 -0700
```

updated README formatting and added blame

```
commit f7f3f6dd8fd3fa40f052427c32785a0fa01aaa5f
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:45:01 2008 -0700
```

changed my name a bit

```
commit 710f0f8d2cdf5af87033b9ec08859a505f9a6af5
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:34:16 2008 -0700
```

added ls-files

```
commit c110d7ff8cfb86fd5cce9a8aee462678dbb4ef9b
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 6 12:13:36 2008 -0700
```

made the ls-tree function recursive and list trees

```
$>git log
```

```
commit 310154e3c7db47d8bac935c2c43aee6afac11aae
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 13 10:49:15 2008 -0700
```

updated README formatting and added blame

```
commit f7f3f6dd8fd3fa40f052427c32785a0fa01aaa5f
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:45:01 2008 -0700
```

changed my name a bit

```
commit 710f0f8d2cdf5af87033b9ec08859a505f9a6af5
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:34:16 2008 -0700
```

added ls-files

```
commit c110d7ff8cfb86fd5cce9a8aee462678dbb4ef9b
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 6 12:13:36 2008 -0700
```

made the ls-tree function recursive and list trees

```
$>git log
```

```
commit 310154e3c7db47d8bac935c2c43aee6afac11aae
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 13 10:49:15 2008 -0700
```

updated README formatting and added blame

```
commit f7f3f6dd8fd3fa40f052427c32785a0fa01aaa5f
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:45:01 2008 -0700
```

changed my name a bit

```
commit 710f0f8d2cdf5af87033b9ec08859a505f9a6af5
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:34:16 2008 -0700
```

added ls-files

```
commit c110d7ff8cfb86fd5cce9a8aee462678dbb4ef9b
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 6 12:13:36 2008 -0700
```

made the ls-tree function recursive and list trees

```
$>git log
```

```
commit 310154e3c7db47d8bac935c2c43aee6afac11aae
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 13 10:49:15 2008 -0700
```

updated README formatting and added blame

```
commit f7f3f6dd8fd3fa40f052427c32785a0fa01aaa5f
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:45:01 2008 -0700
```

changed my name a bit

```
commit 710f0f8d2cdf5af87033b9ec08859a505f9a6af5
```

```
Author: Magnus Chacon <mchacon@gmail.com>
```

```
Date: Sun Apr 13 10:34:16 2008 -0700
```

added ls-files

```
commit c110d7ff8cfb86fd5cce9a8aee462678dbb4ef9b
```

```
Author: Scott Chacon <schacon@gmail.com>
```

```
Date: Sun Apr 6 12:13:36 2008 -0700
```

made the ls-tree function recursive and list trees

**What am I about to
submit?**

git log [branch] ..

```
git log origin/master..
```

```
git log origin/master..
```

what you've committed that isn't pushed yet

Git Help

git [command] --help

git help [command]

git-scm.com

git-scm.com

+++ git

git

the fast version control system

Home [About Git](#) [Documentation](#) [Download](#) [Tools & Hosting](#) [Wiki](#)

Git is...

Git is an open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Projects using Git

- [Git](#)
- [Linux Kernel](#)
- [Perl](#)

Download Git

The latest stable Git release is
v1.6.1.1
release notes (2009-01-25)

[Git Community Book](#) : The git-scm.com community-built comprehensive online book

Tutorials

Short and Sweet

The [official Git tutorial](#) is a good place to get started.

[Everyday Git](#) in 20 commands is good for a useful minimum set of commands.

The [SVN Crash Course](#) might be helpful if you're coming from the SVN world.

[Git for the lazy](#) is a great guide for beginners.

Longer, More In Depth

[Git for Designers](#) – No knowledge of version control? No problem.

[Git for Computer Scientists](#) – A quick introduction to git internals for people who are not scared by words like Directed Acyclic Graph.

The [Git User's Manual](#) is a comprehensive resource, covering a lot of Git functionality.

[Git Magic](#) – An alternative online book with the source [online](#).

[The GitHub Guides](#) – Guides on a variety of Git and GitHub related topics

The official and comprehensive [reference manual](#) comes as part of the Git package itself

[Visual Git Cheat Sheet](#) – Everyone loves to cheat

[37signals' Git Resources](#) – Useful lightweight reference to keep handy

Books

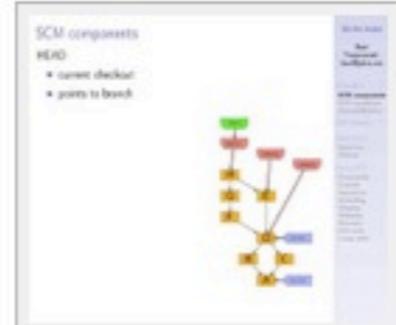


[Git Internals PDF](#)
by Scott Chacon



[Pragmatic Version Control Using Git](#)
by Travis Swicegood

Videos



 [Git Community Book](#) : The git-scm.com community-built comprehensive online book

TUTORIALS

REFERENCE

Short and Sweet

The [official Git tutorial](#) is a good place to get started.

[Everyday Git](#) in 20 commands is good for a useful minimum set of commands.

The [SVN Crash Course](#) might be helpful if you're coming from the SVN world.

[Git for the lazy](#) is a great guide for beginners.

Longer, More In Depth

[Git for Designers](#) – No knowledge of version control? No problem.

[Git for Computer Scientists](#) – A quick introduction to git internals for people who are not scared by words like Directed Acyclic Graph.

The [Git User's Manual](#) is a comprehensive resource, covering a lot of Git functionality.

[Git Magic](#) – An alternative online book with the source [online](#).

[The GitHub Guides](#) – Guides on a variety of Git and GitHub related topics

The official and comprehensive [reference manual](#) comes as part of the Git package itself

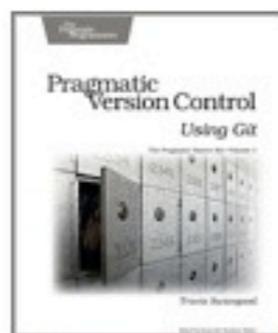
[Visual Git Cheat Sheet](#) – Everyone loves to cheat

[37signals' Git Resources](#) – Useful lightweight reference to keep handy

Books

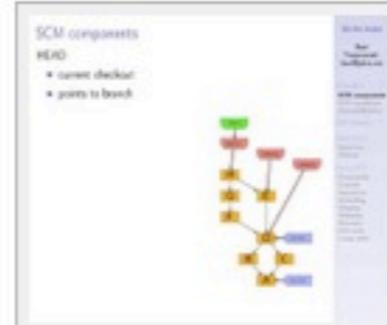


[Git Internals PDF](#)
by Scott Chacon



[Pragmatic Version Control Using Git](#)
by Travis Swicegood

Videos



[Git Community Book](#) : The git-scm.com community-built comprehensive online book

Tutorials

Short and Sweet

The [official Git tutorial](#) is a good place to get started.

[Everyday Git](#) in 20 commands is good for a useful minimum set of commands.

The [SVN Crash Course](#) might be helpful if you're coming from the SVN world.

[Git for the lazy](#) is a great guide for beginners.

Longer, More in Depth

[Git for Designers](#) – No knowledge of version control? No problem.

[Git for Computer Scientists](#) – A quick introduction to git internals for people who are not scared by words like Directed Acyclic Graph.

The [Git User's Manual](#) is a comprehensive resource, covering a lot of Git functionality.

[Git Magic](#) – An alternative online book with the source [online](#).

[The GitHub Guides](#) – Guides on a variety of Git and GitHub related topics

Reference

The official and comprehensive [reference manual](#) comes as part of the Git package itself

[Visual Git Cheat Sheet](#) – Everyone loves to cheat

[37signals' Git Resources](#) – Useful lightweight reference to keep handy

Books

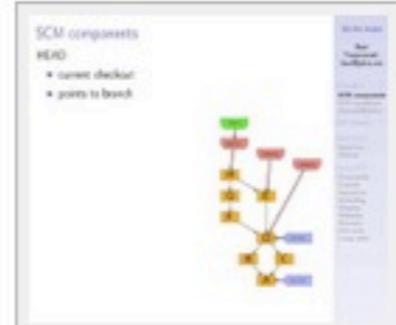


[Git Internals PDF](#)
by Scott Chacon



[Pragmatic Version Control Using Git](#)
by Travis Swicegood

Videos



[Git Community Book](#) : The git-scm.com community-built comprehensive online book

Tutorials

Short and Sweet

The [official Git tutorial](#) is a good place to get started.

[Everyday Git](#) in 20 commands is good for a useful minimum set of commands.

The [SVN Crash Course](#) might be helpful if you're coming from the SVN world.

[Git for the lazy](#) is a great guide for beginners.

Longer, More In Depth

[Git for Designers](#) – No knowledge of version control? No problem.

[Git for Computer Scientists](#) – A quick introduction to git internals for people who are not scared by words like Directed Acyclic Graph.

The [Git User's Manual](#) is a comprehensive resource, covering a lot of Git functionality.

[Git Magic](#) – An alternative online book with the source [online](#).

[The GitHub Guides](#) – Guides on a variety of Git and GitHub related topics

The official and comprehensive [reference manual](#) comes as part of the Git package itself

[Visual Git Cheat Sheet](#) – Everyone loves to cheat

[37signals' Git Resources](#) – Useful lightweight reference to keep handy

Books

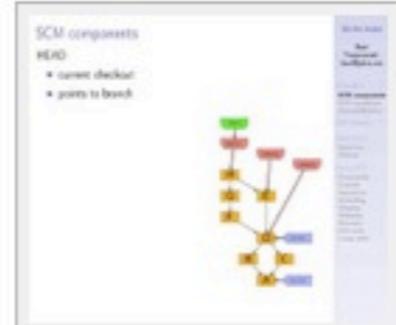


[Git Internals PDF](#)
by Scott Chacon



[Pragmatic Version Control Using Git](#)
by Travis Swicegood

Videos



[Git Community Book](#) : The git-scm.com community-built comprehensive online book

Tutorials

Short and Sweet

The [official Git tutorial](#) is a good place to get started.

[Everyday Git](#) in 20 commands is good for a useful minimum set of commands.

The [SVN Crash Course](#) might be helpful if you're coming from the SVN world.

[Git for the lazy](#) is a great guide for beginners.

Longer, More In Depth

[Git for Designers](#) – No knowledge of version control? No problem.

[Git for Computer Scientists](#) – A quick introduction to git internals for people who are not scared by words like Directed Acyclic Graph.

The [Git User's Manual](#) is a comprehensive resource, covering a lot of Git functionality.

[Git Magic](#) – An alternative online book with the source [online](#).

[The GitHub Guides](#) – Guides on a variety of Git and GitHub related topics

Reference

The official and comprehensive [reference manual](#) comes as part of the Git package itself

[Visual Git Cheat Sheet](#) – Everyone loves to cheat

[37signals' Git Resources](#) – Useful lightweight reference to keep handy

BOOKS

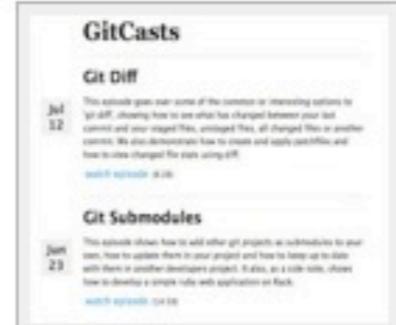
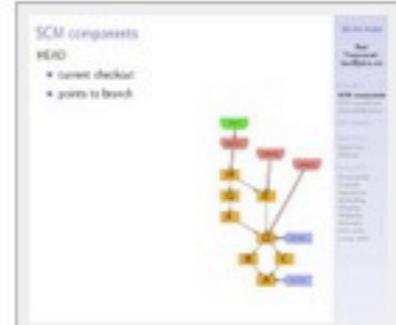


[Git Internals PDF](#)
by Scott Chacon



[Pragmatic Version Control Using Git](#)
by Travis Swicegood

Videos



 [Git Community Book](#) : The git-scm.com community-built comprehensive online book

Tutorials

Reference

Short and Sweet

The [official Git tutorial](#) is a good place to get started.

[Everyday Git](#) in 20 commands is good for a useful minimum set of commands.

The [SVN Crash Course](#) might be helpful if you're coming from the SVN world.

[Git for the lazy](#) is a great guide for beginners.

Longer, More In Depth

[Git for Designers](#) – No knowledge of version control? No problem.

[Git for Computer Scientists](#) – A quick introduction to git internals for people who are not scared by words like Directed Acyclic Graph.

The [Git User's Manual](#) is a comprehensive resource, covering a lot of Git functionality.

[Git Magic](#) – An alternative online book with the source [online](#).

The GitHub Guides – Guides on a variety of Git and GitHub related topics

The official and comprehensive [reference manual](#) comes as part of the Git package itself

[Visual Git Cheat Sheet](#) – Everyone loves to cheat

[37signals' Git Resources](#) – Useful lightweight reference to keep handy

Books

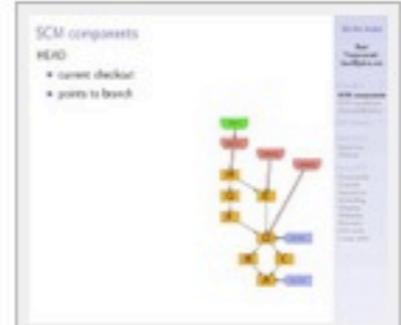


[Git Internals PDF](#)
by Scott Chacon



[Pragmatic Version Control Using Git](#)
by Travis Swicegood

Videos



[Git Community Book](#) : The git-scm.com community-built comprehensive online book

Tutorials

Short and Sweet

The [official Git tutorial](#) is a good place to get started.

[Everyday Git](#) in 20 commands is good for a useful minimum set of commands.

The [SVN Crash Course](#) might be helpful if you're coming from the SVN world.

[Git for the lazy](#) is a great guide for beginners.

Longer, More In Depth

[Git for Designers](#) – No knowledge of version control? No problem.

[Git for Computer Scientists](#) – A quick introduction to git internals for people who are not scared by words like Directed Acyclic Graph.

The [Git User's Manual](#) is a comprehensive resource, covering a lot of Git functionality.

[Git Magic](#) – An alternative online book with the source [online](#).

[The GitHub Guides](#) – Guides on a variety of Git and GitHub related topics

The official and comprehensive [reference manual](#) comes as part of the Git package itself

[Visual Git Cheat Sheet](#) – Everyone loves to cheat

[37signals' Git Resources](#) – Useful lightweight reference to keep handy

Books

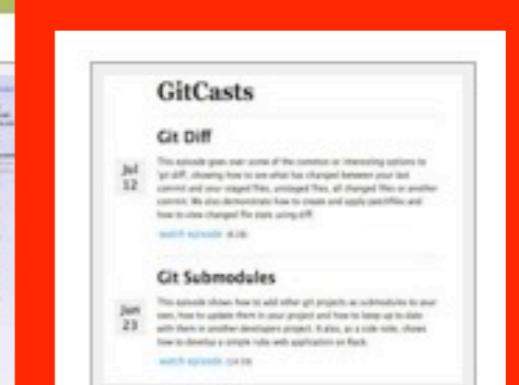
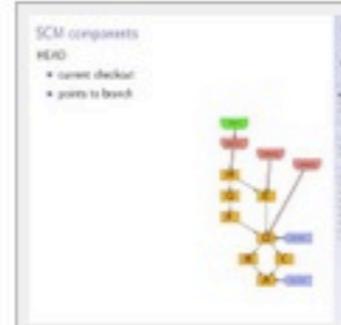


[Git Internals PDF](#)
by Scott Chacon



[Pragmatic Version Control Using Git](#)
by Travis Swicegood

Videos



Basic Commands

Review

14

git init

| 4

git init

git clone

14

git init

git clone

git add

| 4

git init

git clone

git add

git status

| 4

git init

git clone

git add

git status

git commit

| 4

git init

git clone

git add

git status

git commit

git branch

| 4

`git init`

`git clone`

`git add`

`git status`

`git commit`

`git branch`

`git checkout`

| 4

`git init`

`git merge`

`git clone`

`git add`

`git status`

`git commit`

`git branch`

`git checkout`

| 4

`git init`

`git clone`

`git add`

`git status`

`git commit`

`git branch`

`git checkout`

`git merge`

`git rebase`

| 4

`git init`

`git clone`

`git add`

`git status`

`git commit`

`git branch`

`git checkout`

`git merge`

`git rebase`

`git remote`

| 4

`git init`

`git clone`

`git add`

`git status`

`git commit`

`git branch`

`git checkout`

`git merge`

`git rebase`

`git remote`

`git fetch`

| 4

`git init`

`git clone`

`git add`

`git status`

`git commit`

`git branch`

`git checkout`

`git merge`

`git rebase`

`git remote`

`git fetch`

`git push`

| 4

`git init`

`git clone`

`git add`

`git status`

`git commit`

`git branch`

`git checkout`

`git merge`

`git rebase`

`git remote`

`git fetch`

`git push`

`git diff`

| 4

`git init`

`git clone`

`git add`

`git status`

`git commit`

`git branch`

`git checkout`

`git merge`

`git rebase`

`git remote`

`git fetch`

`git push`

`git diff`

`git log`

| 4

fun stuff

Colors

```
$ git config --global color.ui true
```

Patch Staging

git add -p

```
$ git status
```

```
$ git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes)
#
# modified:    ticgit.gemspec
#
no changes added to commit (use "git add" and/
```

```
$ git add -p
```

```
$ git add -p  
diff --git a/ticgit.gemspec b/ticgit.gemspec  
index 9c32bd4..a44667b 100644  
--- a/ticgit.gemspec  
+++ b/ticgit.gemspec  
@@ -1,7 +1,7 @@
```

```
Gem::Specification.new do |s|  
  s.platform = Gem::Platform::RUBY  
  s.name = "ticgit"  
-  s.version = "0.3.5"  
+  s.version = "0.3.6"  
  s.date = "2008-05-10"  
  s.author = "Scott Chacon"  
  s.email = "schacon@gmail.com"
```

```
Stage this hunk [y,n,a,d/,j,J,g,e,?]?
```

```
$ git add -p
diff --git a/ticgit.gemspec b/ticgit.gemspec
index 9c32bd4..a44667b 100644
--- a/ticgit.gemspec
+++ b/ticgit.gemspec
@@ -1,7 +1,7 @@
Gem::Specification.new do |s|
  s.platform = Gem::Platform::RUBY
  s.name = "ticgit"
-  s.version = "0.3.5"
+  s.version = "0.3.6"
  s.date = "2008-05-10"
  s.author = "Scott Chacon"
  s.email = "schacon@gmail.com"
```

Stage this hunk [y,n,a,d/,j,J,g,e,?]?

```
$ git add -p
diff --git a/ticgit.gemspec b/ticgit.gemspec
index 9c32bd4..a44667b 100644
--- a/ticgit.gemspec
+++ b/ticgit.gemspec
@@ -1,7 +1,7 @@
Gem::Specification.new do |s|
  s.platform = Gem::Platform::RUBY
  s.name = "ticgit"
-  s.version = "0.3.5"
+  s.version = "0.3.6"
  s.date = "2008-05-10"
  s.author = "Scott Chacon"
  s.email = "schacon@gmail.com"
Stage this hunk [y,n,a,d/,j,J,g,e,?]?
```

y

```
@@ -9,9 +9,10 @@ Gem::Specification.new do |s|
  s.files      = ["lib/ticgit/base.rb", "lib/ticgit",
"lib/ticgit/comment.rb", "lib/ticgit/ticket.rb", "lib/
ticgit.rb", "bin(ti", "bin/ticgitweb"]

  s.bindir = 'bin'
-
-  s.executables << "ti"
-
-  s.executables << "ticgitweb"
+
+  s.executables = ["ti", "ticgitweb"]
+
+  s.default_executable = %q{ti}
  s.homepage = "http://github/schacon/ticgit"

  s.require_paths = ["lib", "bin"]
+
+  s.specification_version = 2 if
s.respond_to? :specification_version=
end
```

Stage this hunk [y,n,a,d,/ ,K,g,s,e,?]?

```
@@ -9,9 +9,10 @@ Gem::Specification.new do |s|
  s.files      = ["lib/ticgit/base.rb", "lib/ticgit",
"lib/ticgit/comment.rb", "lib/ticgit/ticket.rb", "lib/
ticgit.rb", "bin(ti", "bin/ticgitweb"]

  s.bindir = 'bin'
-
-  s.executables << "ti"
-  s.executables << "ticgitweb"
+
+  s.executables = ["ti", "ticgitweb"]
+  s.default_executable = %q{ti}
  s.homepage = "http://github/schacon/ticgit"

  s.require_paths = ["lib", "bin"]
+
+  s.specification_version = 2 if
s.respond_to? :specification_version=
end
```

Stage this hunk [y,n,a,d/,K,g,s,e,?]?

```
@@ -9,9 +9,10 @@ Gem::Specification.new do |s|
  s.files      = ["lib/ticgit/base.rb", "lib/ticgit",
"lib/ticgit/comment.rb", "lib/ticgit/ticket.rb", "lib/
ticgit.rb", "bin(ti", "bin/ticgitweb"]

  s.bindir = 'bin'
-
-  s.executables << "ti"
-  s.executables << "ticgitweb"
+
+  s.executables = ["ti", "ticgitweb"]
+  s.default_executable = %q{ti}
  s.homepage = "http://github/schacon/ticgit"

  s.require_paths = ["lib", "bin"]
+
+  s.specification_version = 2 if
s.respond_to? :specification_version=
end
```

Stage this hunk [y,n,a,d/,K,g,s,e,?]?



```
$ git status
```

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:    ticgit.gemspec
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes)
#
# modified:    ticgit.gemspec
#
```

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:    ticgit.gemspec
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes)
#
# modified:    ticgit.gemspec
#
```



```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:    ticgit.gemspec
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes)
#
# modified:    ticgit.gemspec
```



partially stage files

git add -p

Annotation

git blame

git blame

ie: “what dumbass did this? oh, it was me...”

git blame daemon.c

979e32fa (Randal L. Schwartz	2005-10-25 16:29:09 -0700	1) #include "cache.h"
85023577 (Junio C Hamano	2006-12-19 14:34:12 -0800	2) #include "pkt-line.h"
77cb17e9 (Michal Ostrowski	2006-01-10 21:12:17 -0500	3) #include "exec_cmd.h"
49ba83fb (Jon Loeliger	2006-09-19 20:31:51 -0500	4) #include " interpolate.h"
f8ff0c06 (Petr Baudis	2005-09-22 11:25:28 +0200	5)
85023577 (Junio C Hamano	2006-12-19 14:34:12 -0800	6) #include <syslog.h>
85023577 (Junio C Hamano	2006-12-19 14:34:12 -0800	7)
695dffe2 (Johannes Schindelin	2006-09-28 12:00:35 +0200	8) #ifndef HOST_NAME_MAX
695dffe2 (Johannes Schindelin	2006-09-28 12:00:35 +0200	9) #define HOST_NAME_MAX 256
695dffe2 (Johannes Schindelin	2006-09-28 12:00:35 +0200	10) #endif
695dffe2 (Johannes Schindelin	2006-09-28 12:00:35 +0200	11)
415e7b87 (Patrick Welche	2007-10-18 18:17:39 +0100	12) #ifndef NI_MAXSERV
415e7b87 (Patrick Welche	2007-10-18 18:17:39 +0100	13) #define NI_MAXSERV 32
415e7b87 (Patrick Welche	2007-10-18 18:17:39 +0100	14) #endif
415e7b87 (Patrick Welche	2007-10-18 18:17:39 +0100	15)
9048fe1c (Petr Baudis	2005-09-24 16:13:01 +0200	16) static int log_syslog;
f8ff0c06 (Petr Baudis	2005-09-22 11:25:28 +0200	17) static int verbose;
1955fabf (Mark Wooding	2006-02-03 20:27:04 +0000	18) static int reuseaddr;
f8ff0c06 (Petr Baudis	2005-09-22 11:25:28 +0200	19)
960deccb (H. Peter Anvin	2005-10-19 14:27:01 -0700	20) static const char daemon_usage[] =
1b1dd23f (Stephan Beyer	2008-07-13 15:36:15 +0200	21) "git daemon [--verbose] [--syslog] [
3bd62c21 (Stephen R. van den Berg	2008-08-14 20:02:20 +0200	22) " --timeout=n] [--init-ti
3bd62c21 (Stephen R. van den Berg	2008-08-14 20:02:20 +0200	23) " --strict-paths] [--base
73a7a656 (Jens Axboe	2007-07-27 14:00:29 -0700	24) " --user-path --user-pa
49ba83fb (Jon Loeliger	2006-09-19 20:31:51 -0500	25) " --interpolated-path=pat
678dac6b (Tilman Sauerbeck	2006-08-22 19:37:41 +0200	26) " --reuseaddr] [--detach]
d9edcbd6 (Junio C Hamano	2006-09-07 01:40:04 -0700	27) " --[enable disable allow
dd467629 (Jon Loeliger	2006-09-26 09:47:43 -0500	28) " --inetd [--listen=hos
dd467629 (Jon Loeliger	2006-09-26 09:47:43 -0500	29) " --user=user
dd467629 (Jon Loeliger	2006-09-26 09:47:43 -0500	30) " [directory...]";
4ae95682 (H. Peter Anvin	2005-09-26 19:10:55 -0700	31)
4ae95682 (H. Peter Anvin	2005-09-26 19:10:55 -0700	32) /* List of acceptable pathname prefi
96f1e58f (David Rienties	2006-08-15 10:23:48 -0700	33) static char **ok_paths:

git blame daemon.c

979e32fa	(Randal L. Schwartz	2005-10-25 16:29:09 -0700	1) #include "cache.h"
85023577	(Junio C Hamano	2006-12-19 14:34:12 -0800	2) #include "pkt-line.h"
77cb17e9	(Michal Ostrowski	2006-01-10 21:12:17 -0500	3) #include "exec_cmd.h"
49ba83fb	(Jon Loeliger	2006-09-19 20:31:51 -0500	4) #include " interpolate.h"
f8ff0c06	(Petr Baudis	2005-09-22 11:25:28 +0200	5)
85023577	(Junio C Hamano	2006-12-19 14:34:12 -0800	6) #include <syslog.h>
85023577	(Junio C Hamano	2006-12-19 14:34:12 -0800	7)
695dffe2	(Johannes Schindelin	2006-09-28 12:00:35 +0200	8) #ifndef HOST_NAME_MAX
695dffe2	(Johannes Schindelin	2006-09-28 12:00:35 +0200	9) #define HOST_NAME_MAX 256
695dffe2	(Johannes Schindelin	2006-09-28 12:00:35 +0200	10) #endif
695dffe2	(Johannes Schindelin	2006-09-28 12:00:35 +0200	11)
415e7b87	(Patrick Welche	2007-10-18 18:17:39 +0100	12) #ifndef NI_MAXSERV
415e7b87	(Patrick Welche	2007-10-18 18:17:39 +0100	13) #define NI_MAXSERV 32
415e7b87	(Patrick Welche	2007-10-18 18:17:39 +0100	14) #endif
415e7b87	(Patrick Welche	2007-10-18 18:17:39 +0100	15)
9048fe1c	(Petr Baudis	2005-09-24 16:13:01 +0200	16) static int log_syslog;
f8ff0c06	(Petr Baudis	2005-09-22 11:25:28 +0200	17) static int verbose;
1955fabf	(Mark Wooding	2006-02-03 20:27:04 +0000	18) static int reuseaddr;
f8ff0c06	(Petr Baudis	2005-09-22 11:25:28 +0200	19)
960deccb	(H. Peter Anvin	2005-10-19 14:27:01 -0700	20) static const char daemon_usage[] =
1b1dd23f	(Stephan Beyer	2008-07-13 15:36:15 +0200	21) "git daemon [--verbose] [--syslog] [
3bd62c21	(Stephen R. van den Berg	2008-08-14 20:02:20 +0200	22) " --timeout=n] [--init-ti
3bd62c21	(Stephen R. van den Berg	2008-08-14 20:02:20 +0200	23) " --strict-paths] [--base
73a7a656	(Jens Axboe	2007-07-27 14:00:29 -0700	24) " --user-path --user-pa
49ba83fb	(Jon Loeliger	2006-09-19 20:31:51 -0500	25) " --interpolated-path=pat
678dac6b	(Tilman Sauerbeck	2006-08-22 19:37:41 +0200	26) " --reuseaddr] [--detach]
d9edcbd6	(Junio C Hamano	2006-09-07 01:40:04 -0700	27) " --[enable disable allow
dd467629	(Jon Loeliger	2006-09-26 09:47:43 -0500	28) " --inetd [--listen=hos
dd467629	(Jon Loeliger	2006-09-26 09:47:43 -0500	29) " [--user=user
dd467629	(Jon Loeliger	2006-09-26 09:47:43 -0500	30) " [directory...]";
4ae95682	(H. Peter Anvin	2005-09-26 19:10:55 -0700	31)
4ae95682	(H. Peter Anvin	2005-09-26 19:10:55 -0700	32) /* List of acceptable pathname prefi
96f1e58f	(David Rientjes	2006-08-15 10:23:48 -0700	33) static char **ok_paths:

git blame daemon.c

979e32fa	(Randal L. Schwartz	2005-10-25 16:29:09 -0700	1) #include "cache.h"
85023577	(Junio C Hamano	2006-12-19 14:34:12 -0800	2) #include "pkt-line.h"
77cb17e9	(Michał Ostrowski	2006-01-10 21:12:17 -0500	3) #include "exec_cmd.h"
49ba83fb	(Jon Loeliger	2006-09-19 20:31:51 -0500	4) #include " interpolate.h"
f8ff0c06	(Petr Baudis	2005-09-22 11:25:28 +0200	5)
85023577	(Junio C Hamano	2006-12-19 14:34:12 -0800	6) #include <syslog.h>
85023577	(Junio C Hamano	2006-12-19 14:34:12 -0800	7)
695dffe2	(Johannes Schindelin	2006-09-28 12:00:35 +0200	8) #ifndef HOST_NAME_MAX
695dffe2	(Johannes Schindelin	2006-09-28 12:00:35 +0200	9) #define HOST_NAME_MAX 256
695dffe2	(Johannes Schindelin	2006-09-28 12:00:35 +0200	10) #endif
695dffe2	(Johannes Schindelin	2006-09-28 12:00:35 +0200	11)
415e7b87	(Patrick Welche	2007-10-18 18:17:39 +0100	12) #ifndef NI_MAXSERV
415e7b87	(Patrick Welche	2007-10-18 18:17:39 +0100	13) #define NI_MAXSERV 32
415e7b87	(Patrick Welche	2007-10-18 18:17:39 +0100	14) #endif
415e7b87	(Patrick Welche	2007-10-18 18:17:39 +0100	15)
9048fe1c	(Petr Baudis	2005-09-24 16:13:01 +0200	16) static int log_syslog;
f8ff0c06	(Petr Baudis	2005-09-22 11:25:28 +0200	17) static int verbose;
1955fabf	(Mark Wooding	2006-02-03 20:27:04 +0000	18) static int reuseaddr;
f8ff0c06	(Petr Baudis	2005-09-22 11:25:28 +0200	19)
960deccb	(H. Peter Anvin	2005-10-19 14:27:01 -0700	20) static const char daemon_usage[] =
1b1dd23f	(Stephan Beyer	2008-07-13 15:36:15 +0200	21) "git daemon [--verbose] [--syslog] [
3bd62c21	(Stephen R. van den Berg	2008-08-14 20:02:20 +0200	22) " --timeout=n] [--init-ti
3bd62c21	(Stephen R. van den Berg	2008-08-14 20:02:20 +0200	23) " --strict-paths] [--base
73a7a656	(Jens Axboe	2007-07-27 14:00:29 -0700	24) " --user-path --user-pa
49ba83fb	(Jon Loeliger	2006-09-19 20:31:51 -0500	25) " --interpolated-path=pat
678dac6b	(Tilman Sauerbeck	2006-08-22 19:37:41 +0200	26) " --reuseaddr] [--detach]
d9edcbd6	(Junio C Hamano	2006-09-07 01:40:04 -0700	27) " --[enable disable allow
dd467629	(Jon Loeliger	2006-09-26 09:47:43 -0500	28) " --inetd [--listen=hos
dd467629	(Jon Loeliger	2006-09-26 09:47:43 -0500	29) " --user=user
dd467629	(Jon Loeliger	2006-09-26 09:47:43 -0500	30) " [directory...]";
4ae95682	(H. Peter Anvin	2005-09-26 19:10:55 -0700	31)
4ae95682	(H. Peter Anvin	2005-09-26 19:10:55 -0700	32) /* List of acceptable pathname prefi
96f1e58f	(David Rientjes	2006-08-15 10:23:48 -0700	33) static char **ok_paths:

git blame daemon.c

```
979e32fa (Randal L. Schwartz
85023577 (Junio C Hamano
77cb17e9 (Michal Ostrowski
49ba83fb (Jon Loeliger
f8ff0c06 (Petr Baudis
85023577 (Junio C Hamano
85023577 (Junio C Hamano
695dffe2 (Johannes Schindelin
695dffe2 (Johannes Schindelin
695dffe2 (Johannes Schindelin
695dffe2 (Johannes Schindelin
415e7b87 (Patrick Welche
415e7b87 (Patrick Welche
415e7b87 (Patrick Welche
415e7b87 (Patrick Welche
9048fe1c (Petr Baudis
f8ff0c06 (Petr Baudis
1955fabf (Mark Wooding
f8ff0c06 (Petr Baudis
960deccb (H. Peter Anvin
1b1dd23f (Stephan Beyer
3bd62c21 (Stephen R. van den Berg
3bd62c21 (Stephen R. van den Berg
73a7a656 (Jens Axboe
49ba83fb (Jon Loeliger
678dac6b (Tilman Sauerbeck
d9edcbd6 (Junio C Hamano
dd467629 (Jon Loeliger
dd467629 (Jon Loeliger
dd467629 (Jon Loeliger
4ae95682 (H. Peter Anvin
4ae95682 (H. Peter Anvin
96f1e58f (David Rientjes
```

```
2005-10-25 16:29:09 -0700
2006-12-19 14:34:12 -0800
2006-01-10 21:12:17 -0500
2006-09-19 20:31:51 -0500
2005-09-22 11:25:28 +0200
2006-12-19 14:34:12 -0800
2006-12-19 14:34:12 -0800
2006-09-28 12:00:35 +0200
2006-09-28 12:00:35 +0200
2006-09-28 12:00:35 +0200
2006-09-28 12:00:35 +0200
2007-10-18 18:17:39 +0100
2007-10-18 18:17:39 +0100
2007-10-18 18:17:39 +0100
2007-10-18 18:17:39 +0100
2005-09-24 16:13:01 +0200
2005-09-22 11:25:28 +0200
2006-02-03 20:27:04 +0000
2005-09-22 11:25:28 +0200
2005-10-19 14:27:01 -0700
2008-07-13 15:36:15 +0200
2008-08-14 20:02:20 +0200
2008-08-14 20:02:20 +0200
2007-07-27 14:00:29 -0700
2006-09-19 20:31:51 -0500
2006-08-22 19:37:41 +0200
2006-09-07 01:40:04 -0700
2006-09-26 09:47:43 -0500
2006-09-26 09:47:43 -0500
2006-09-26 09:47:43 -0500
2005-09-26 19:10:55 -0700
2005-09-26 19:10:55 -0700
2006-08-15 10:23:48 -0700
```

```
1) #include "cache.h"
2) #include "pkt-line.h"
3) #include "exec_cmd.h"
4) #include " interpolate.h"
5)
6) #include <syslog.h>
7)
8) #ifndef HOST_NAME_MAX
9) #define HOST_NAME_MAX 256
10) #endif
11)
12) #ifndef NI_MAXSERV
13) #define NI_MAXSERV 32
14) #endif
15)
16) static int log_syslog;
17) static int verbose;
18) static int reuseaddr;
19)
20) static const char daemon_usage[] =
21) "git daemon [--verbose] [--syslog] [
22) " --timeout=n ] [--init-ti
23) " --strict-paths] [--base
24) " --user-path | --user-pa
25) " --interpolated-path=pat
26) " --reuseaddr] [--detach]
27) " [--enable|disable|allow
28) " --inetd | [--listen=hos
29) " --user=user
30) " [directory...]";
31)
32) /* List of acceptable pathname prefi
33) static char **ok_paths;
```

git blame daemon.c

```
979e32fa (Randal L. Schwartz 2005-10-25 16:29:09 -0700
85023577 (Junio C Hamano 2006-12-19 14:34:12 -0800
77cb17e9 (Michal Ostrowski 2006-01-10 21:12:17 -0500
49ba83fb (Jon Loeliger 2006-09-19 20:31:51 -0500
f8ff0c06 (Petr Baudis 2005-09-22 11:25:28 +0200
85023577 (Junio C Hamano 2006-12-19 14:34:12 -0800
85023577 (Junio C Hamano 2006-12-19 14:34:12 -0800
695dffe2 (Johannes Schindelin 2006-09-28 12:00:35 +0200
415e7b87 (Patrick Welche 2007-10-18 18:17:39 +0100
9048fe1c (Petr Baudis 2005-09-24 16:13:01 +0200
f8ff0c06 (Petr Baudis 2005-09-22 11:25:28 +0200
1955fabf (Mark Wooding 2006-02-03 20:27:04 +0000
f8ff0c06 (Petr Baudis 2005-09-22 11:25:28 +0200
960deccb (H. Peter Anvin 2005-10-19 14:27:01 -0700
1b1dd23f (Stephan Beyer 2008-07-13 15:36:15 +0200
3bd62c21 (Stephen R. van den Berg 2008-08-14 20:02:20 +0200
3bd62c21 (Stephen R. van den Berg 2008-08-14 20:02:20 +0200
73a7a656 (Jens Axboe 2007-07-27 14:00:29 -0700
49ba83fb (Jon Loeliger 2006-09-19 20:31:51 -0500
678dac6b (Tilman Sauerbeck 2006-08-22 19:37:41 +0200
d9edcbd6 (Junio C Hamano 2006-09-07 01:40:04 -0700
dd467629 (Jon Loeliger 2006-09-26 09:47:43 -0500
dd467629 (Jon Loeliger 2006-09-26 09:47:43 -0500
dd467629 (Jon Loeliger 2006-09-26 09:47:43 -0500
4ae95682 (H. Peter Anvin 2005-09-26 19:10:55 -0700
4ae95682 (H. Peter Anvin 2005-09-26 19:10:55 -0700
96f1e58f (David Rientjes
```

```
2006-08-15 10:23:48 -0700
1) #include "cache.h"
2) #include "pkt-line.h"
3) #include "exec_cmd.h"
4) #include " interpolate.h"
5)
6) #include <syslog.h>
7)
8) #ifndef HOST_NAME_MAX
9) #define HOST_NAME_MAX 256
10) #endif
11)
12) #ifndef NI_MAXSERV
13) #define NI_MAXSERV 32
14) #endif
15)
16) static int log_syslog;
17) static int verbose;
18) static int reuseaddr;
19)
20) static const char daemon_usage[] =
21) "git daemon [--verbose] [--syslog] [
22) " --timeout=n ] [--init-ti
23) " --strict-paths] [--base
24) " --user-path | --user-pa
25) " --interpolated-path=pat
26) " --reuseaddr] [--detach]
27) " [--enable|disable|allow
28) " --inetd | [--listen=hos
29) " --user=user
30) " [directory...]";
31)
32) /* List of acceptable pathname prefi
33) static char **ok_paths;
```

git blame daemon.c

979e32fa (Randal L. Schwartz	2005-10-25 16:29:09 -0700	1) #include "cache.h"
85023577 (Junio C Hamano	2006-12-19 14:34:12 -0800	2) #include "pkt-line.h"
77cb17e9 (Michal Ostrowski	2006-01-10 21:12:17 -0500	3) #include "exec_cmd.h"
49ba83fb (Jon Loeliger	2006-09-19 20:31:51 -0500	4) #include " interpolate.h"
f8ff0c06 (Petr Baudis	2005-09-22 11:25:28 +0200	5)
85023577 (Junio C Hamano	2006-12-19 14:34:12 -0800	6) #include <syslog.h>
85023577 (Junio C Hamano	2006-12-19 14:34:12 -0800	7)
695dffe2 (Johannes Schindelin	2006-09-28 12:00:35 +0200	8) #ifndef HOST_NAME_MAX
695dffe2 (Johannes Schindelin	2006-09-28 12:00:35 +0200	9) #define HOST_NAME_MAX 256
695dffe2 (Johannes Schindelin	2006-09-28 12:00:35 +0200	10) #endif
695dffe2 (Johannes Schindelin	2006-09-28 12:00:35 +0200	11)
415e7b87 (Patrick Welche	2007-10-18 18:17:39 +0100	12) #ifndef NI_MAXSERV
415e7b87 (Patrick Welche	2007-10-18 18:17:39 +0100	13) #define NI_MAXSERV 32
415e7b87 (Patrick Welche	2007-10-18 18:17:39 +0100	14) #endif
415e7b87 (Patrick Welche	2007-10-18 18:17:39 +0100	15)
9048fe1c (Petr Baudis	2005-09-24 16:13:01 +0200	16) static int log_syslog;
f8ff0c06 (Petr Baudis	2005-09-22 11:25:28 +0200	17) static int verbose;
1955fabf (Mark Wooding	2006-02-03 20:27:04 +0000	18) static int reuseaddr;
f8ff0c06 (Petr Baudis	2005-09-22 11:25:28 +0200	19)
960deccb (H. Peter Anvin	2005-10-19 14:27:01 -0700	20) static const char daemon_usage[] =
1b1dd23f (Stephan Beyer	2008-07-13 15:36:15 +0200	21) "git daemon [--verbose] [--syslog] [
3bd62c21 (Stephen R. van den Berg	2008-08-14 20:02:20 +0200	22) " --timeout=n] [--init-ti
3bd62c21 (Stephen R. van den Berg	2008-08-14 20:02:20 +0200	23) " --strict-paths] [--base
73a7a656 (Jens Axboe	2007-07-27 14:00:29 -0700	24) " --user-path --user-pa
49ba83fb (Jon Loeliger	2006-09-19 20:31:51 -0500	25) " --interpolated-path=pat
678dac6b (Tilman Sauerbeck	2006-08-22 19:37:41 +0200	26) " --reuseaddr] [--detach]
d9edcbd6 (Junio C Hamano	2006-09-07 01:40:04 -0700	27) " --[enable disable allow
dd467629 (Jon Loeliger	2006-09-26 09:47:43 -0500	28) " --inetd [--listen=hos
dd467629 (Jon Loeliger	2006-09-26 09:47:43 -0500	29) " [--user=user
dd467629 (Jon Loeliger	2006-09-26 09:47:43 -0500	30) " [directory...]";
4ae95682 (H. Peter Anvin	2005-09-26 19:10:55 -0700	31)
4ae95682 (H. Peter Anvin	2005-09-26 19:10:55 -0700	32) /* List of acceptable pathname prefi
96f1e58f (David Rienties	2006-08-15 10:23:48 -0700	33) static char **ok_paths:

```
git blame -c GITPackUpload.m
```

git blame -C GITPackUpload.m

```
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 12)
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 13) #define PACK_SIGN
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 14) #define PACK_VERS
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 15)
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 16) @implementation G
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 17)
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 18) @synthesize gitRe
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 19) @synthesize needR
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 20) @synthesize gitSo
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 21) @synthesize refDi
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 22)
a2cbabf5 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-25 22:29:39 +0100 23) - (id) initWithGi
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 24) {
a2cbabf5 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-25 22:29:39 +0100 25)     gitRepo =
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 26)     needRefs =
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 27)     gitSocket
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 28)     return sel
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 29) }
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 30)
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 31) - (bool) uploadPa
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 32) {
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 33)     NSLog(@"up
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 34)     NSString *
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 35)     NSArray *t
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 36)
56ef2caf Source/Network/GITServerHandler.m (Scott Chacon 2009-01-05 21:44:26 -0800 37)     refDict =
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 38)
```

git blame -C GITPackUpload.m

f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 12)
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 13) #define PACK_SIGN
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 14) #define PACK_VERS
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 15)
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 16) @implementation G
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 17)
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 18) @synthesize gitRe
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 19) @synthesize needR
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 20) @synthesize gitSo
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 21) @synthesize refDi
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 22)
a2cbabf5	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-25 22:29:39 +0100 23) - (id) initWithGi
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 24) {
a2cbabf5	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-25 22:29:39 +0100 25) gitRepo =
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 26) needRefs =
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 27) gitSocket
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 28) return sel
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 29) }
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 30)
ad11ac80	Source/Network/GITPackUpload.m	(Scott Chacon 2009-03-24 18:32:50 +0100 31) - (bool) uploadPa
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 32) {
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 33) NSLog(@"up
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 34) NSString *
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 35) NSArray *t
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 36)
56ef2caf	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-05 21:44:26 -0800 37) refDict =
f344f58d	Source/Network/GITServerHandler.m	(Scott Chacon 2009-01-04 18:59:04 -0800 38)

git blame -C GITPackUpload.m

```
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 12)
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 13) #define PACK_SIGN
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 14) #define PACK_VERS
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 15)
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 16) @implementation G
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 17)
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 18) @synthesize gitRe
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 19) @synthesize needR
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 20) @synthesize gitSo
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 21) @synthesize refDi
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 22)
a2cbabf5 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-25 22:29:39 +0100 23) - (id) initWithGi
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 24) {
a2cbabf5 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-25 22:29:39 +0100 25)     gitRepo =
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 26)     needRefs =
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 27)     gitSocket
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 28)     return sel
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 29) }
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 30)
ad11ac80 Source/Network/GITPackUpload.m (Scott Chacon 2009-03-24 18:32:50 +0100 31) - (bool) uploadPa
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 32) {
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 33)     NSLog(@"up
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 34)     NSString *
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 35)     NSArray *t
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 36)
56ef2caf Source/Network/GITServerHandler.m (Scott Chacon 2009-01-05 21:44:26 -0800 37)     refDict =
f344f58d Source/Network/GITServerHandler.m (Scott Chacon 2009-01-04 18:59:04 -0800 38)
```

Bisecting

**binary search for where
a bug was introduced**

```
$ git bisect start
```

```
$ git bisect start  
$ git bisect bad
```

```
$ git bisect start  
$ git bisect bad  
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
```

```
$ git bisect start
$ git bisect bad
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo
```

```
$ git bisect start
$ git bisect bad
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo
$ git bisect good
```

```
$ git bisect start
$ git bisect bad
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo
$ git bisect good
Bisecting: 3 revisions left to test after this
[b047b02ea83310a70fd603dc8cd7a6cd13d15c04] secure this thing
```

```
$ git bisect start
$ git bisect bad
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo
$ git bisect good
Bisecting: 3 revisions left to test after this
[b047b02ea83310a70fd603dc8cd7a6cd13d15c04] secure this thing
$ git bisect bad
```

```
$ git bisect start
$ git bisect bad
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo
$ git bisect good
Bisecting: 3 revisions left to test after this
[b047b02ea83310a70fd603dc8cd7a6cd13d15c04] secure this thing
$ git bisect bad
Bisecting: 1 revisions left to test after this
[f71ce38690acf49c1f3c9bea38e09d82a5ce6014] drop exceptions table
```

```
$ git bisect start
$ git bisect bad
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo
$ git bisect good
Bisecting: 3 revisions left to test after this
[b047b02ea83310a70fd603dc8cd7a6cd13d15c04] secure this thing
$ git bisect bad
Bisecting: 1 revisions left to test after this
[f71ce38690acf49c1f3c9bea38e09d82a5ce6014] drop exceptions table
$ git bisect good
```

```
$ git bisect start
$ git bisect bad
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo
$ git bisect good
Bisecting: 3 revisions left to test after this
[b047b02ea83310a70fd603dc8cd7a6cd13d15c04] secure this thing
$ git bisect bad
Bisecting: 1 revisions left to test after this
[f71ce38690acf49c1f3c9bea38e09d82a5ce6014] drop exceptions table
$ git bisect good
b047b02ea83310a70fd603dc8cd7a6cd13d15c04 is first bad commit
commit b047b02ea83310a70fd603dc8cd7a6cd13d15c04
Author: PJ Hyett <pjhyett@gmail.com>
Date: Tue Jan 27 14:48:32 2009 -0800
```

secure this thing

```
:040000 040000 40ee3e7821b895e52c1695092db9bdc4c61d1730
f24d3c6ebcf639b1a3814550e62d60b8e68a8e4 M config
```

```
$ git bisect start
$ git bisect bad
$ git bisect good 3acb4c2c6666ed6cb91cb0b983efe9d50cf8cfbe
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo
$ git bisect good
Bisecting: 3 revisions left to test after this
[b047b02ea83310a70fd603dc8cd7a6cd13d15c04] secure this thing
$ git bisect bad
Bisecting: 1 revisions left to test after this
[f71ce38690acf49c1f3c9bea38e09d82a5ce6014] drop exceptions table
$ git bisect good
```

b047b02ea83310a70fd603dc8cd7a6cd13d15c04 is first bad commit

commit b047b02ea83310a70fd603dc8cd7a6cd13d15c04

Author: PJ Hyett <pjhyett@gmail.com>

Date: Tue Jan 27 14:48:32 2009 -0800

secure this thing

```
:040000 040000 40ee3e7821b895e52c1695092db9bdc4c61d1730
f24d3c6ebcf639b1a3814550e62d60b8e68a8e4 M config
```

```
$ git bisect reset
```

Resources

Resources

git-scm.com

Resources

git-scm.com

gitcasts.com

Resources

git-scm.com

gitcasts.com

learn.github.com

Resources

git-scm.com

gitcasts.com

learn.github.com

#git / #github on IRC

Resources

git-scm.com

gitcasts.com

learn.github.com

#git / #github on IRC

peepcode - git book and screencast

Resources

git-scm.com

gitcasts.com

learn.github.com

#git / #github on IRC

peepcode - git book and screencast

schacon@gmail.com