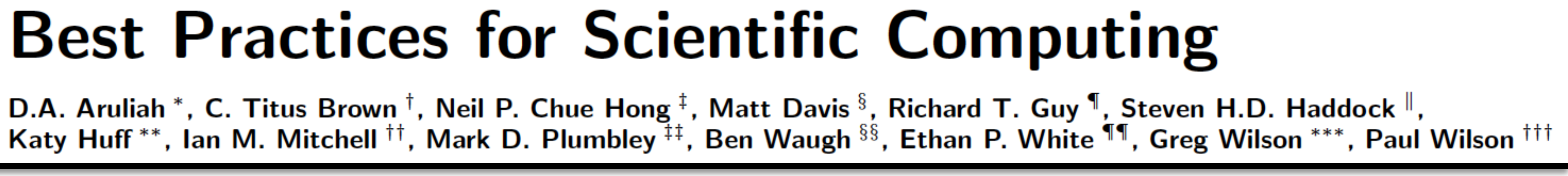
Version Control and Git

A brief and entirely crushing overview of version control systems, Git, and source code management for scientific development…

Joe Hamman

Hydro-group Computing Seminar

October 21, 2013



Best Practices for Scientific Computing

D.A. Aruliah [*], C. Titus Brown [†], Neil P. Chue Hong [‡], Matt Davis [§], Richard T. Guy [¶], Steven H.D. Haddock [‖], Katy Huff [**], Ian M. Mitchell [††], Mark D. Plumbley [‡‡], Ben Waugh [§§], Ethan P. White [¶¶], Greg Wilson [***], Paul Wilson [†††]

1. Write programs for people, not computers.
2. Automate repetitive tasks.
3. Use the computer to record history.
4. Make incremental changes.
5. Use version control.
6. Don’t repeat yourself (or others).
7. Plan for mistakes.
8. Optimize software only after it works correctly.
9. Document design and purpose, not mechanics.
10. Conduct code reviews.

Version Control Systems

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Version Control Systems

Local data model	**Client-server model**	**Distributed model**
All developers must use the same computer system.	Each developers use a shared single repository.	Each developer works directly with his or her own local repository
Examples		
Revision Control System (RCS)	Concurrent Versions System (CVS)	Git
Source Code Control System (SCCS)	Subversion (svn)	Mercurial

This presentation is based on:

Intro to Git

Scott Chacon

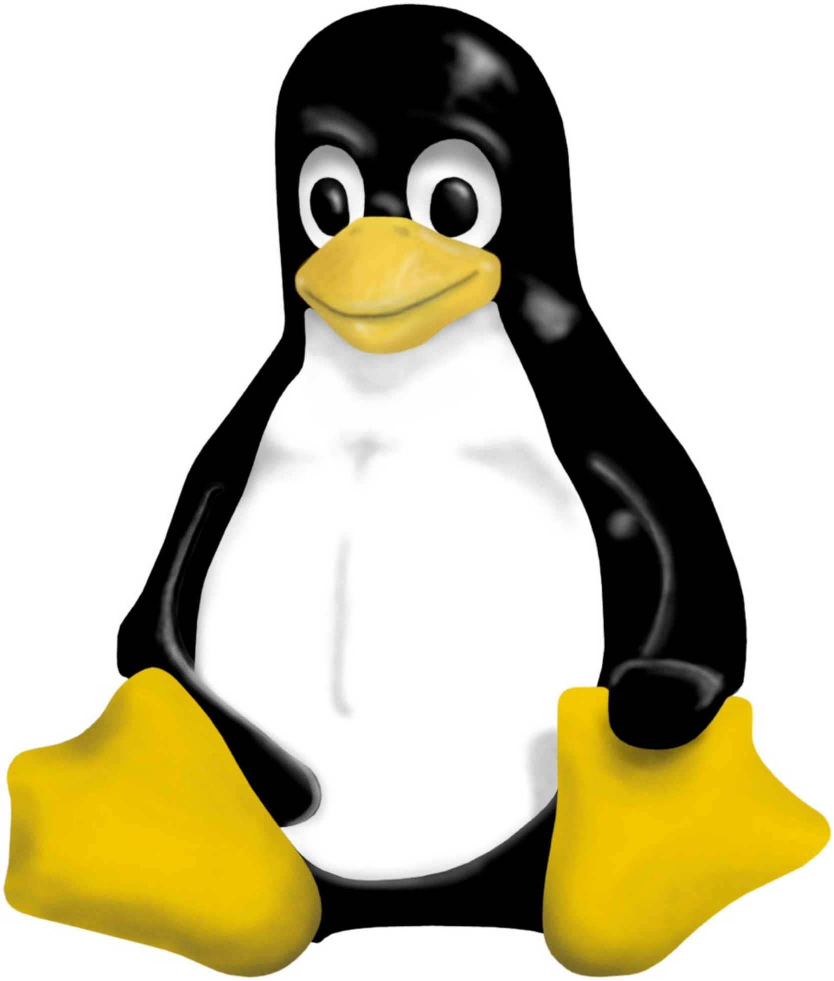
adapted by Bart Nijssen…and Joe Hamman

A PDF of the original presentation can be accessed at: https://github.com/schacon/git-presentations/blob/master/basic_git_talk/BasicGitTalk.pdf?raw=true

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



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

Fully Distributed

(almost) everything is local

No Network Needed

Performing a diff

Viewing file history

Committing changes

Merging branches

Obtaining any other revision of a file

Switching branches

which means

everything is fast

every clone is a backup

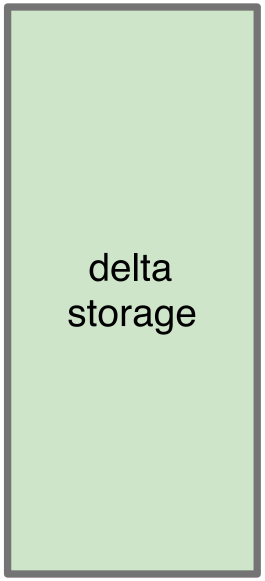
work offline

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

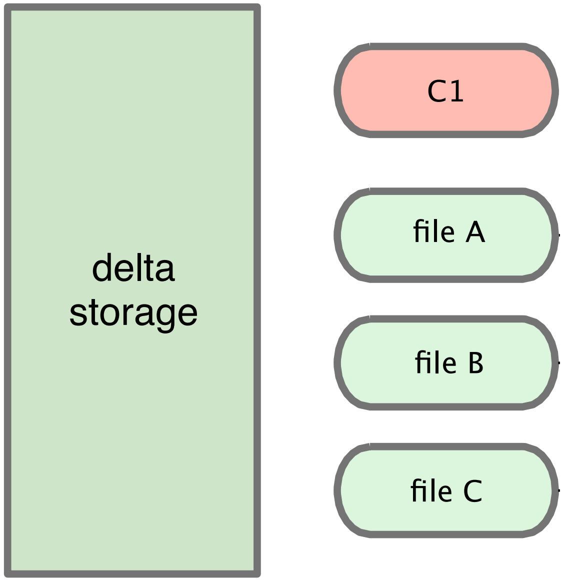
Immutable

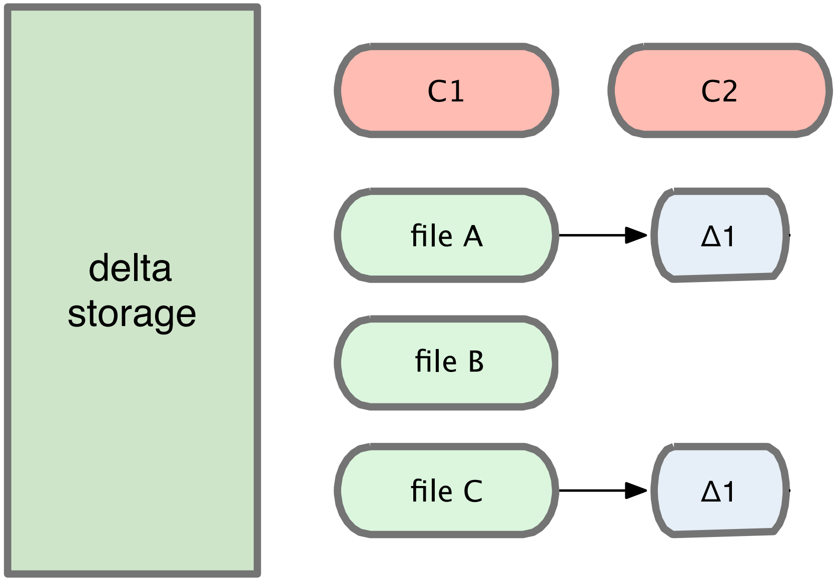
(almost) never removes data

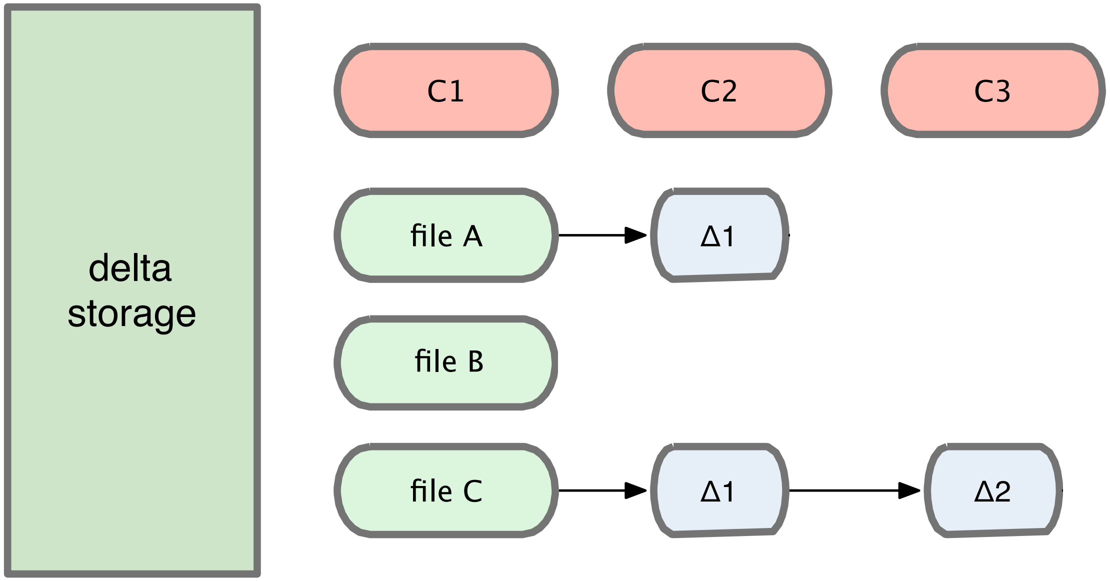
Snapshots, not Patches

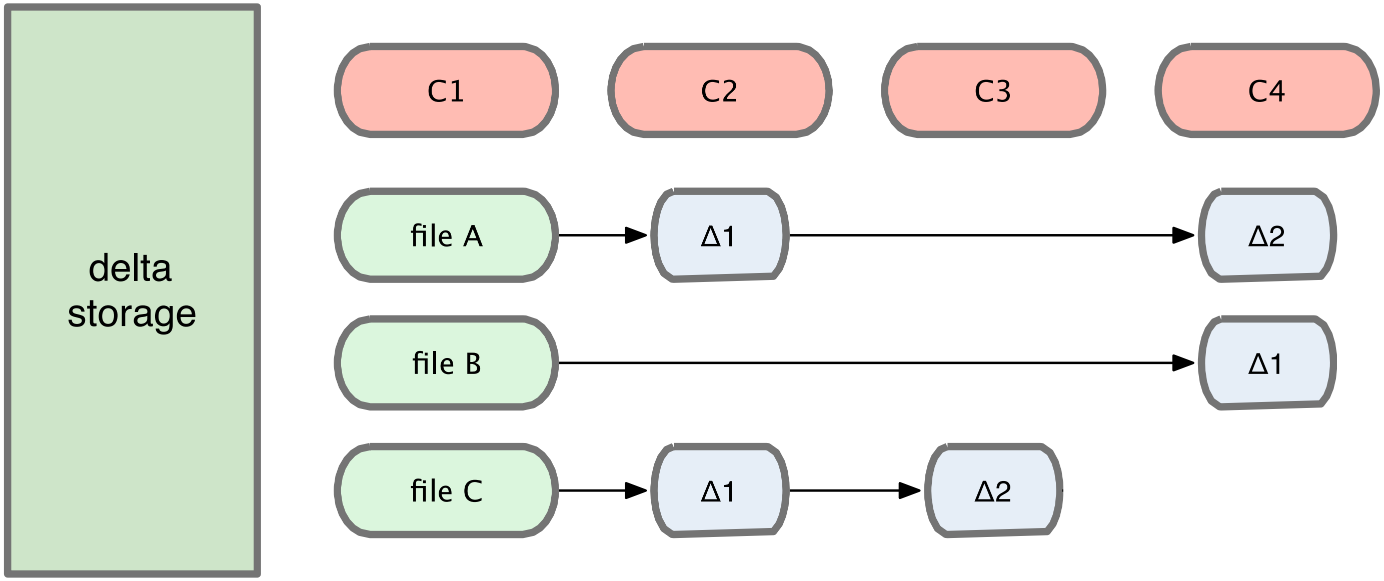


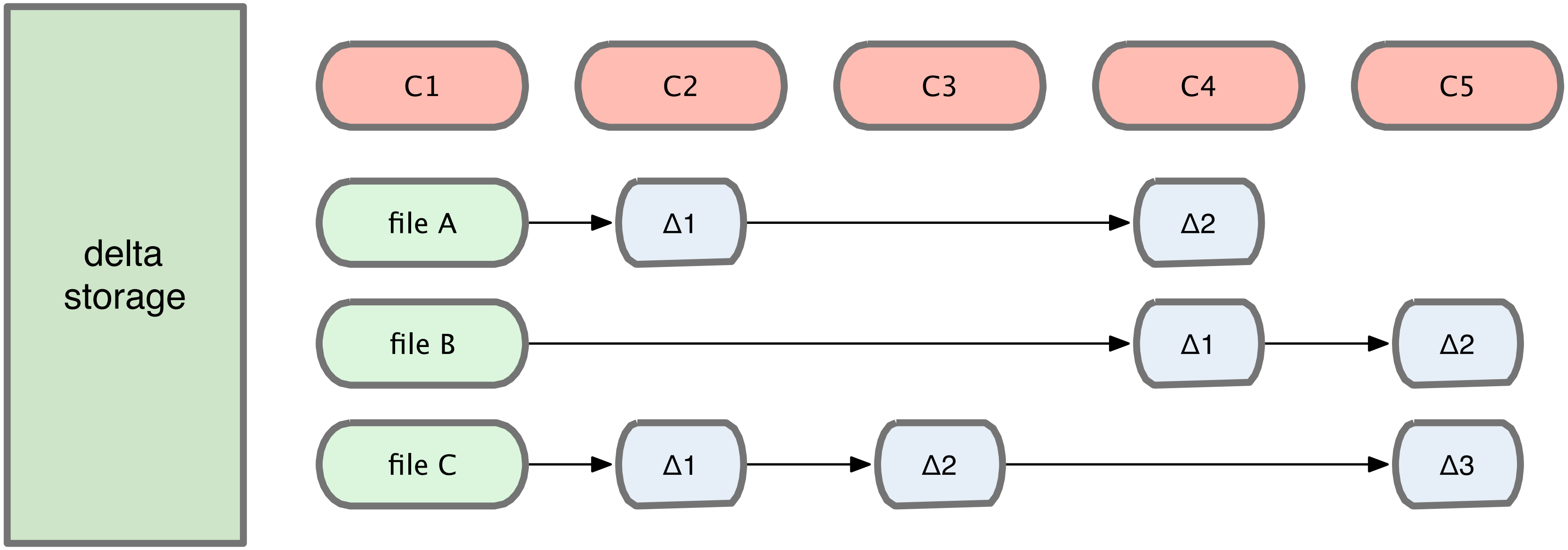
delta
storage

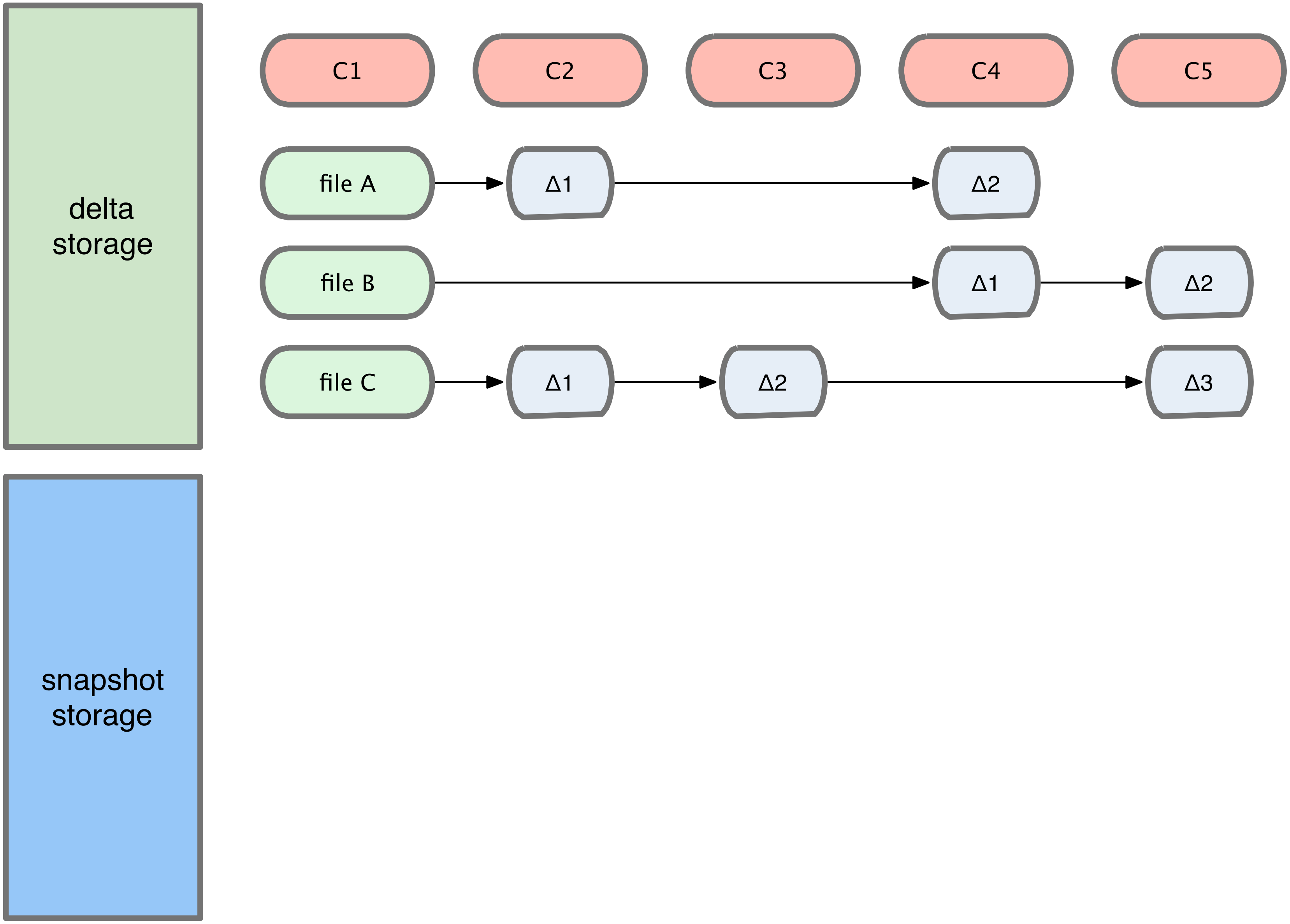

delta storage
C1
file A
file B
file C

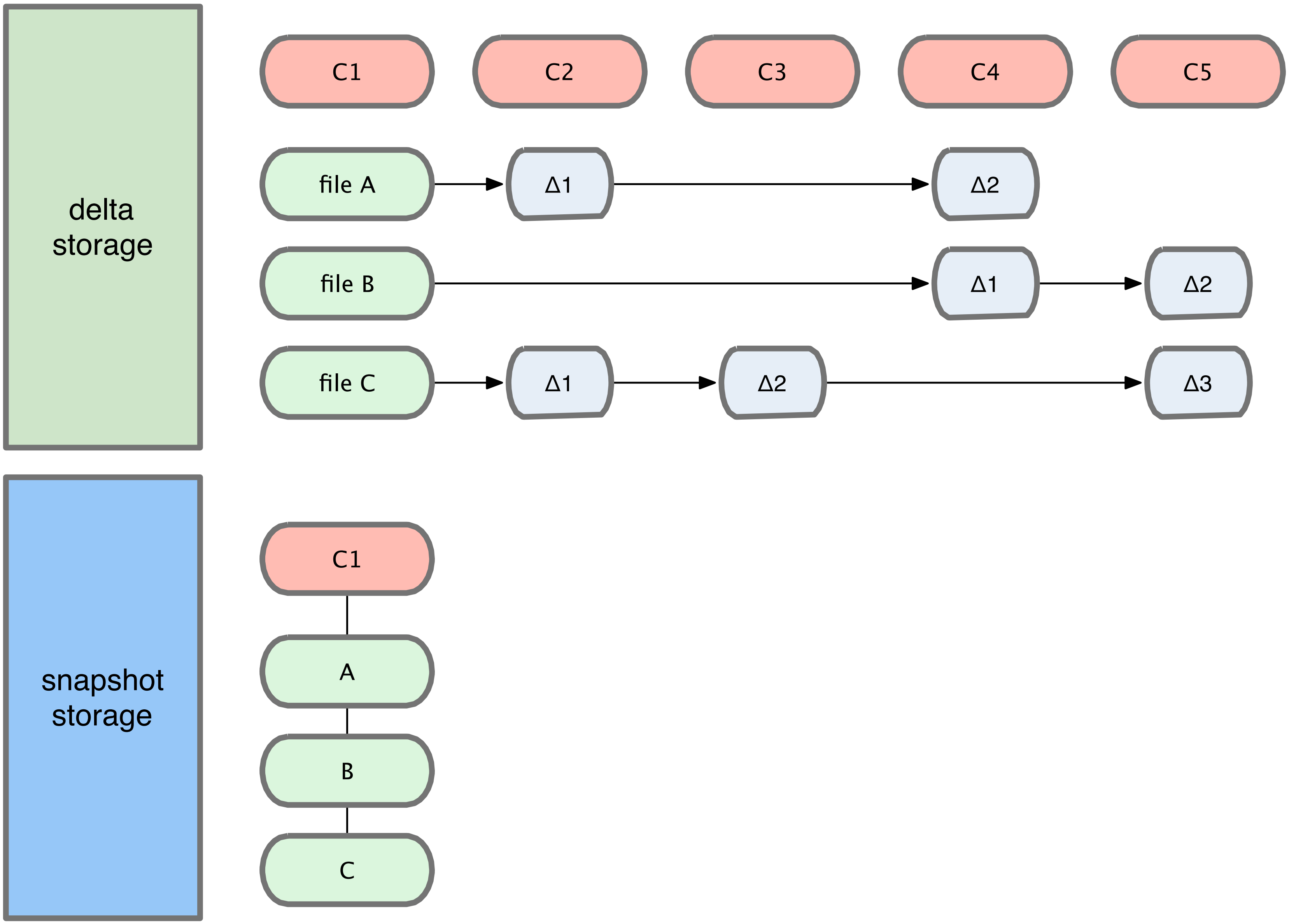

delta storage
C1
C2
file A
Δ1
file B
file C
Δ1

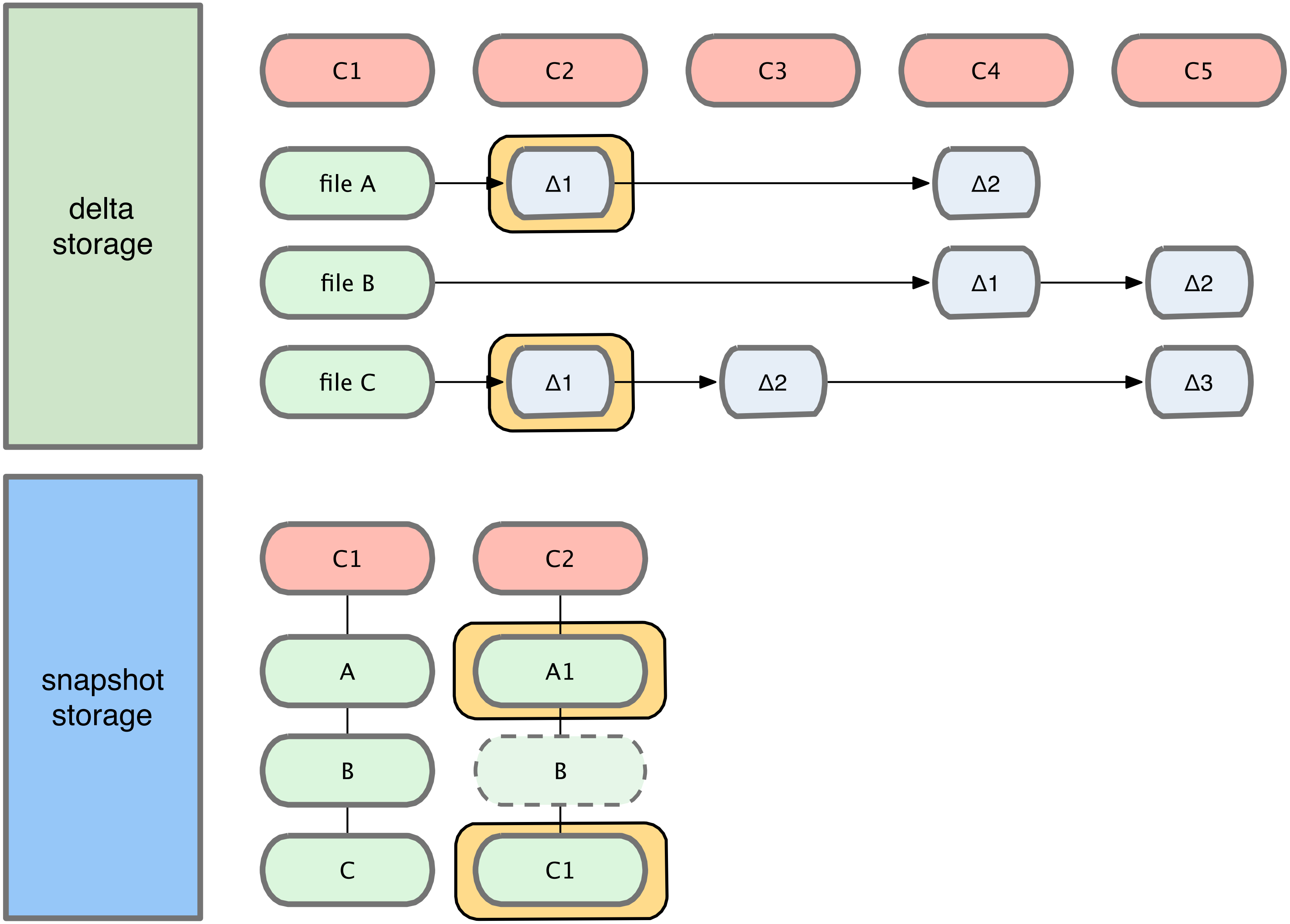

delta storage
C1
C2
C3
file A
Δ1
file B
file C
Δ1
Δ2

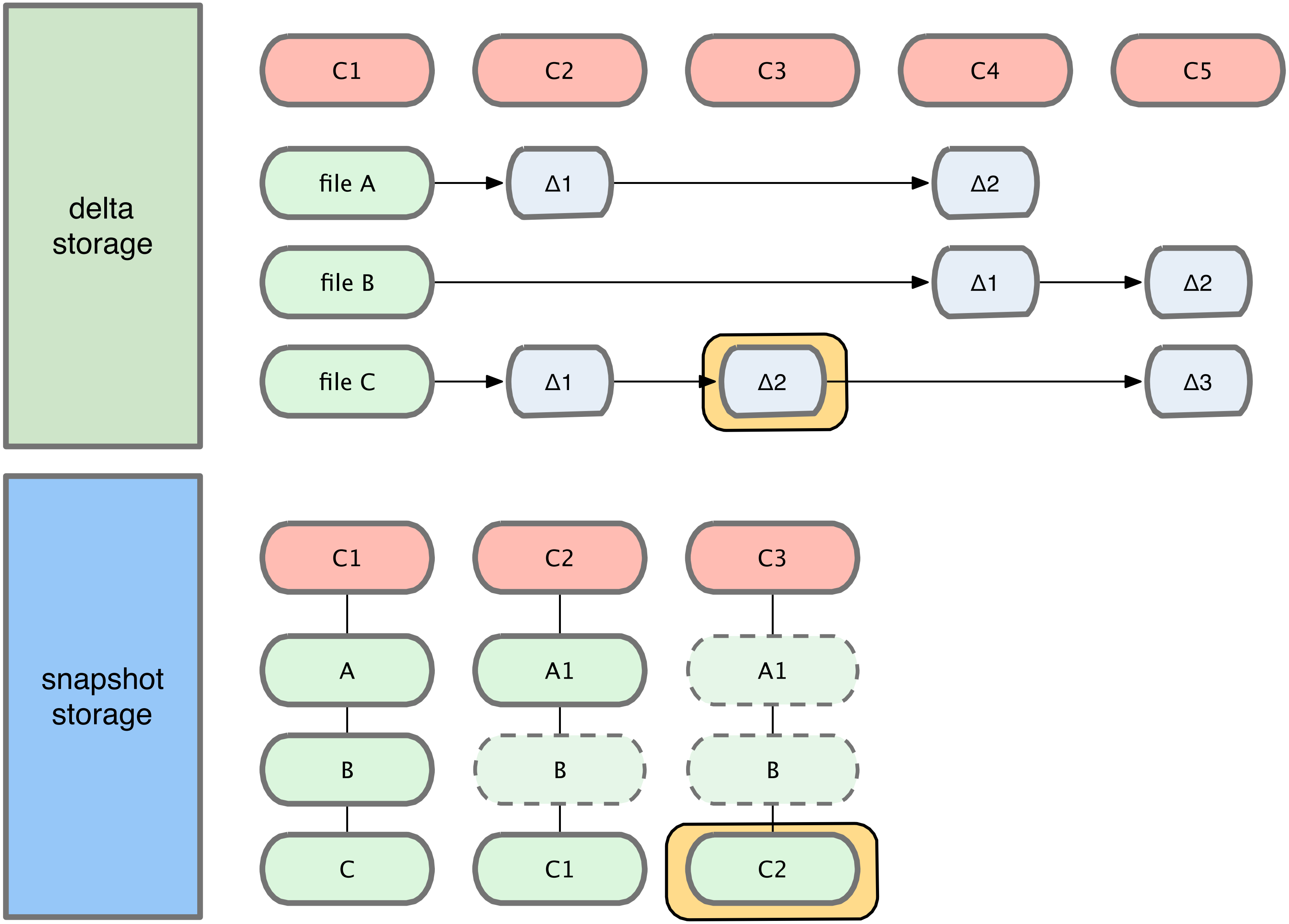

delta storage
C1
C2
C3
C4
file A
Δ1
Δ2
file B
Δ1
file C
Δ1
Δ2

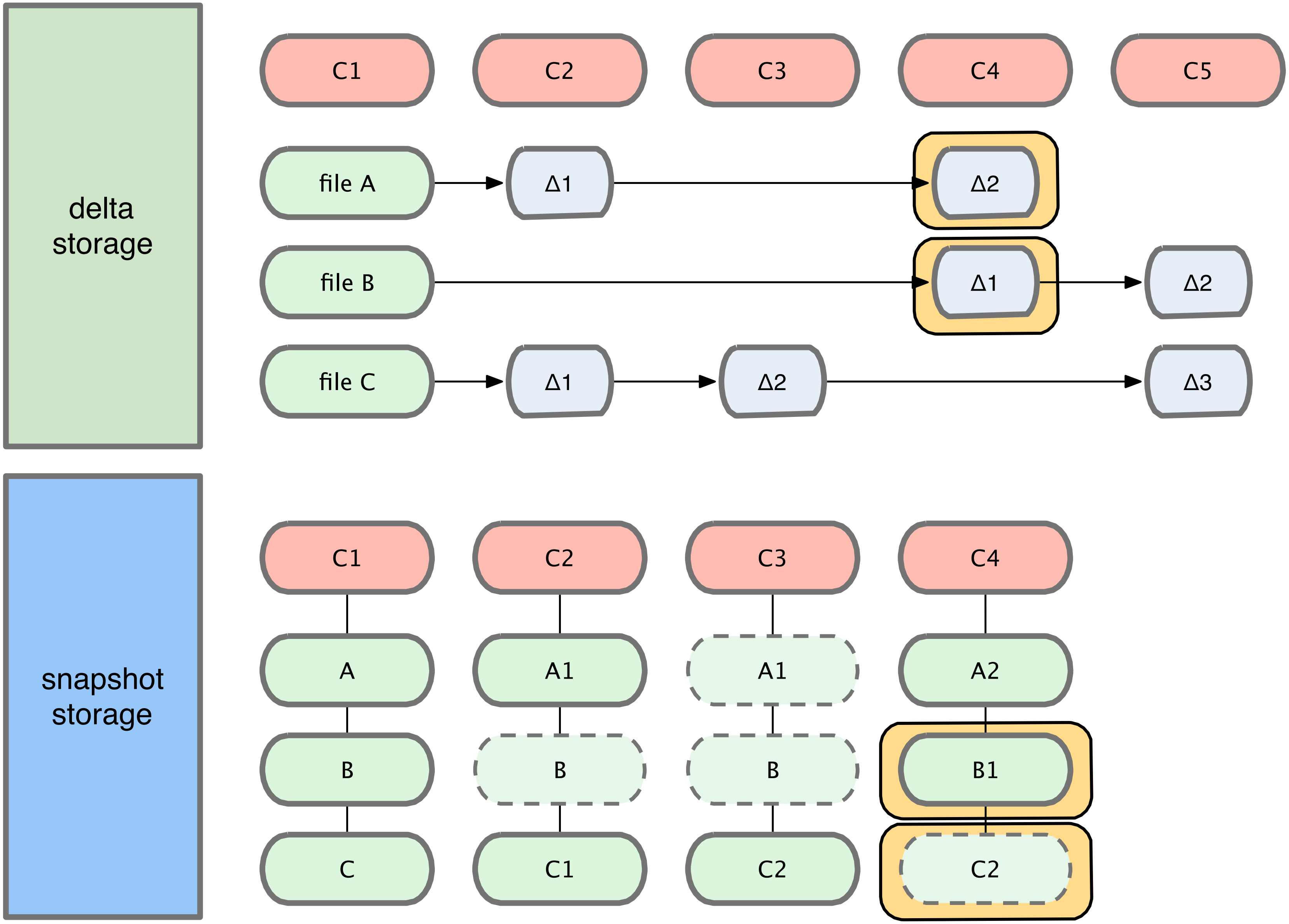

delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3

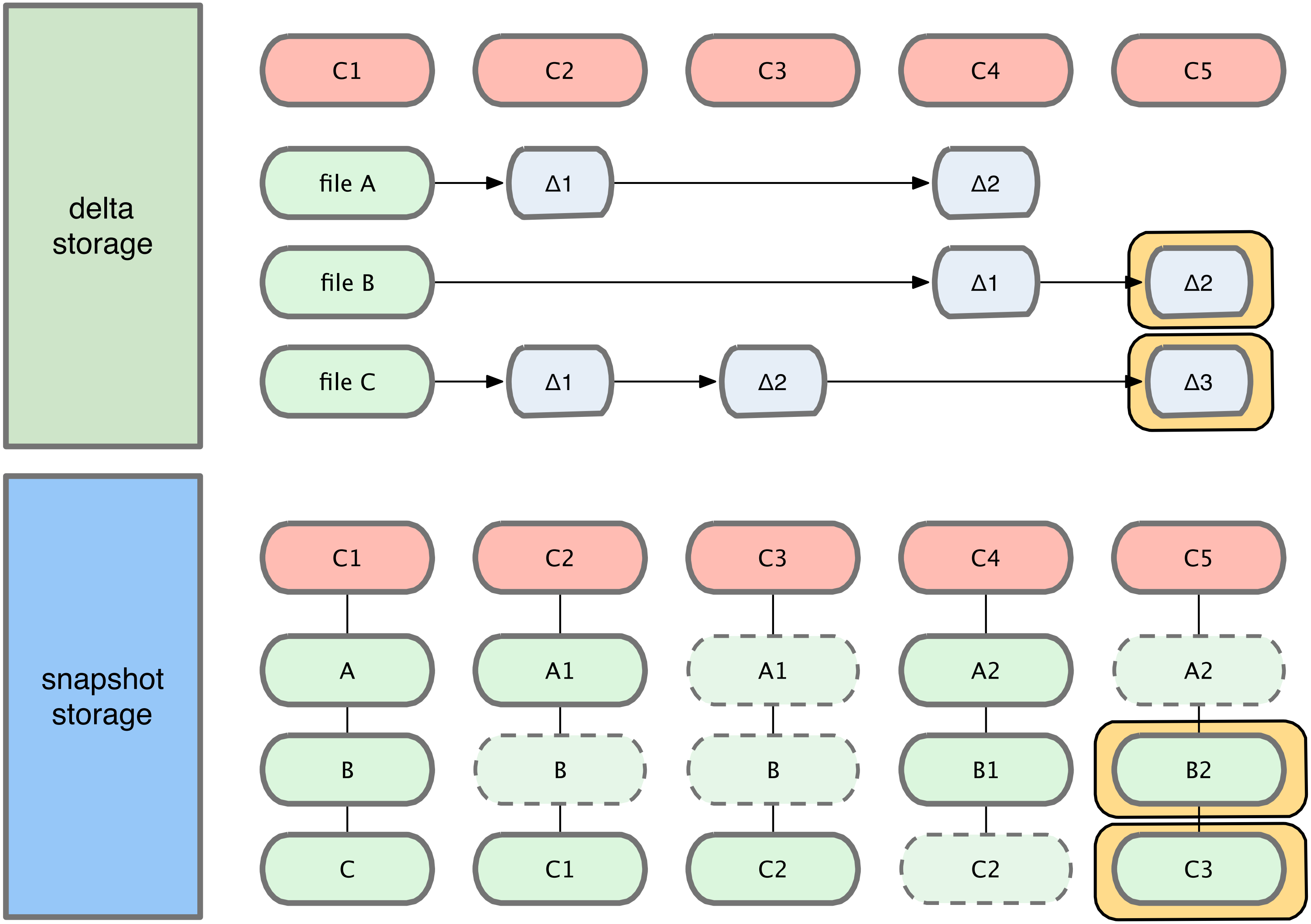

C1
C2
C3
C4
C5
delta storage
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage

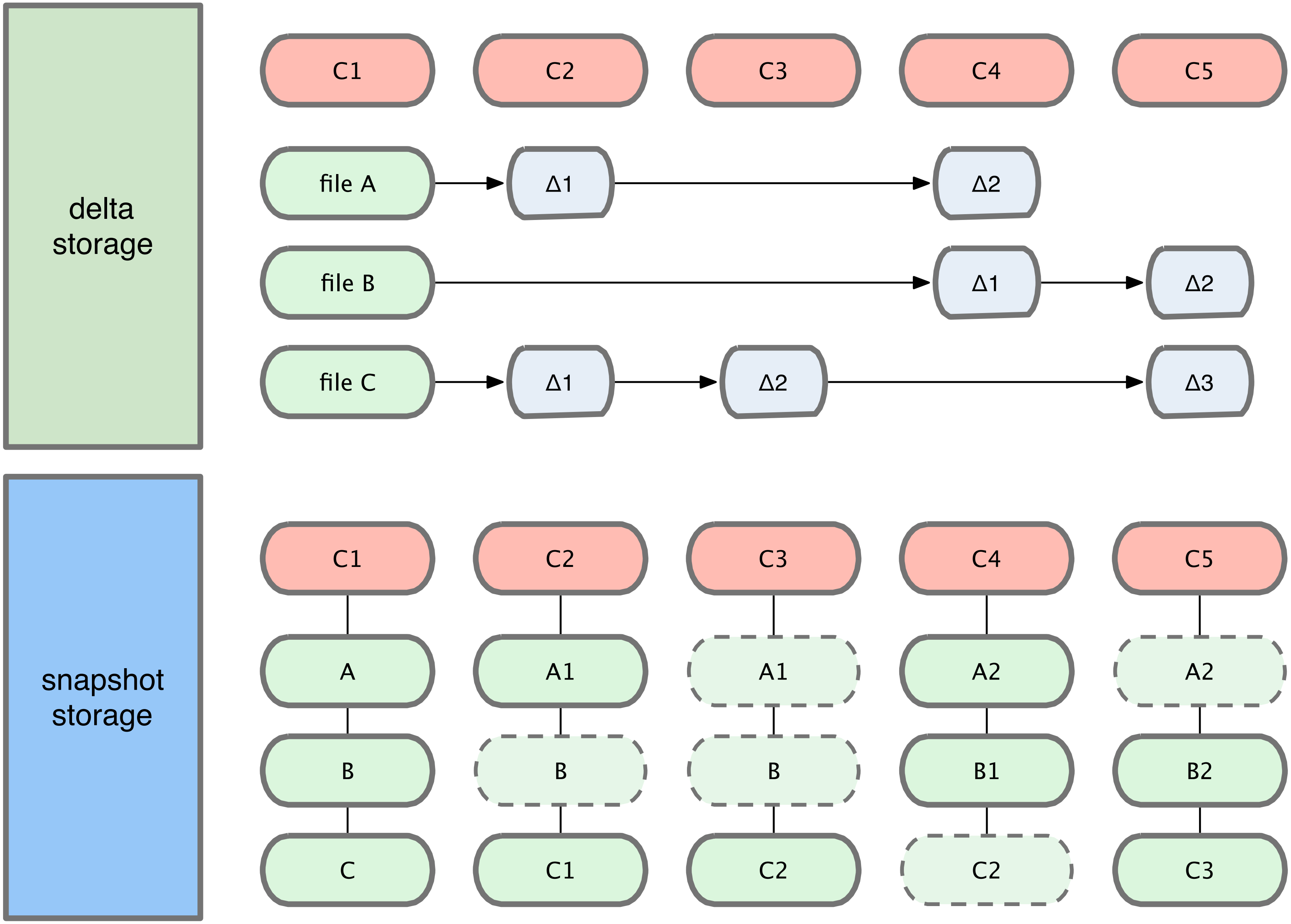

delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
A
B
C

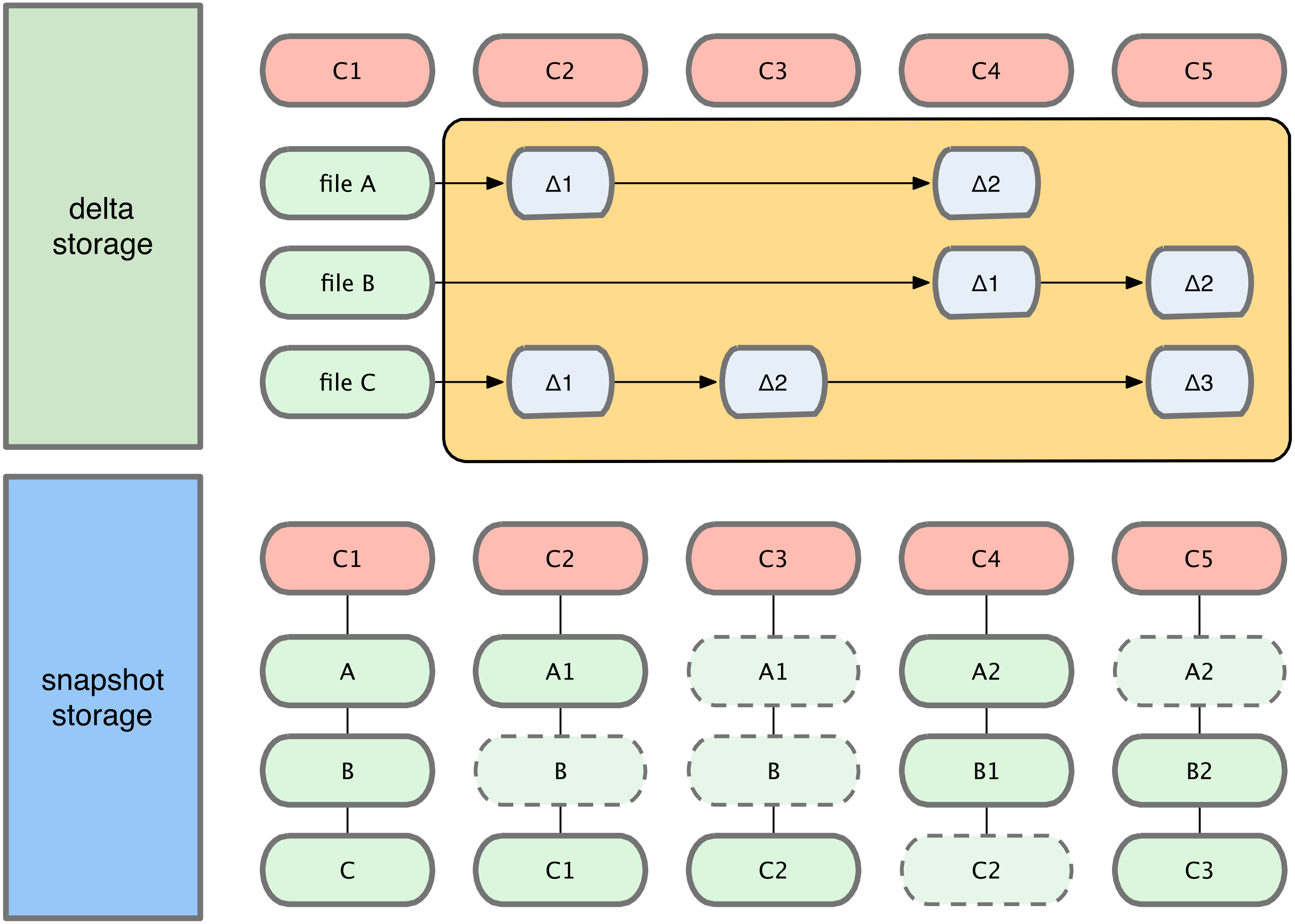

delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
C2
A
A1
B
B
C
C1

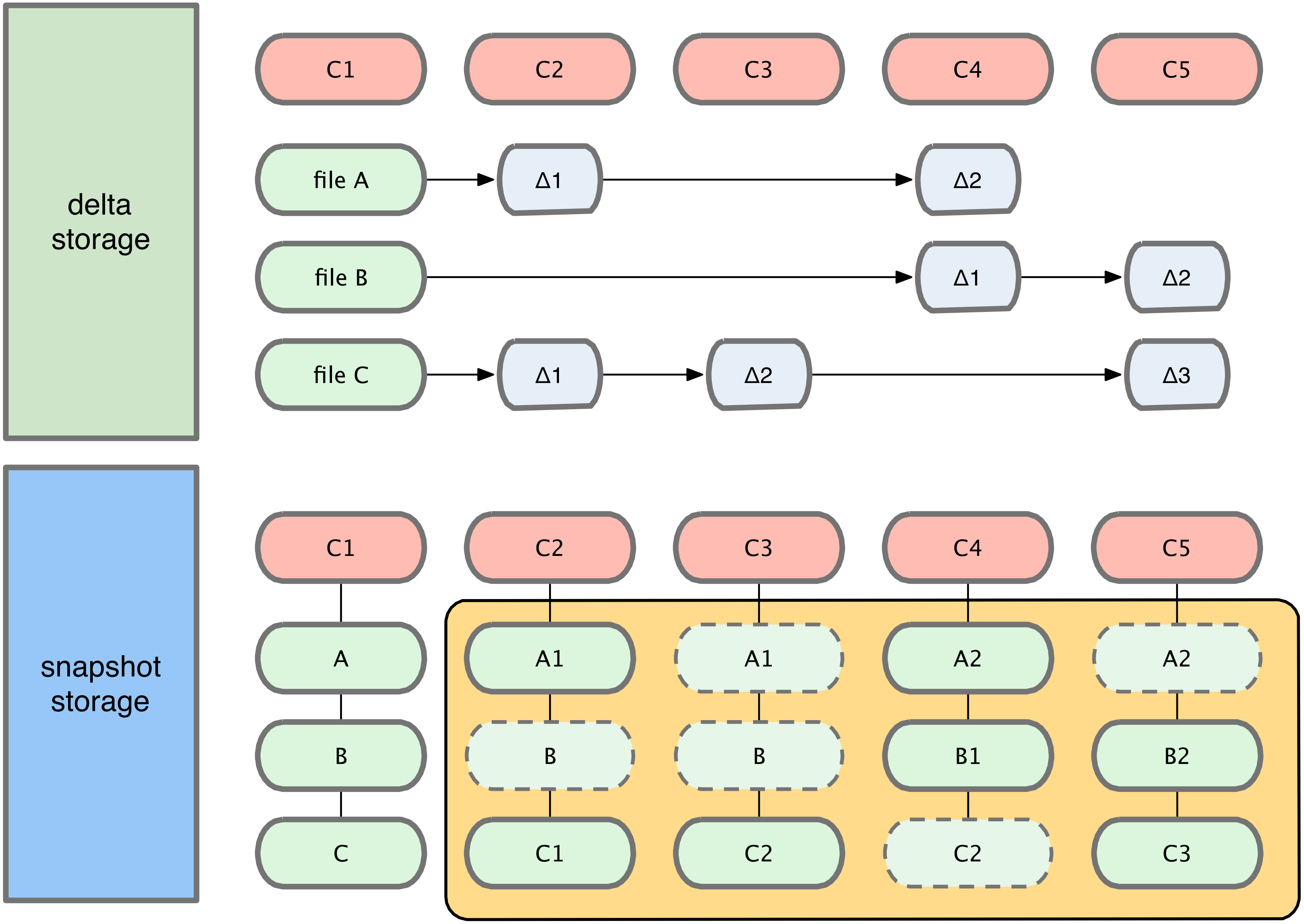

delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
A
B
C
C2
A1
B
C1
C3
A1
B
C2


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
C2
C3
C4
A
A1
A1
A2
B
B
B
B1
C
C1
C2
C2


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
C2
C3
C4
C5
A
A1
A1
A2
A2
B
B
B
B1
B2
C
C1
C2
C2
C3


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
A
B
C
C2
A1
B
C1
C3
A1
B
C2
C4
A2
B1
C2
C5
A2
B2
C3


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
C2
C3
C4
C5
A
A1
A1
A2
A2
B
B
B
B1
B2
C
C1
C2
C2
C3


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
C2
C3
C4
C5
A
A1
A1
A2
A2
B
B
B
B1
B2
C
C1
C2
C2
C3

How Do I Use Git?

First Steps

GettingA Repo

git init -- Start from scratch

git clone -- Start with an existing repo

```
git init
```

In the directory where you want to start a repo:

```
$ git init
```

This creates a .git subdirectory that contains all the behind-the-scenes info that git interacts with (and that you never edit directly).

In the directory where you want to start a repo:

```
$ git init
```

This creates a .git subdirectory that contains all the behind-the-scenes info that git interacts with (and that you **never** edit directly).

```
$ tree -a
.
!"" .git
    #"" HEAD
    #"" config
    #"" description
    #"" hooks
    $   #"" applypatch-msg.sample
    $   #"" commit-msg.sample
    $   #"" post-update.sample
    $   #"" pre-applypatch.sample
    $   #"" pre-commit.sample
    $   #"" pre-rebase.sample
    $   #"" prepare-commit-msg.sample
    $   !"" update.sample
    #"" info
    $   !"" exclude
    #"" objects
    $   #"" info
    $   !"" pack
    !"" refs
        #"" heads
        !"" tags
```

Repository

```
$ tree -a
.
!"" .git
    #"" HEAD
    #"" config
    #"" description
    #"" hooks
    $   #"" applypatch-msg.sample
    $   #"" commit-msg.sample
    $   #"" post-update.sample
    $   #"" pre-applypatch.sample
    $   #"" pre-commit.sample
    $   #"" pre-rebase.sample
    $   #"" prepare-commit-msg.sample
    $   !"" update.sample
    #"" info
    $   !"" exclude
    #"" objects
    $   #"" info
    $   !"" pack
    !"" refs
        #"" heads
        !"" tags
```

Set default settings:

```
$ git config --global user.name “Joe Hamman”

$ git config --global user.email “jhamman1@uw.edu”
```

Note that you override your global settings for a specific repo:

```
$ git config --local <option> <value>
```

You can get a listing of you current settings:

```
$ git config -l
```

You can directly edit your current settings, e.g.:

```
$ git config --global -e
```

git settings

/etc/gitconfig

git settings

/etc/gitconfig

~/.gitconfig

git settings

/etc/gitconfig

~/.gitconfig

.git/config

```
$ git init
$ touch hello_world.py
$ git add .
$ git commit -m ‘first commit’
$ tree -a
.
#"" .git
$   #"" COMMIT_EDITMSG
$   #"" HEAD
$   #"" config
$   #"" description
$   #"" hooks
$   $   #"" applypatch-msg.sample
$   $   #"" commit-msg.sample
$   $   #"" post-update.sample
$   $   #"" pre-applypatch.sample
$   $   #"" pre-commit.sample
$   $   #"" pre-rebase.sample
$   $   #"" prepare-commit-msg.sample
$   $   !"" update.sample
$   #"" index
$   #"" info
$   $   !"" exclude
$   #"" logs
$   $   #"" HEAD
$   $   !"" refs
$   $       !"" heads
$   $           !"" master
$   #"" objects
$   $   #"" 21
$   $   $   !"" 85689457a69bc7fabdc8245a7856bc733d44e4
$   $   #"" e1
$   $   $   !"" 08bfe338fecfbad4b9025092a8c8bbacac977e
$   $   #"" e6
$   $   $   !"" 9de29bb2d1d6434b8b29ae775ad8c2e48c5391
$   $   #"" info
$   $   !"" pack
$   !"" refs
$       #"" heads
$       $   !"" master
$       !"" tags
!"" hello_world.py
```

```
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

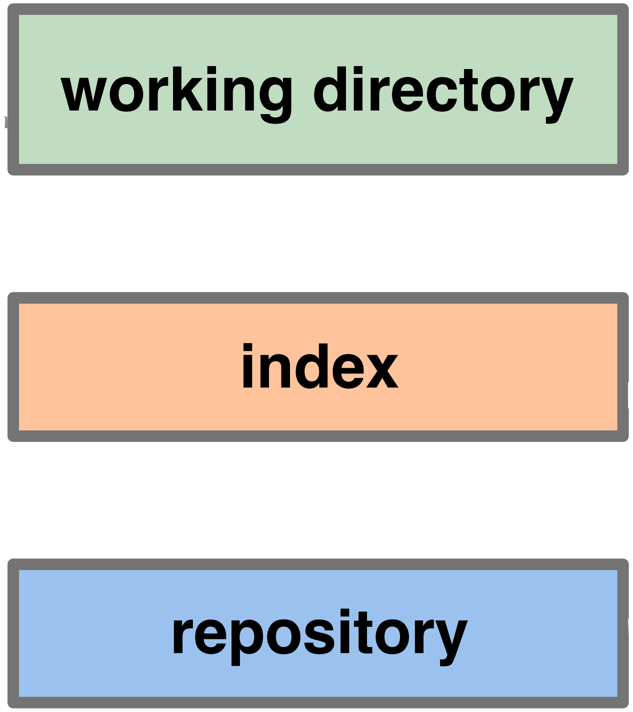
A Basic Workflow

Edit files

Stage the changes

Review your changes

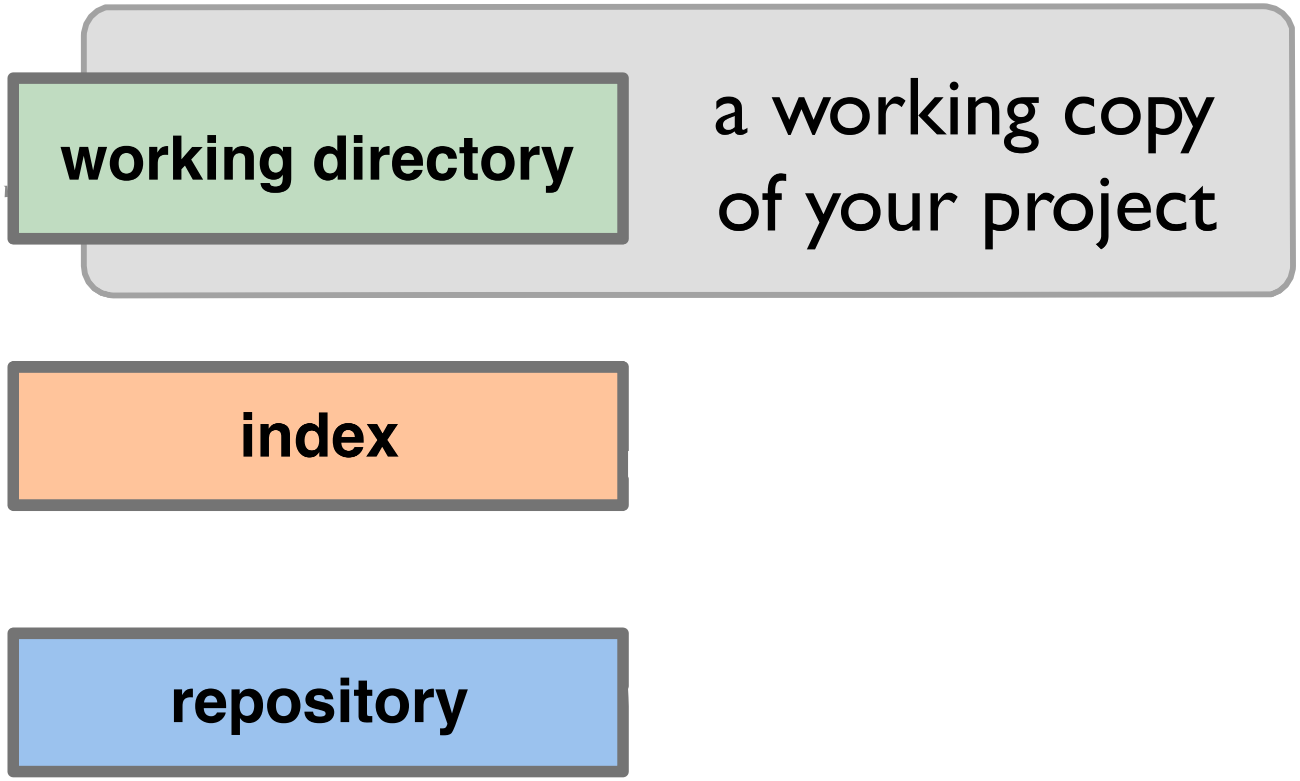
Commit the changes



working directory

index

repository

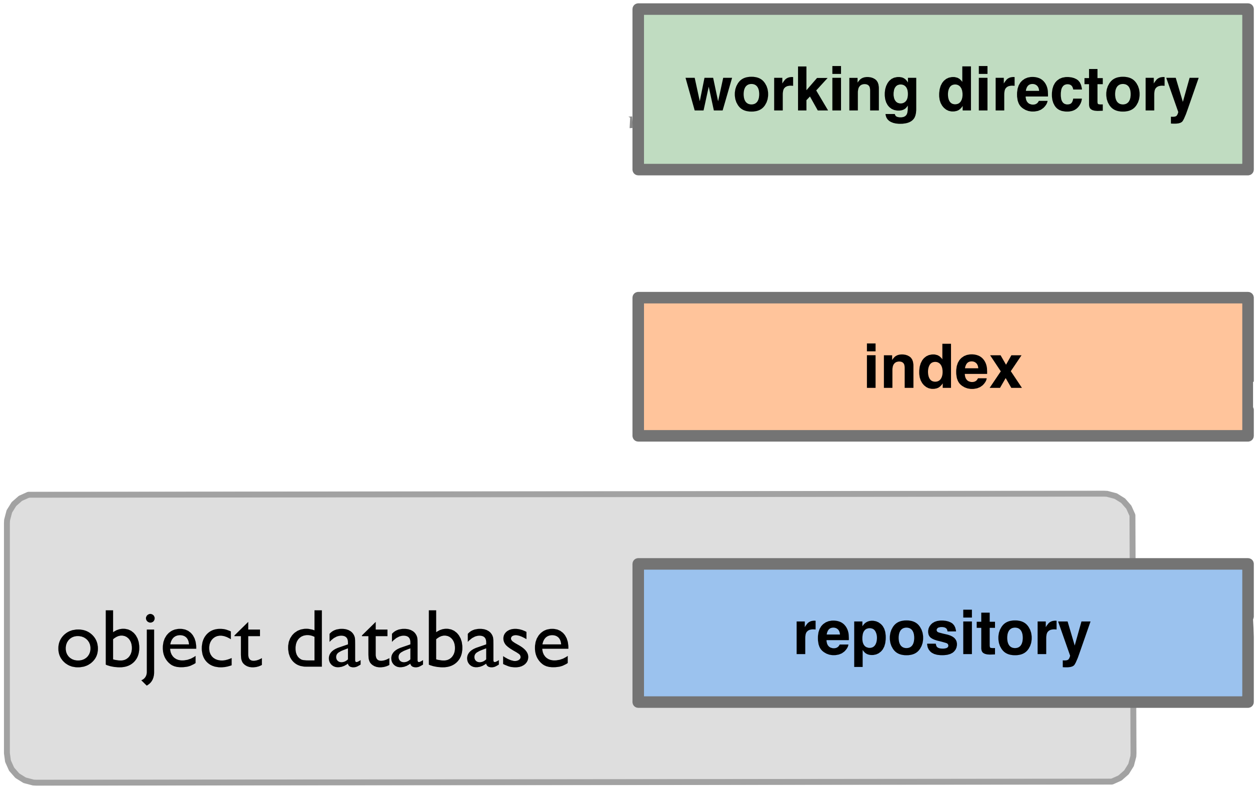


working directory

a working copy of your project

index

repository

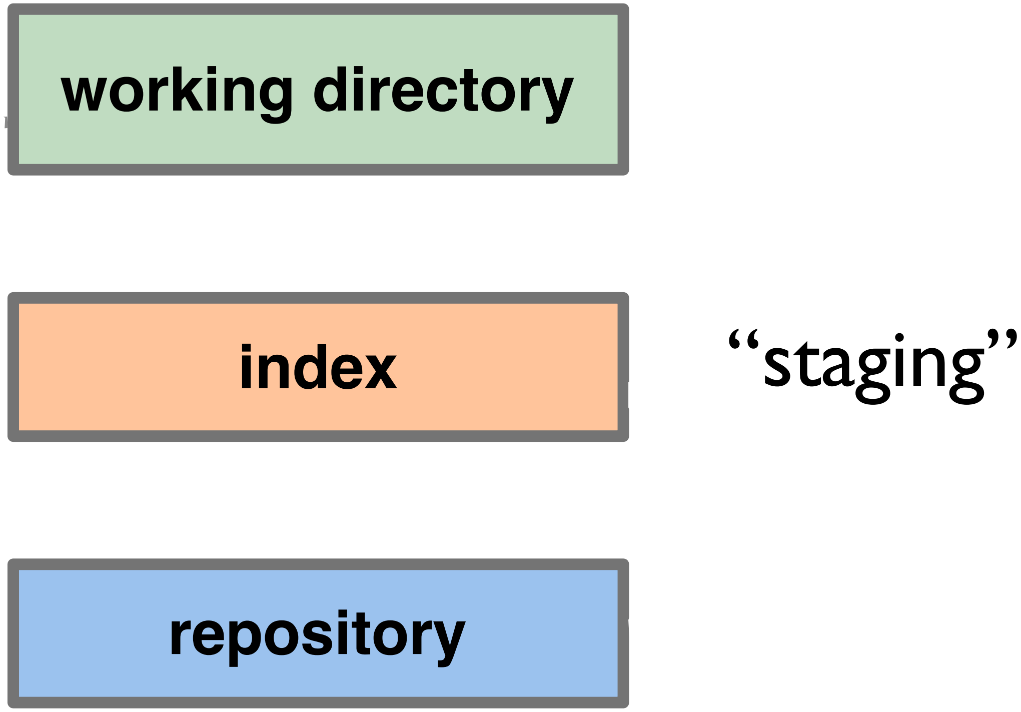


working directory

index

object database

repository



working directory

index

“staging”

repository

A Basic Workflow

Edit files

Stage the changes

Review your changes

Commit the changes

$ edit hello_world.py

```
#!/usr/bin/env python
"""Hello world program
"""
import sys

def hello(name):
    print 'Hello {}'.format(name)

def main():
    name = 'world'
    if (len(sys.argv) > 1):
        name = sys.argv[1]
    hello(name)

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

$ edit hello_world.py

```
#!/usr/bin/env python
"""Hello world program
"""
import sys

# this program prints out ‘hello world’

def hello(name):
    print 'Hello {}'.format(name)

def main():
    name = 'world'
    if (len(sys.argv) > 1):
        name = sys.argv[1]
    hello(name)

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

git status

```
$ git status
```

$ git status

```
bash-4.2$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#      modified:   hello_world.py
#
no changes added to commit (use "git add" and/or "git commit -a")
```

$ git status

```
bash-4.2$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   hello_world.py
#
no changes added to commit (use "git add" and/or "git commit -a")
```

$ git status

```
bash-4.2$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#      modified:   hello_world.py
#
no changes added to commit (use "git add" and/or "git commit -a")
```

$ git status

```
bash-4.2$ git status
# On branch master
# Changes NOT STAGED  for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   hello_world.py
#
no changes added to commit (use "git add" and/or "git commit -a")
```

$ git status

```
bash-4.2$ git status
# On branch master
# Changes NOT STAGED for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#      modified:   hello_world.py
#
no changes added to commit (use "git add" and/or "git commit -a")
```

A Basic Workflow

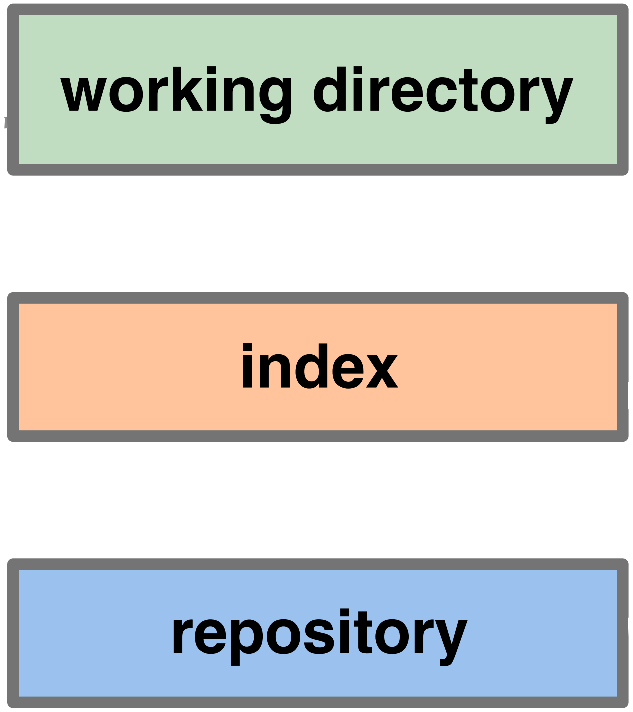
Edit files

Stage the changes

Review your changes

Commit the changes

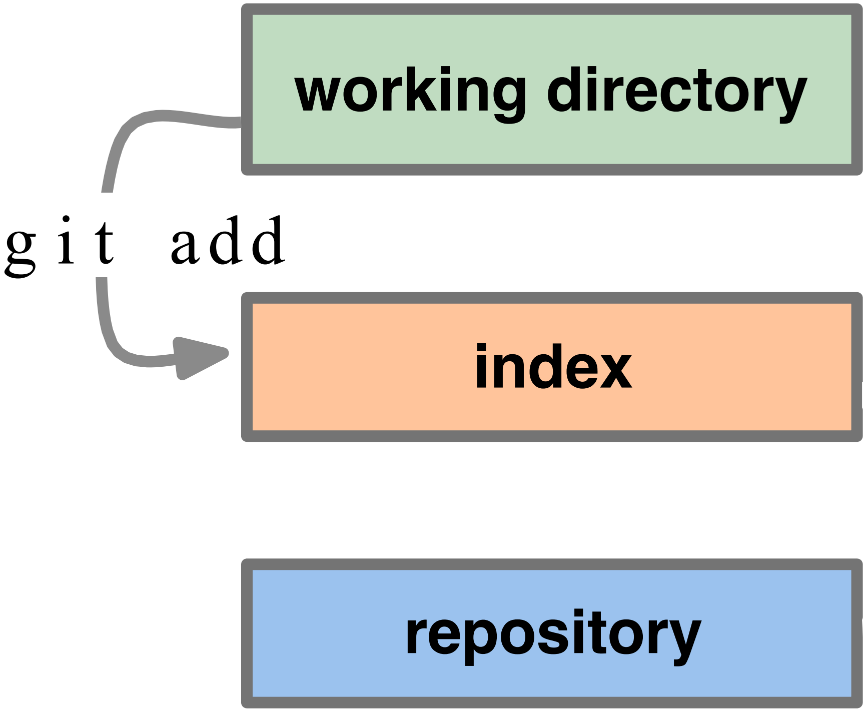
git add



working directory

index

repository


working directory
git add
index
repository

```
$ git add hello_world.py
$ git status
```

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#	modified:   hello_world.py
#
```

```
$ git status

# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#  modified:   hello_world.py
#
```

```
$ git status

# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#  modified:   hello_world.py
#
```

```
$ git status

# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#  modified:   hello_world.py
#
```

staged

$ vim hello_world.py

```
#!/usr/bin/env python
"""Hello world program
"""
import sys

def hello(name):
    print 'Hello {}'.format(name)
    print '... and a good day to you!'

def main():
    name = 'world'
    if (len(sys.argv) > 1):
        name = sys.argv[1]
    hello(name)

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

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#  modified:   hello_world.py
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in
working directory)
#
#  modified:   hello_world.py
#
```

Staged

```
#!/usr/bin/env python
"""Hello world program
"""
import sys

def hello(name):
    print 'Hello {}'.format(name)

def main():
    name = 'world'
    if (len(sys.argv) > 1):
        name = sys.argv[1]
    hello(name)

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

In Working Directory

```
#!/usr/bin/env python
"""Hello world program
"""
import sys

def hello(name):
    print 'Hello {}'.format(name)
    print '... and a good day to you!'

def main():
    name = 'world'
    if (len(sys.argv) > 1):
        name = sys.argv[1]
    hello(name)

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

Staged

```
#!/usr/bin/env python
"""Hello world program
"""
import sys

def hello(name):
    print 'Hello {}'.format(name)

def main():
    name = 'world'
    if (len(sys.argv) > 1):
        name = sys.argv[1]
    hello(name)

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

In Working Directory

```
#!/usr/bin/env python
"""Hello world program
"""
import sys

def hello(name):
    print 'Hello {}'.format(name)
    print '... and a good day to you!'

def main():
    name = 'world'
    if (len(sys.argv) > 1):
        name = sys.argv[1]
    hello(name)

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 hello_world.py
$ git status
```

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#  modified:   hello_world.py
#
#
```

A Basic Workflow

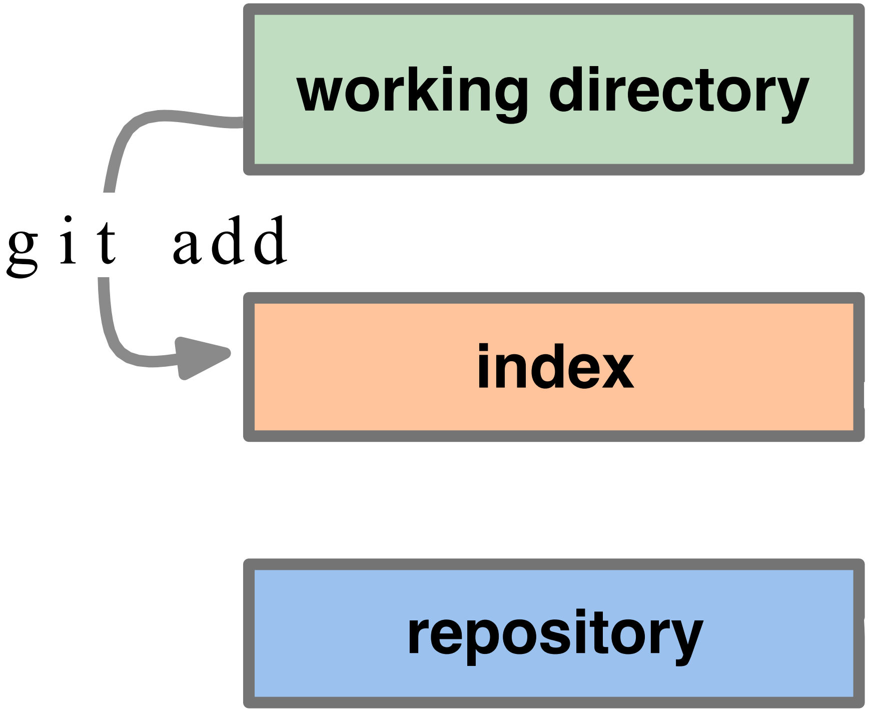
Edit files

Stage the changes

Review your changes

Commit the changes

git commit

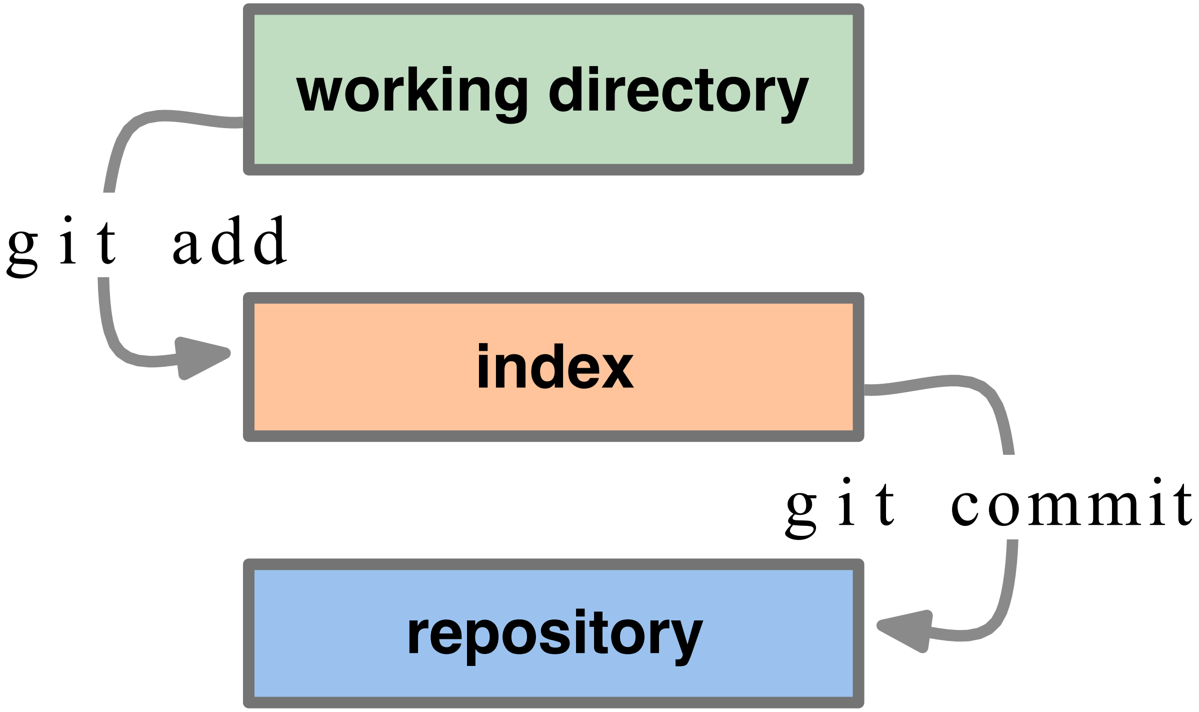


working directory

git add

index

repository



working directory

git add

index

git commit

repository

```
$ git commit
```

```

# 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:   hello_world.py
#
```

```
$ 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:   hello_world.py
#
```

```
$ 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:   hello_world.py
#
```

```
$ git commit

[master 9df86bd] A descriptive commit message
 1 file changed, 17 insertions(+)
```

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)

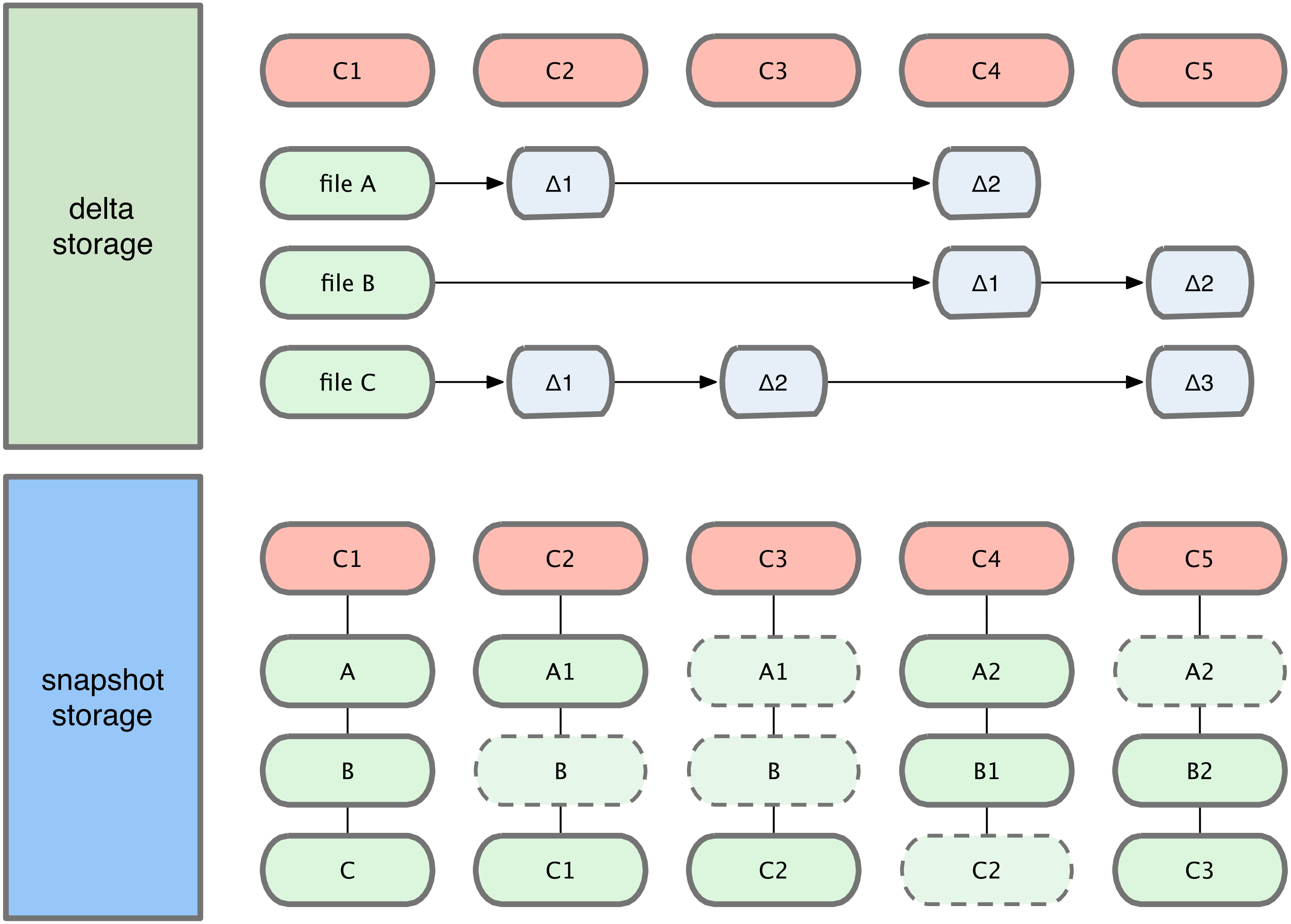
A Basic Workflow

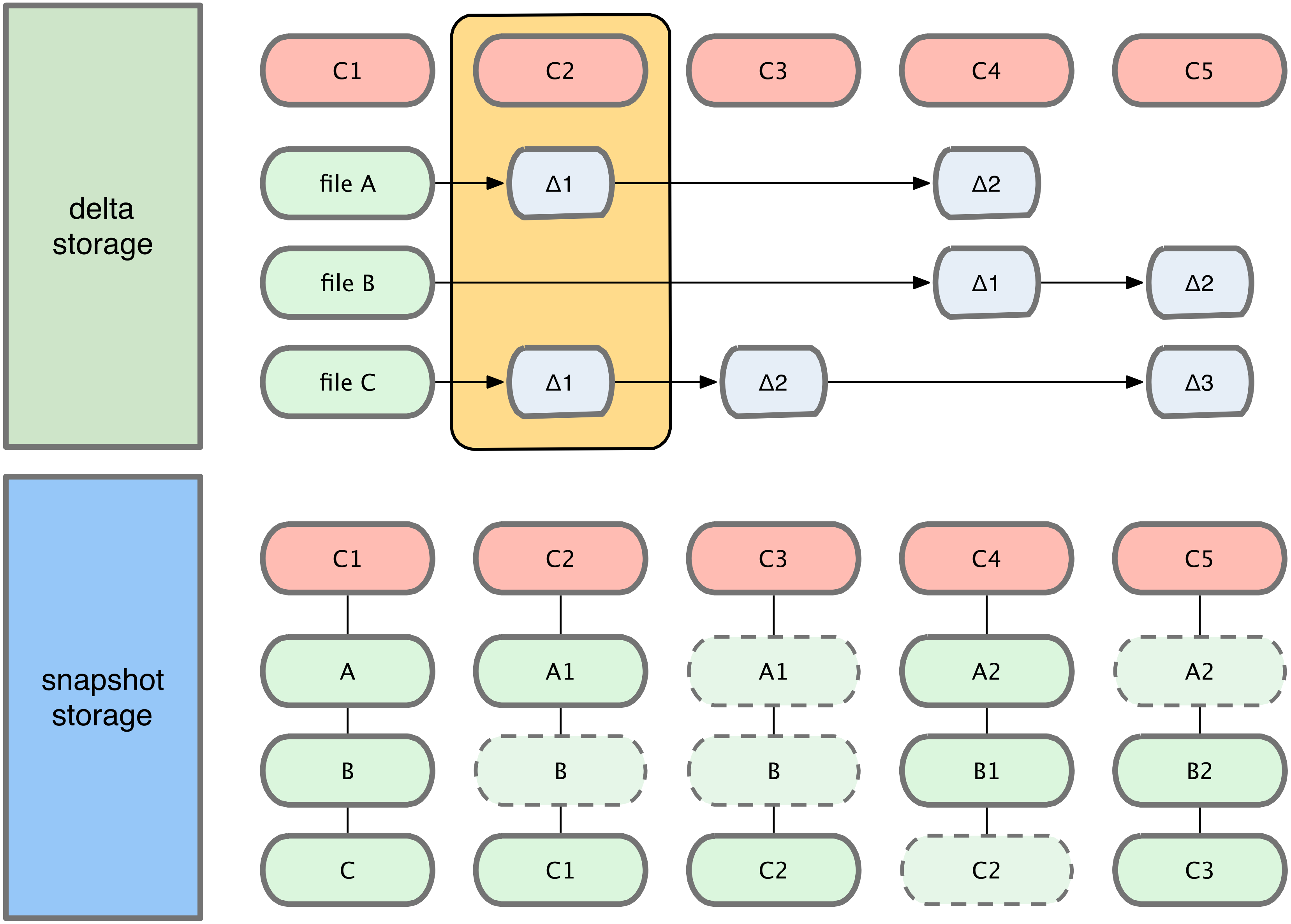
Edit files	vim / emacs / etc
Stage the changes	git add (file)
Review your changes	git status

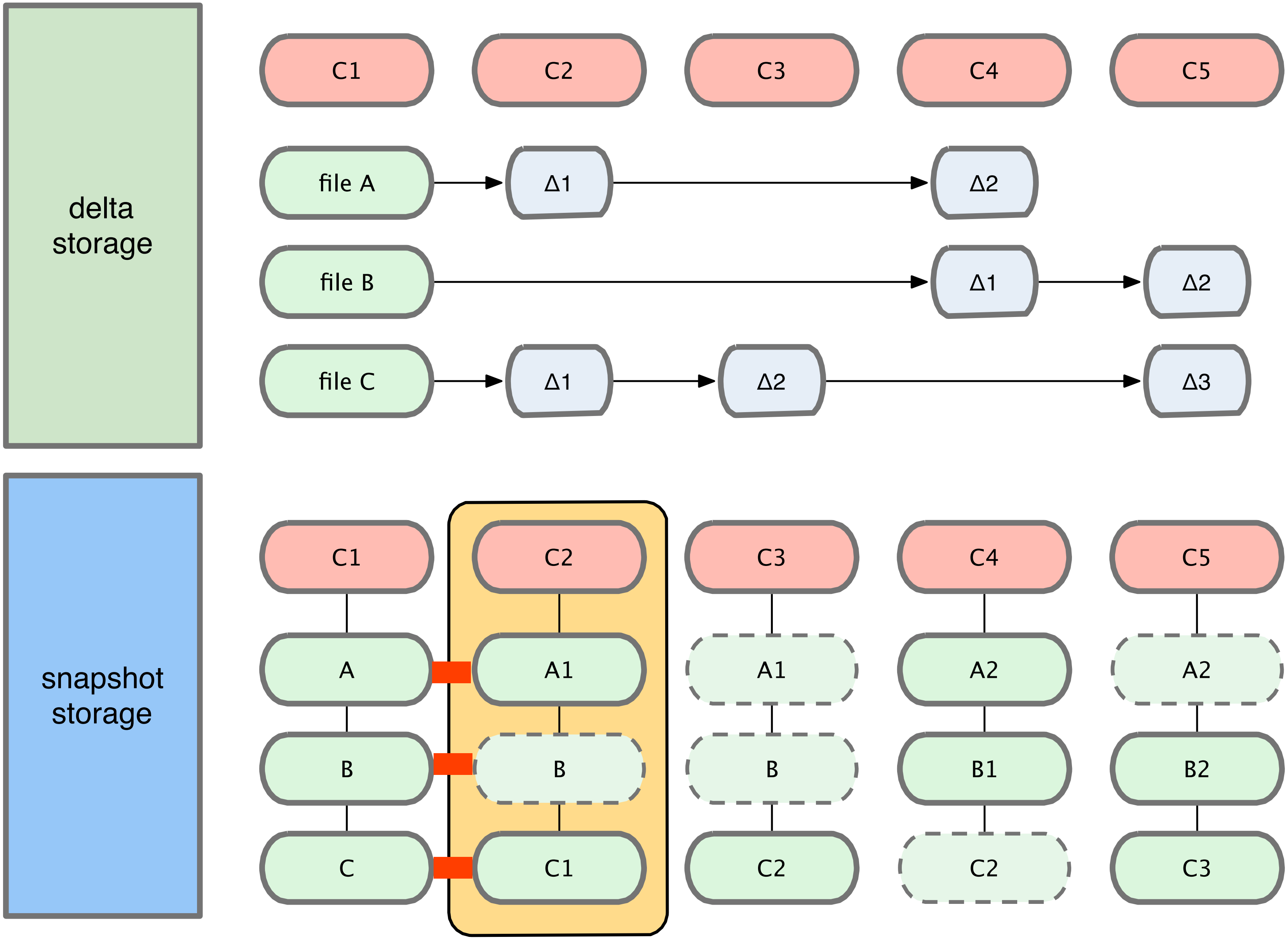
A Basic Workflow

Edit files	vim / emacs / etc
Stage the changes	git add (file)
Review your changes	git status
Commit the changes	git commit

Changes


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
C2
C3
C4
C5
A
A1
A1
A2
A2
B
B
B
B1
B2
C
C1
C2
C2
C3


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
A
B
C
C2
A1
B
C1
C3
A1
B
C2
C4
A2
B1
C2
C5
A2
B2
C3


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
snapshot storage
C1
C2
C3
C4
C5
A
A1
A1
A2
A2
B
B
B
B1
B2
C
C1
C2
C2
C3

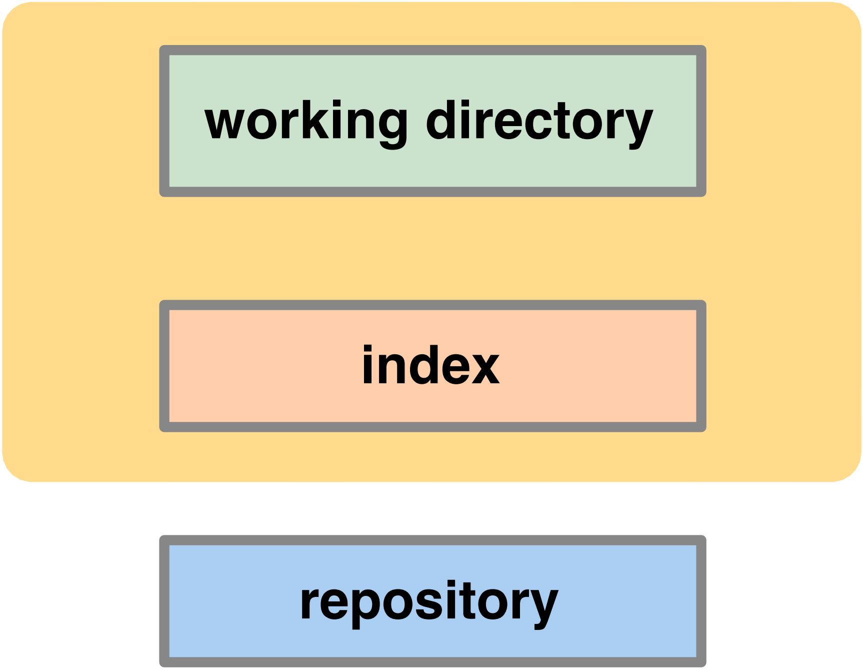
git diff

```
diff --git a/main.py b/
main.py index 6db8b97..b9bcc62
199755
--- 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



working directory

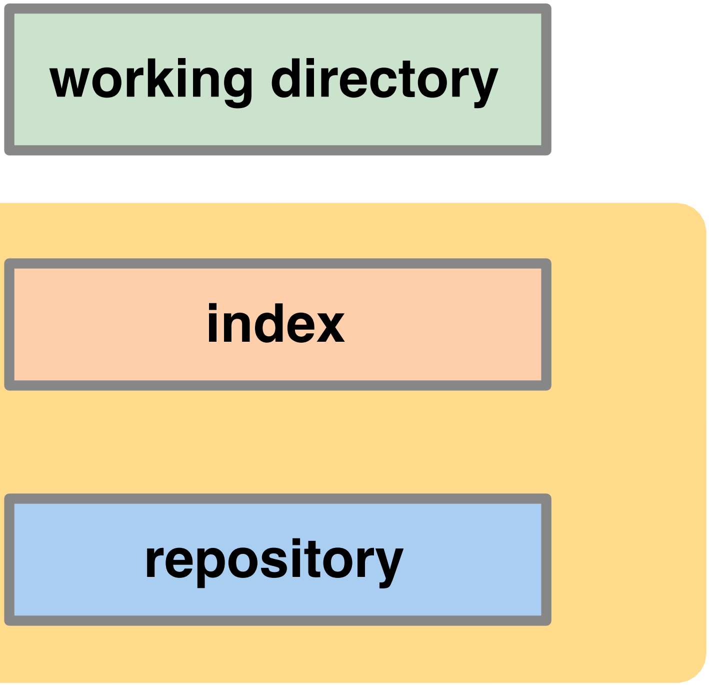
index

repository

What is staged?

```
git diff --cached
```

`git diff --cached`


working directory

index

repository

History

git log

```
$>git log_
```

```
$>git log
```

```
bash-4.2$ git log
commit 431437d5760aada25d0f8794f40414216afd813c
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:16:16 2013 -0700

    a descriptive commit message

commit e261ed6010585c32e395b8c8cd499015fc31345b
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:03:15 2013 -0700

    fix typo in main()

commit 39e22eae7bc6380108c5688764fafa8e43f61eae
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:02:20 2013 -0700

    initial commit of hello_world
```

```
bash-4.2$ git log
commit 431437d5760aada25d0f8794f40414216afd813c
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:16:16 2013 -0700

    a descriptive commit message

commit e261ed6010585c32e395b8c8cd499015fc31345b
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:03:15 2013 -0700

    fix typo in main()

commit 39e22eae7bc6380108c5688764fafa8e43f61eae
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:02:20 2013 -0700

    initial commit of hello_world
```

```
bash-4.2$ git log
commit 431437d5760aada25d0f8794f40414216afd813c
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:16:16 2013 -0700

    a descriptive commit message

commit e261ed6010585c32e395b8c8cd499015fc31345b
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:03:15 2013 -0700

    fix typo in main()

commit 39e22eae7bc6380108c5688764fafa8e43f61eae
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:02:20 2013 -0700

    initial commit of hello_world
```

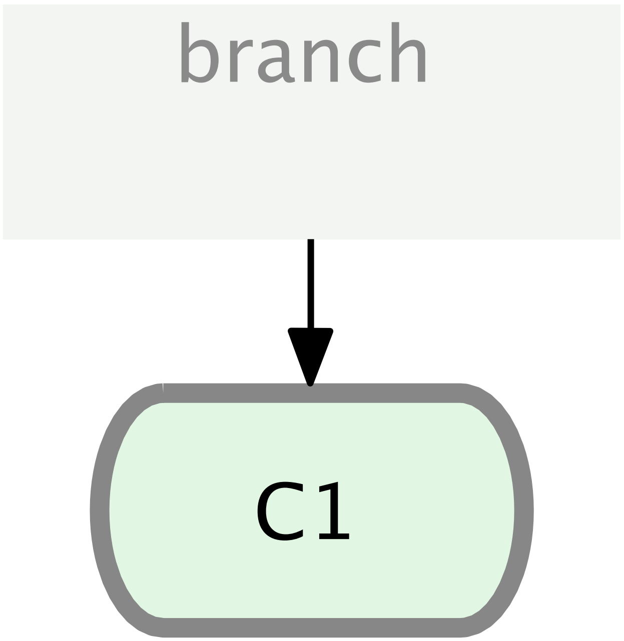
```
bash-4.2$ git log
commit 431437d5760aada25d0f8794f40414216afd813c
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:16:16 2013 -0700

    a descriptive commit message

commit e261ed6010585c32e395b8c8cd499015fc31345b
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:03:15 2013 -0700

    fix typo in main()

commit 39e22eae7bc6380108c5688764fafa8e43f61eae
Author: Joe Hamman <jhamman@hydro.washington.edu>
Date:   Sun Oct 20 15:02:20 2013 -0700

    initial commit of hello_world
```

Branching and Merging

branches

branches

lightweight, movable pointers to a commit



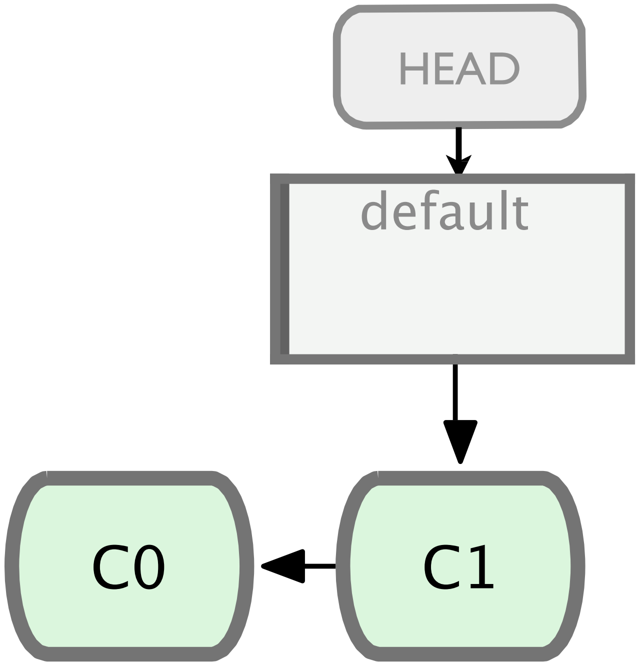
branching

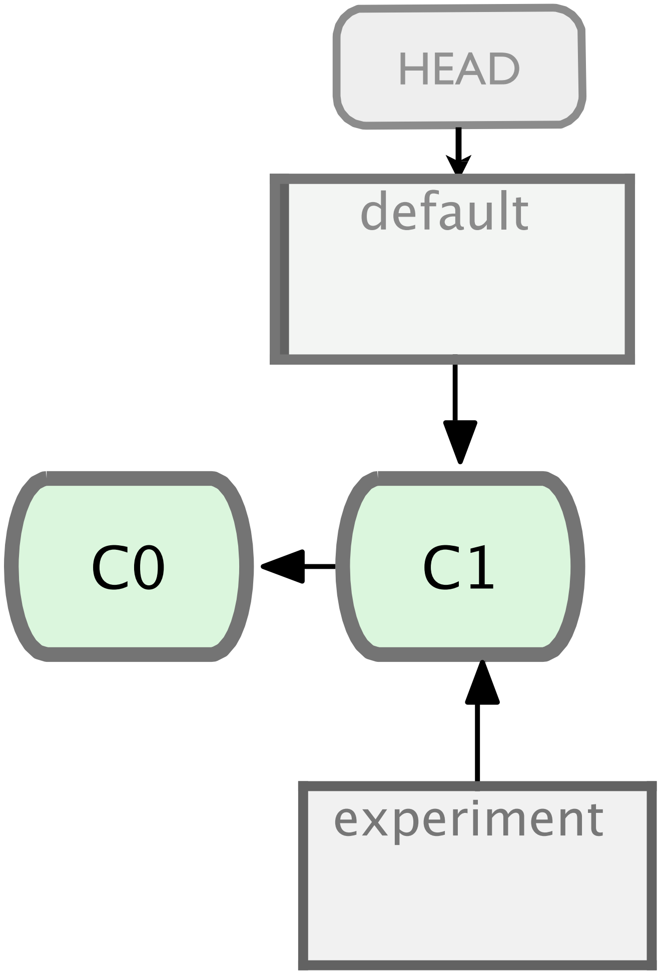
git branch

create/delete/query a branch

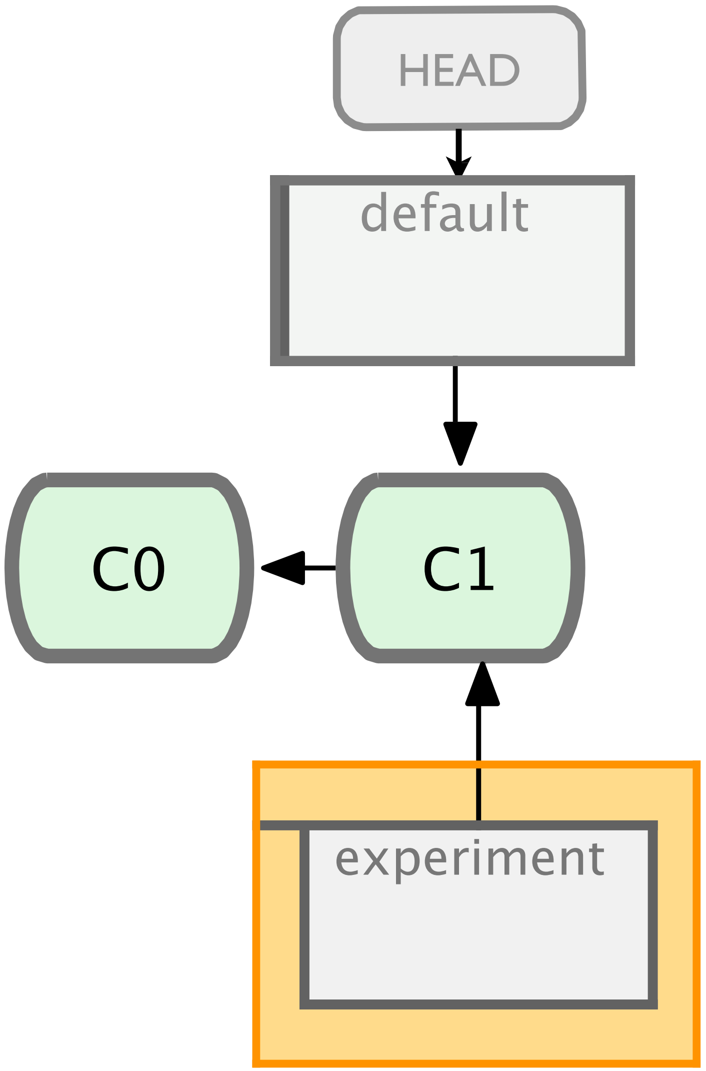
git checkout

switch to/select a branch

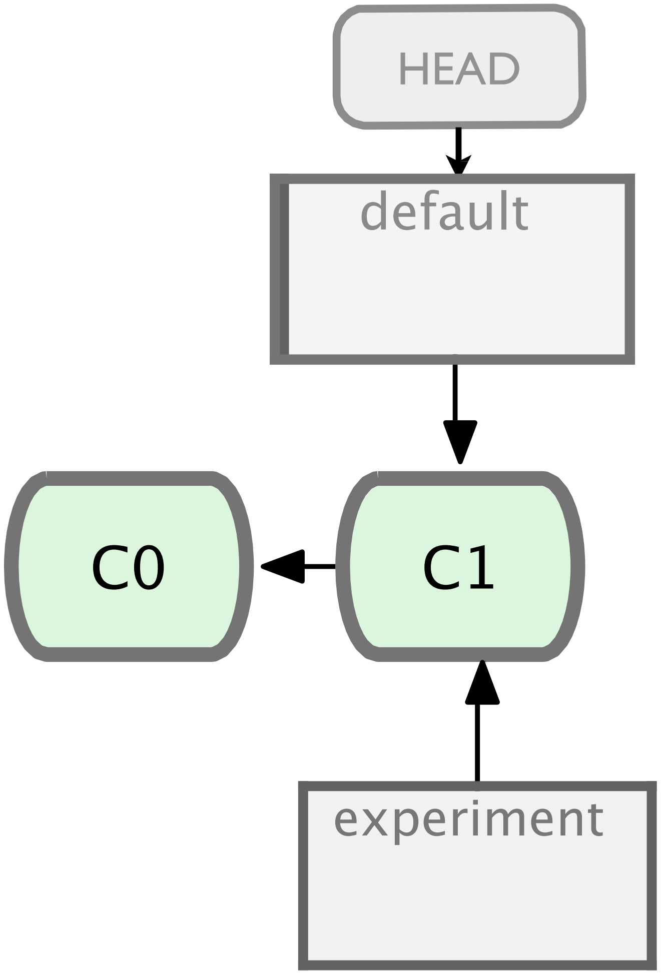

HEAD
default
C0
C1



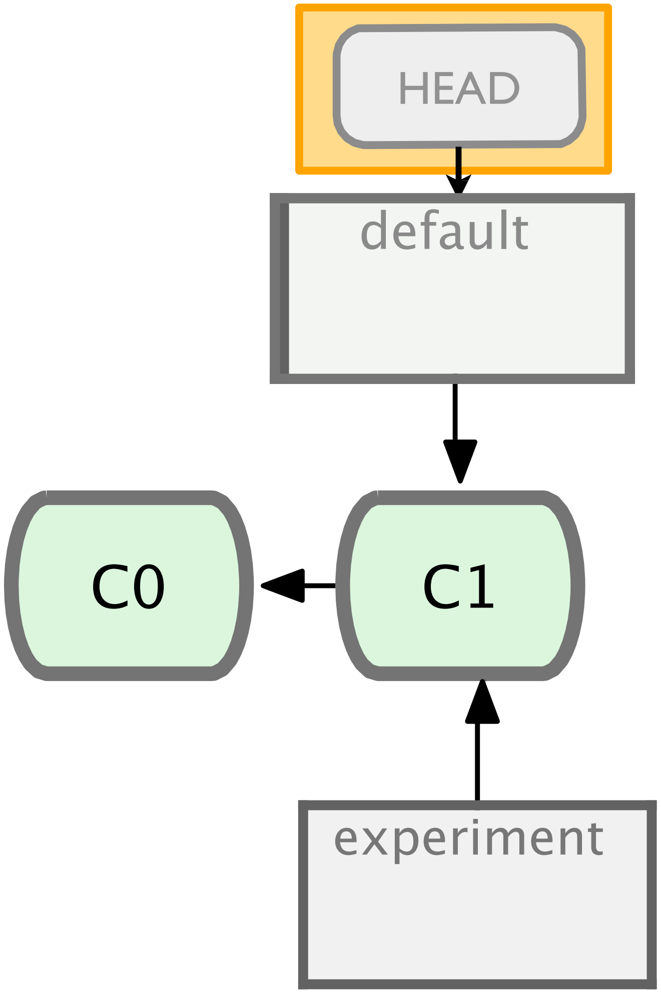
git branch experiment



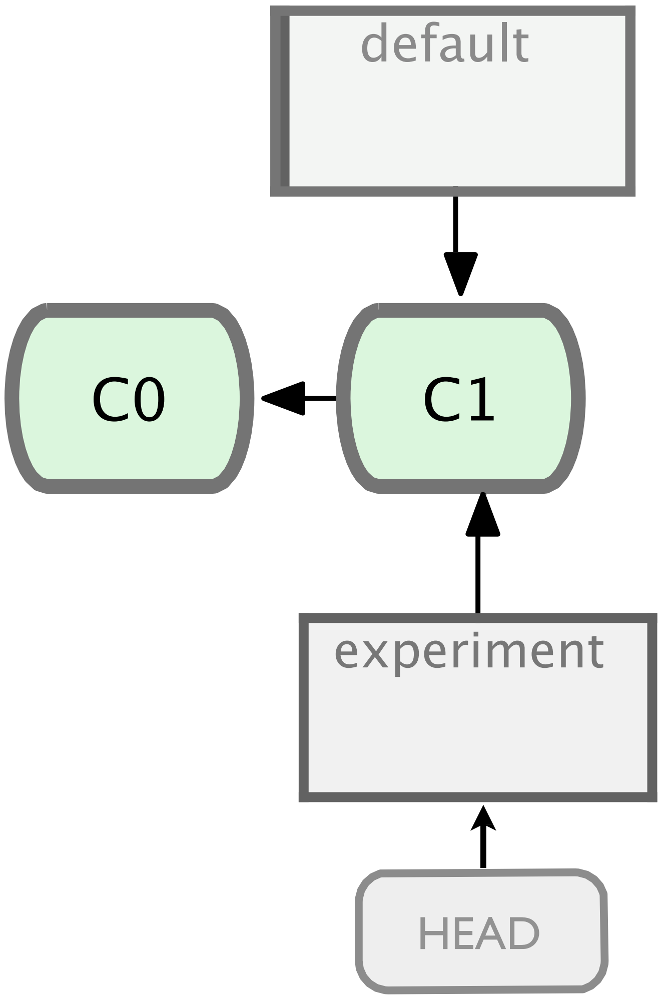
git branch experiment



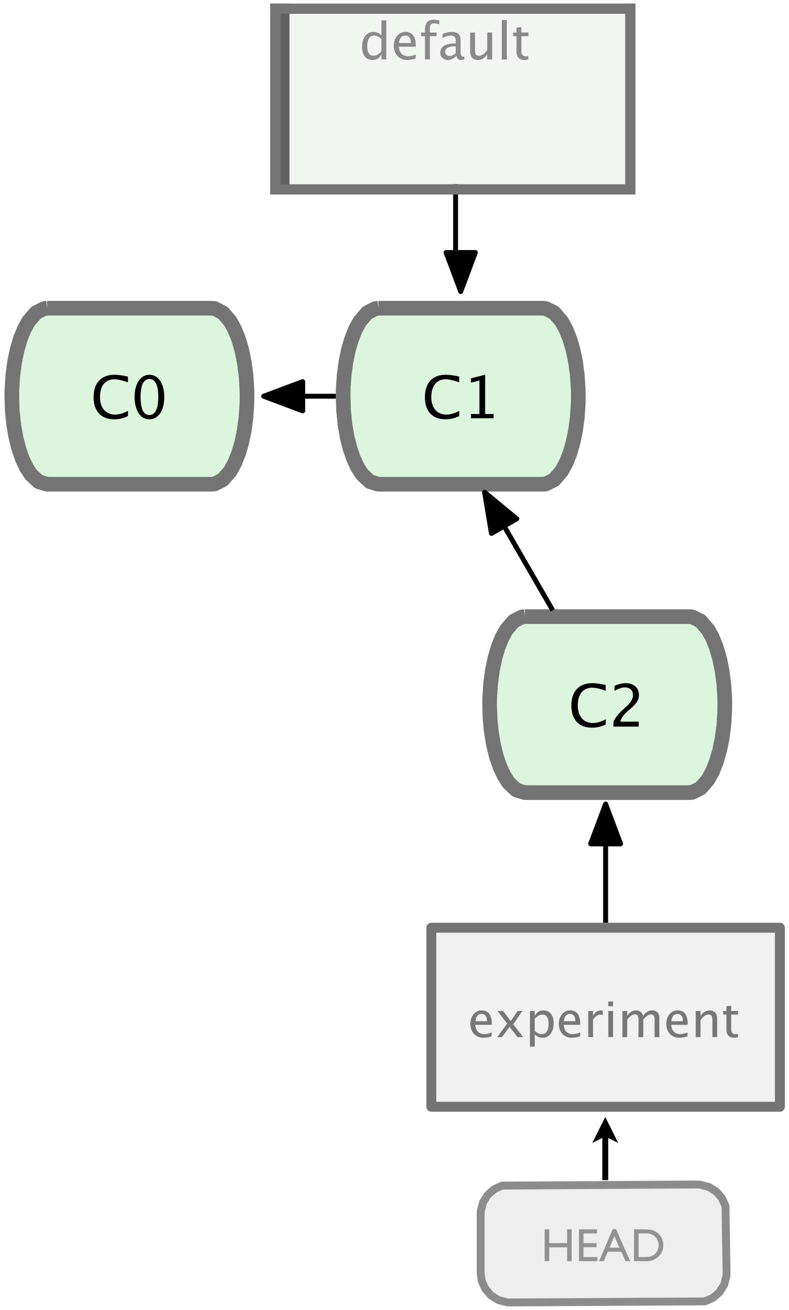
```
$ git branch
* default
  experiment
```



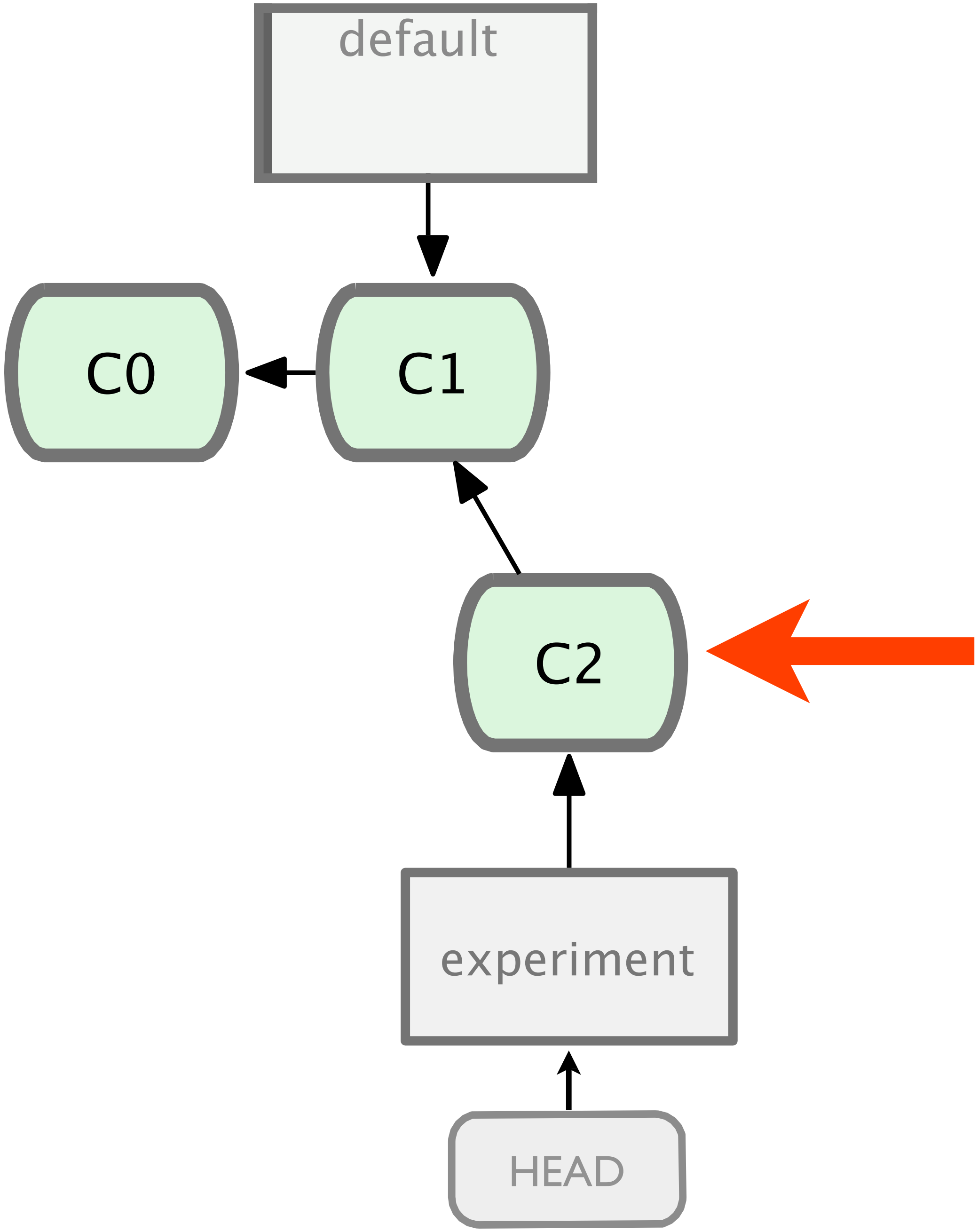
```
$ git branch
* default
  experiment
```



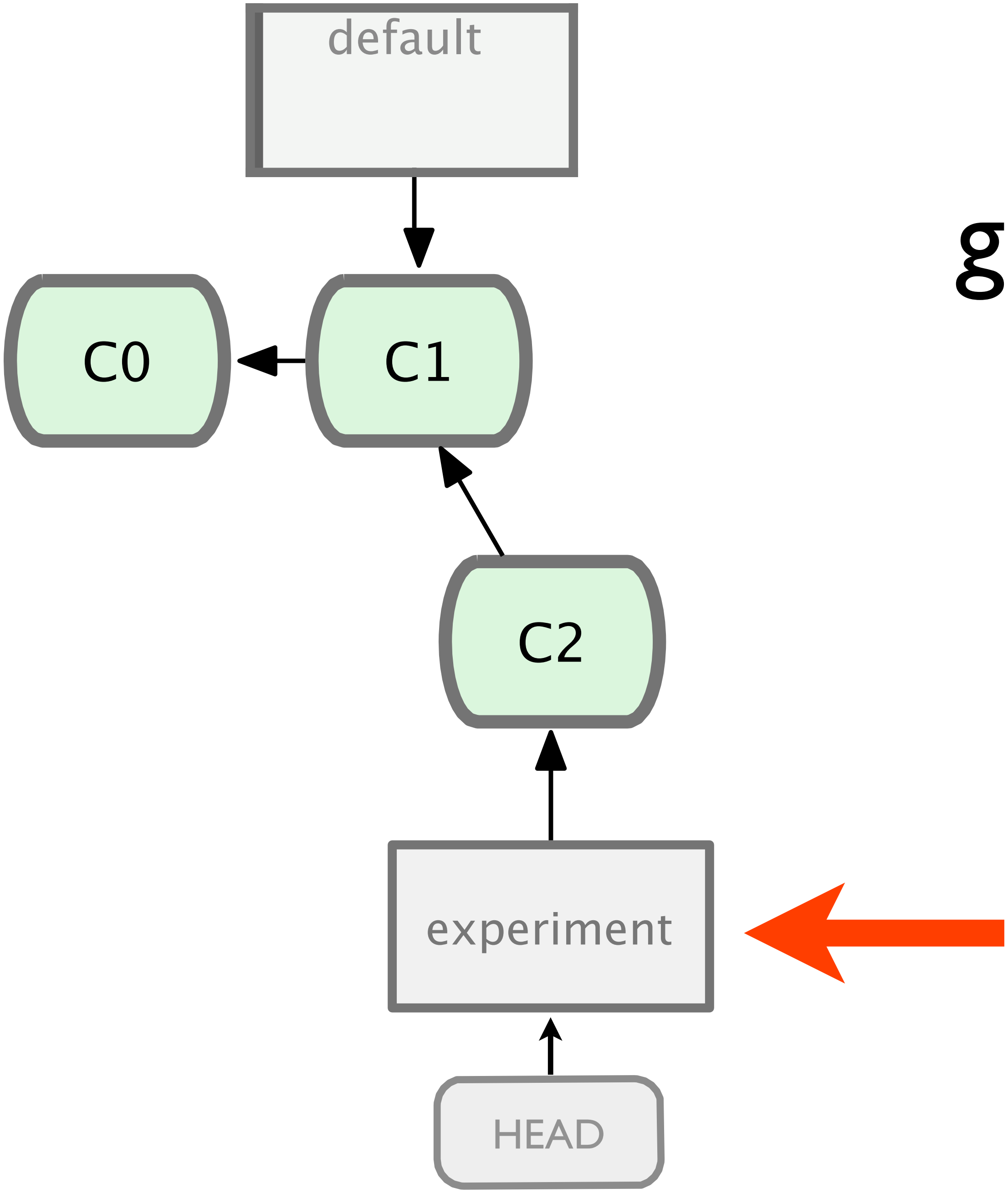
git checkout experiment



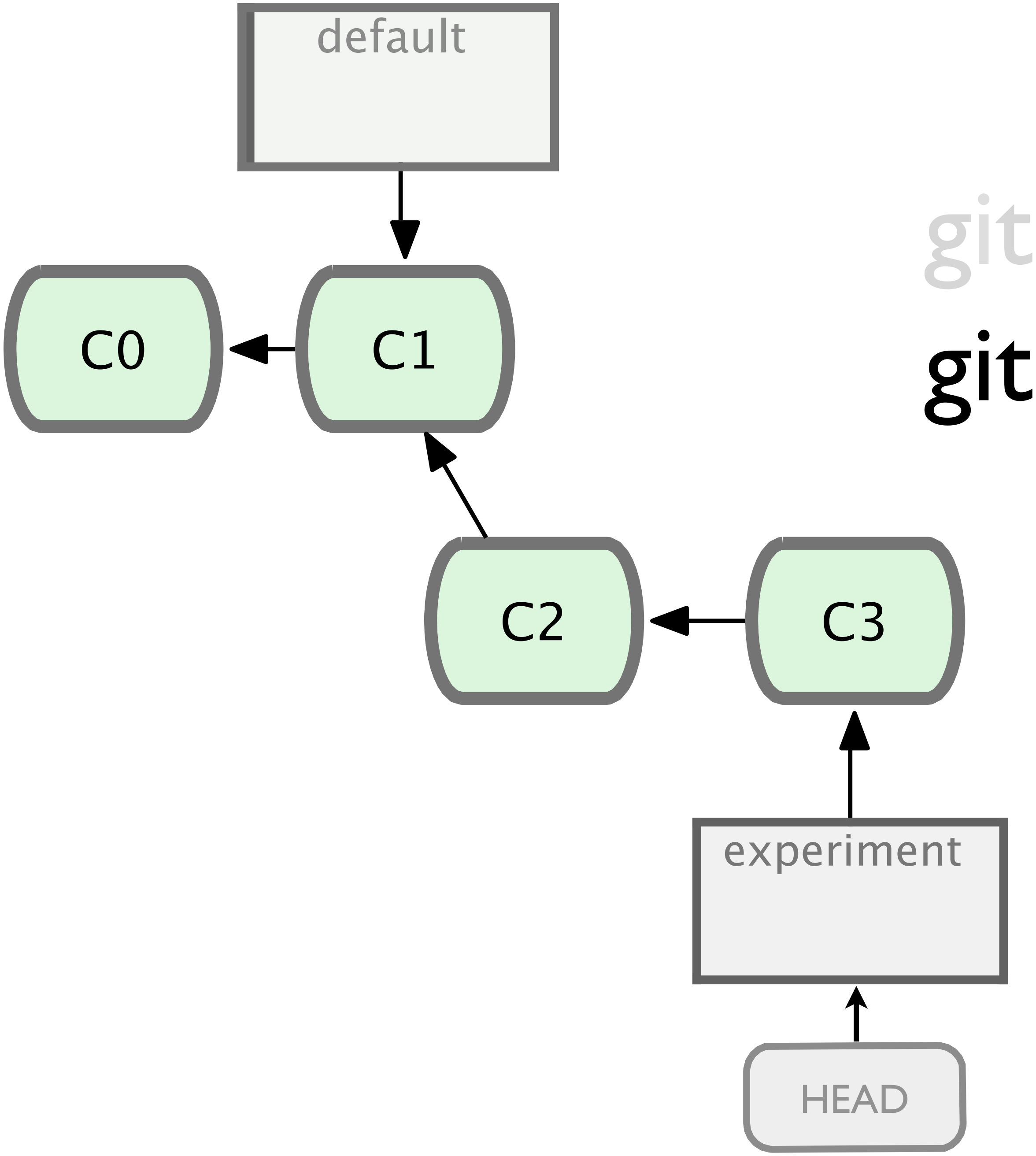
git commit



git commit

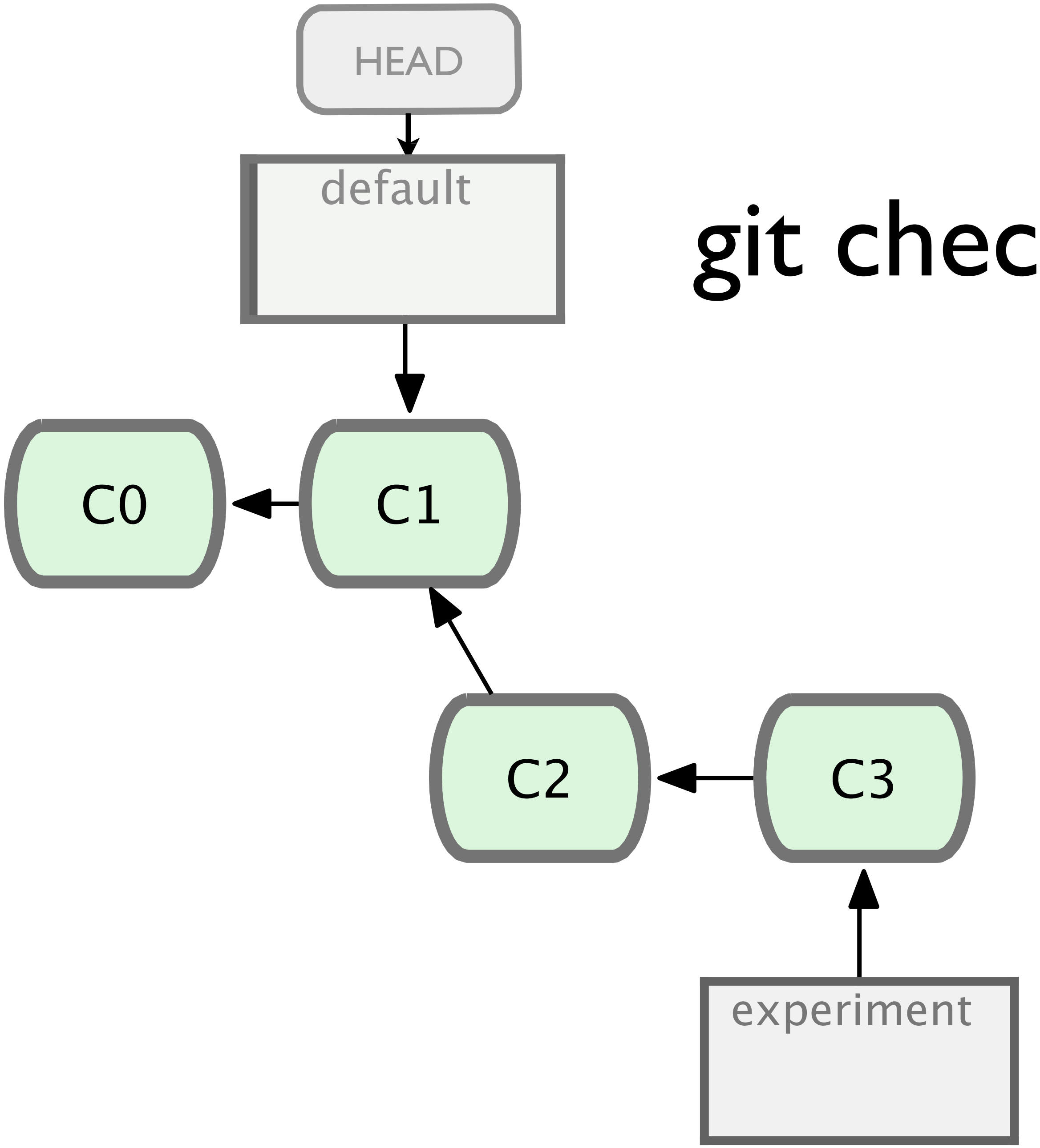


git commit



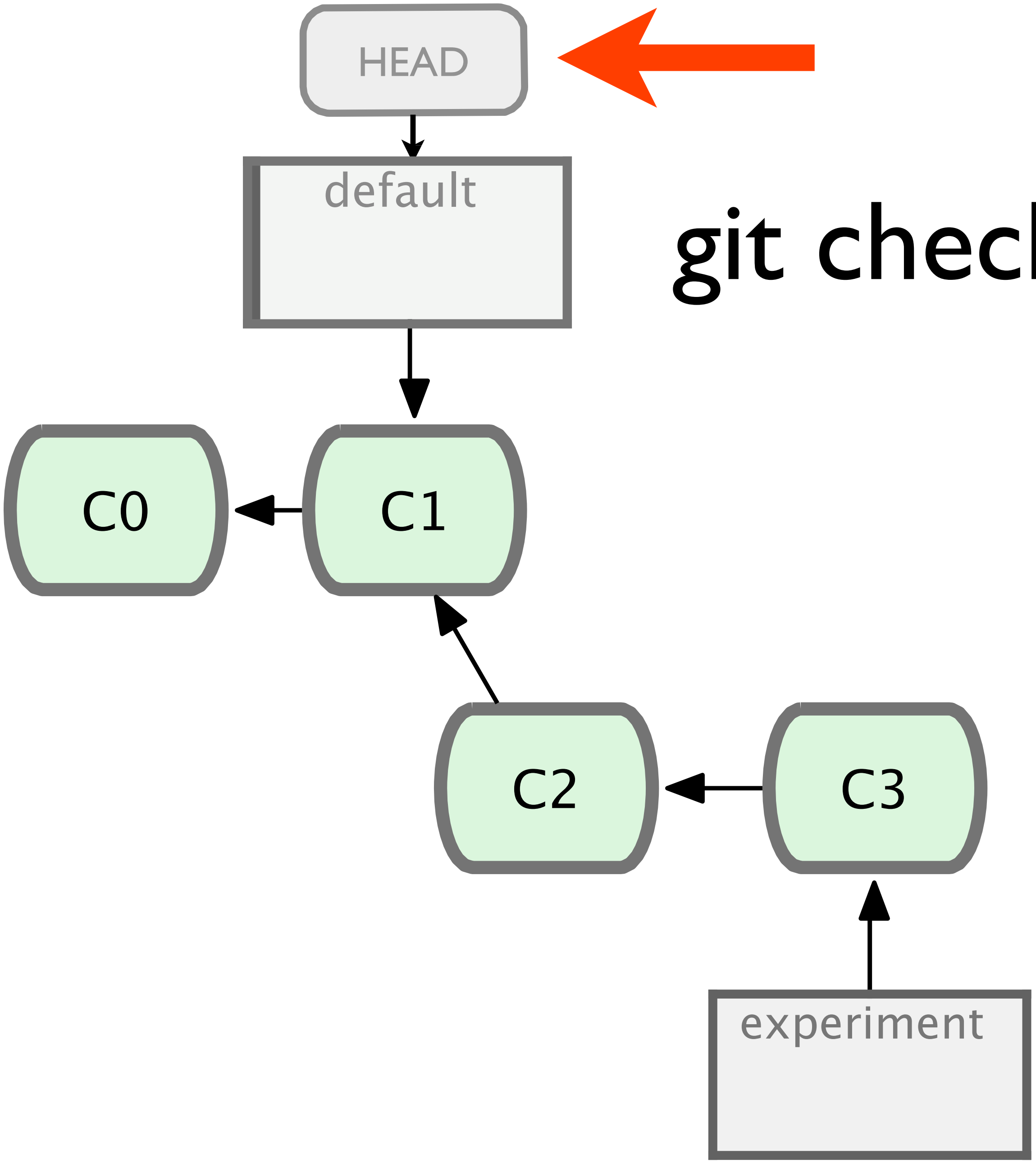
git commit

git commit

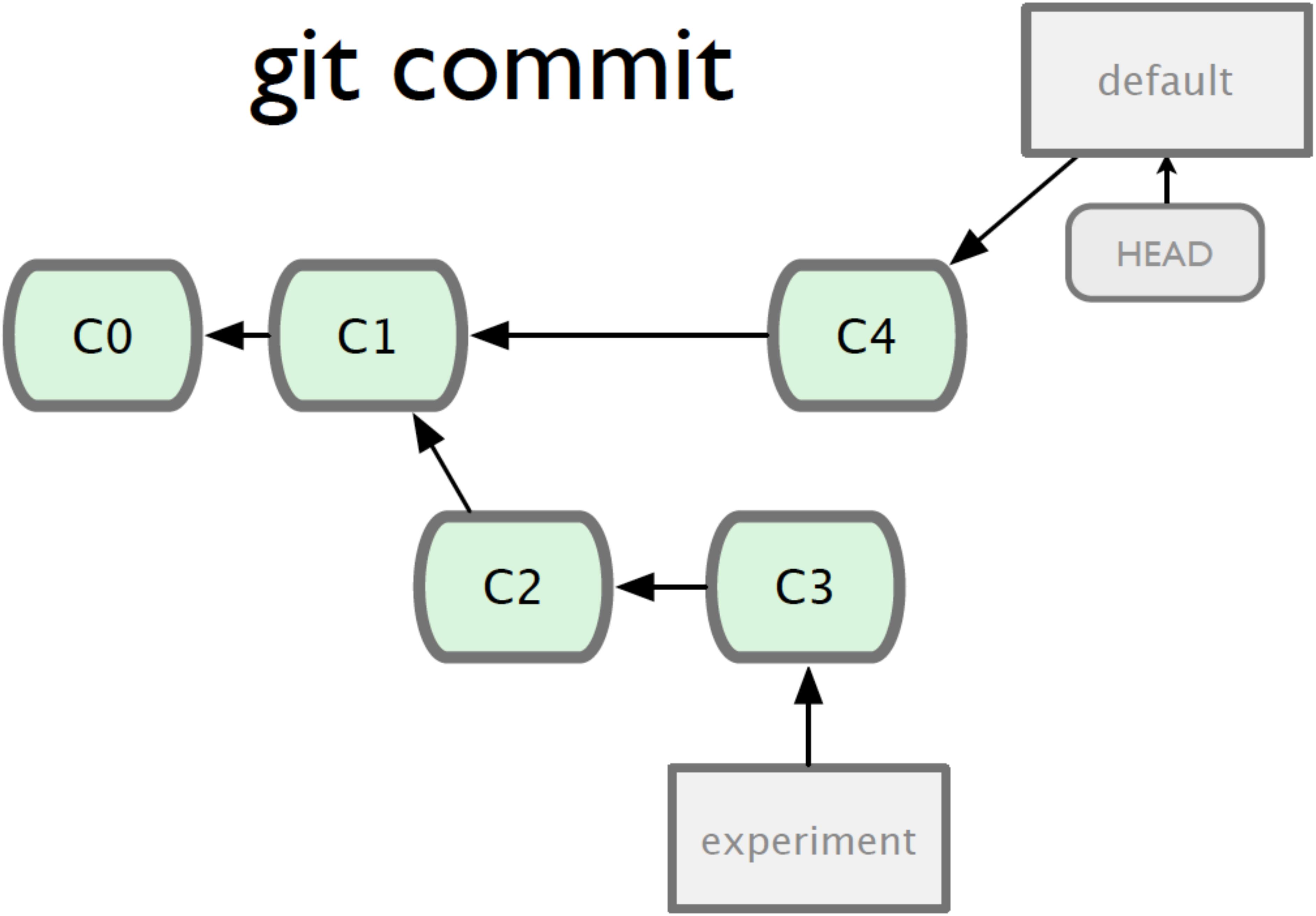


git checkout default

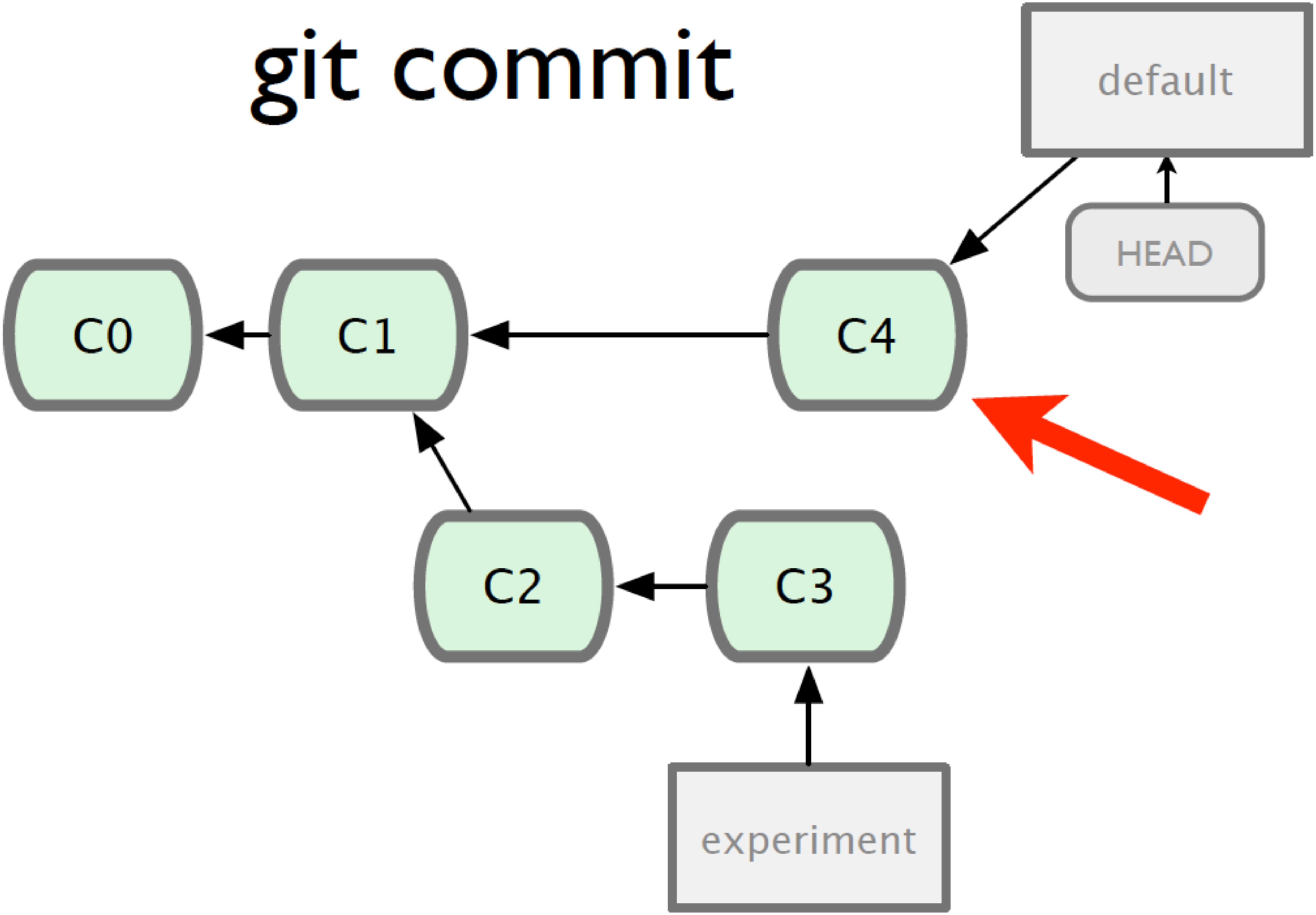
git checkout default



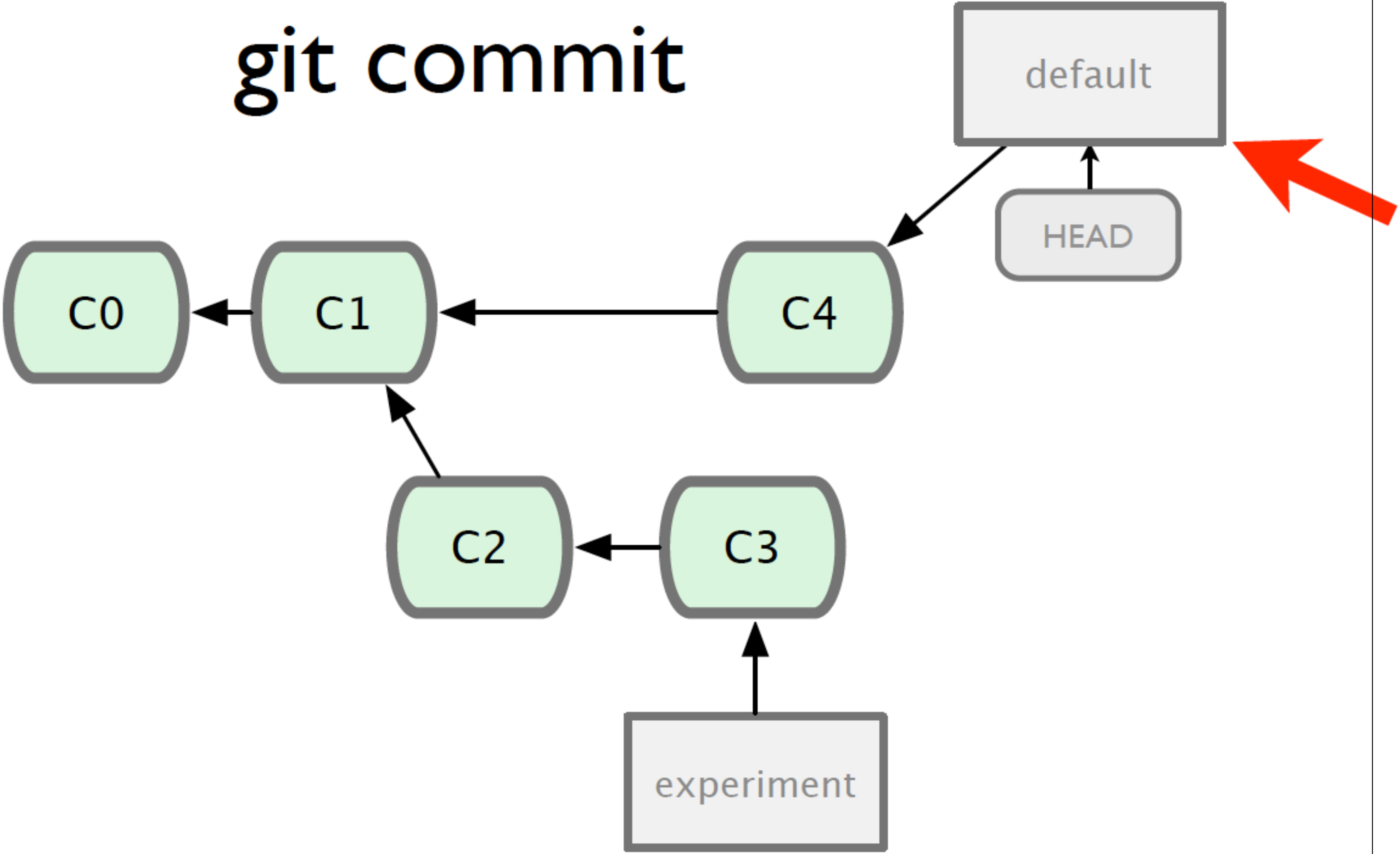
git commit

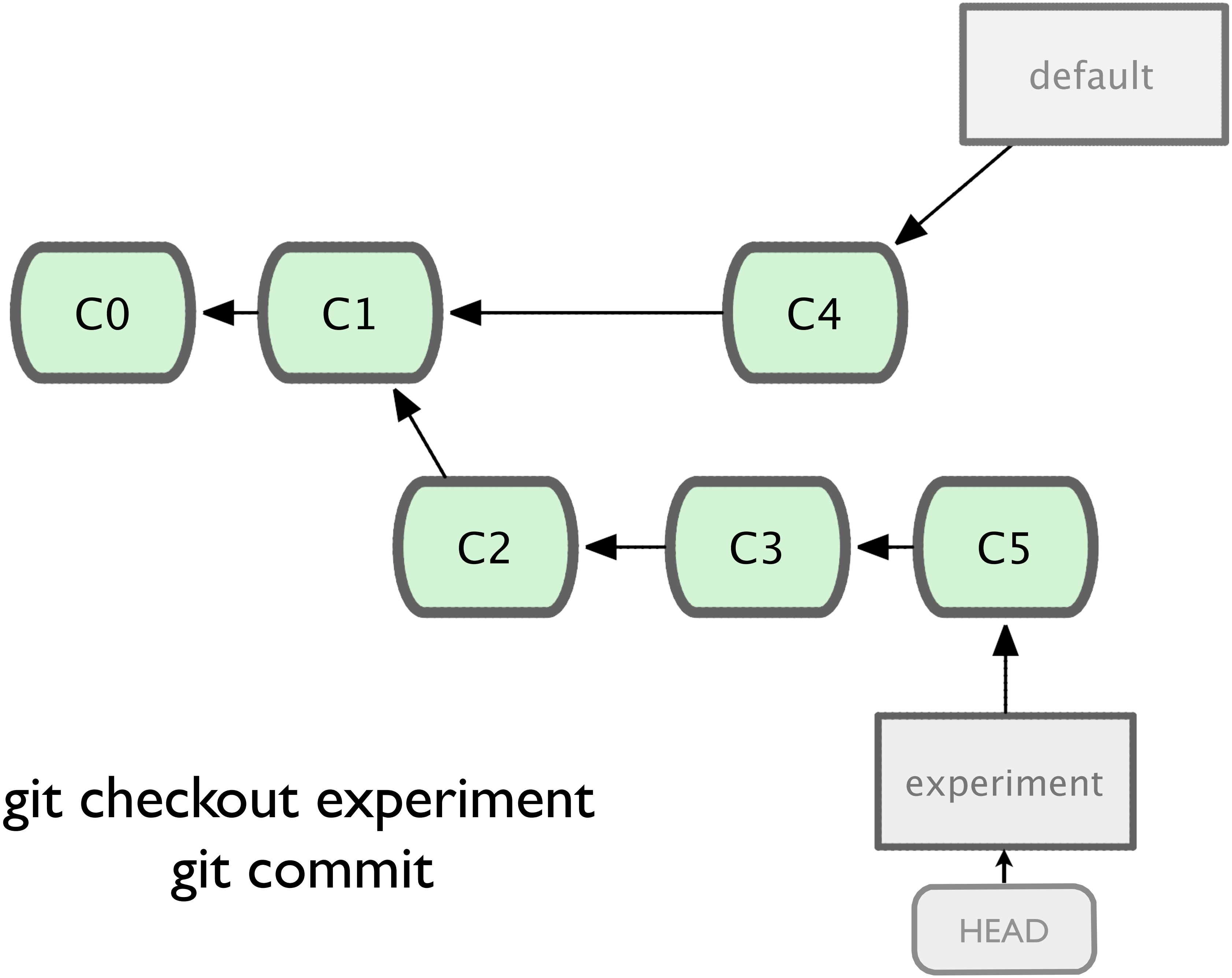


git commit



git commit



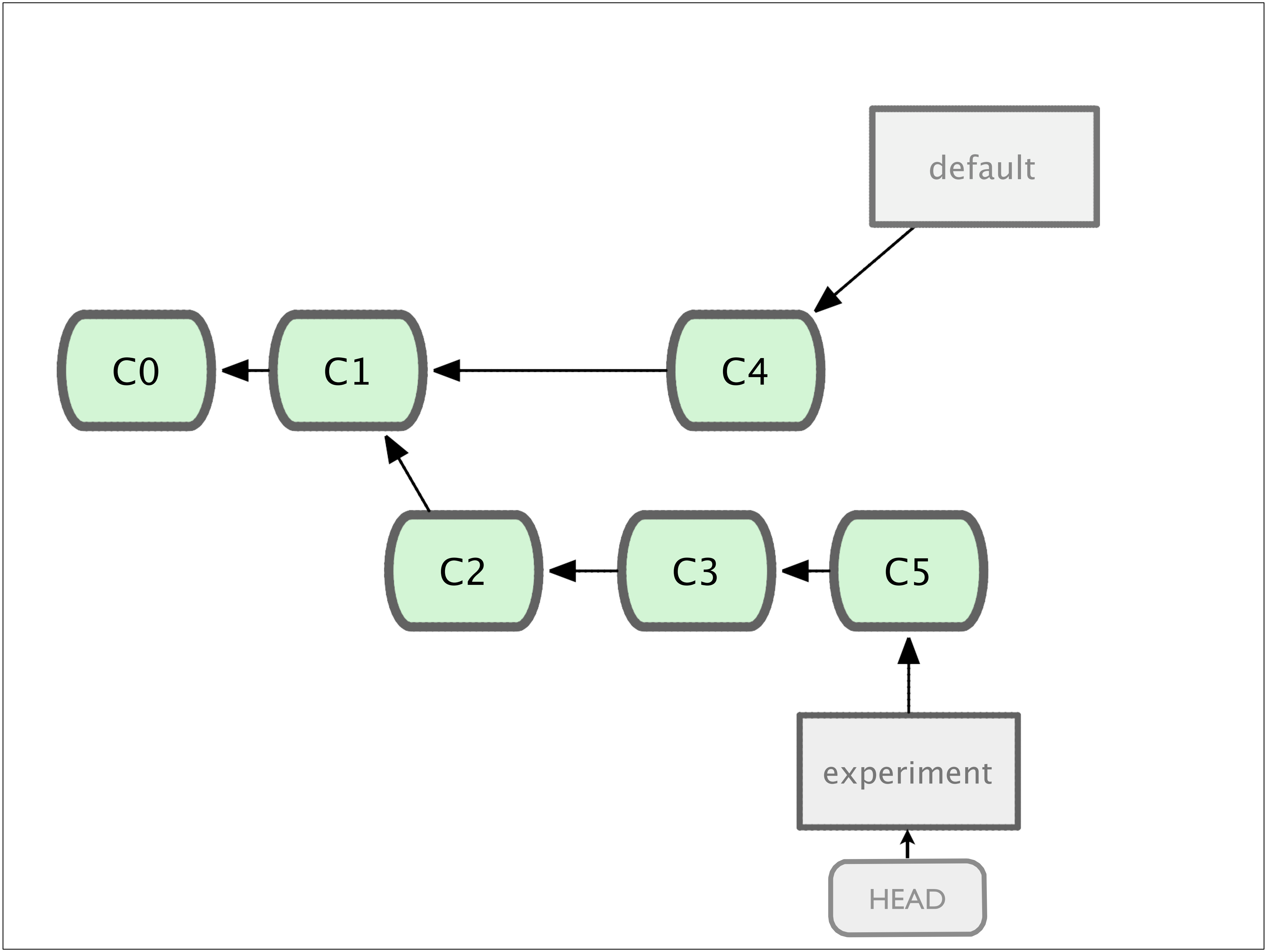


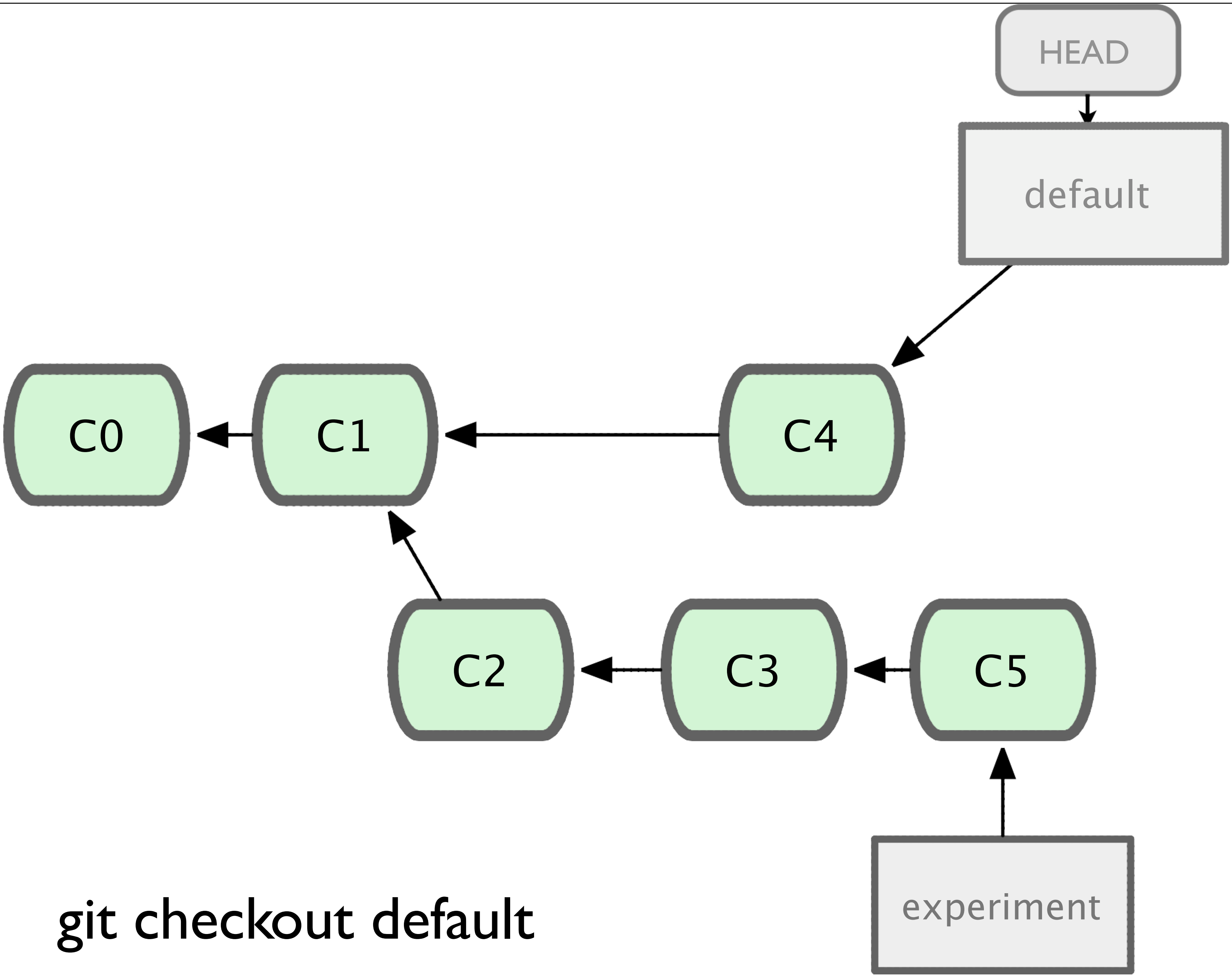
git checkout experiment
git commit

merging

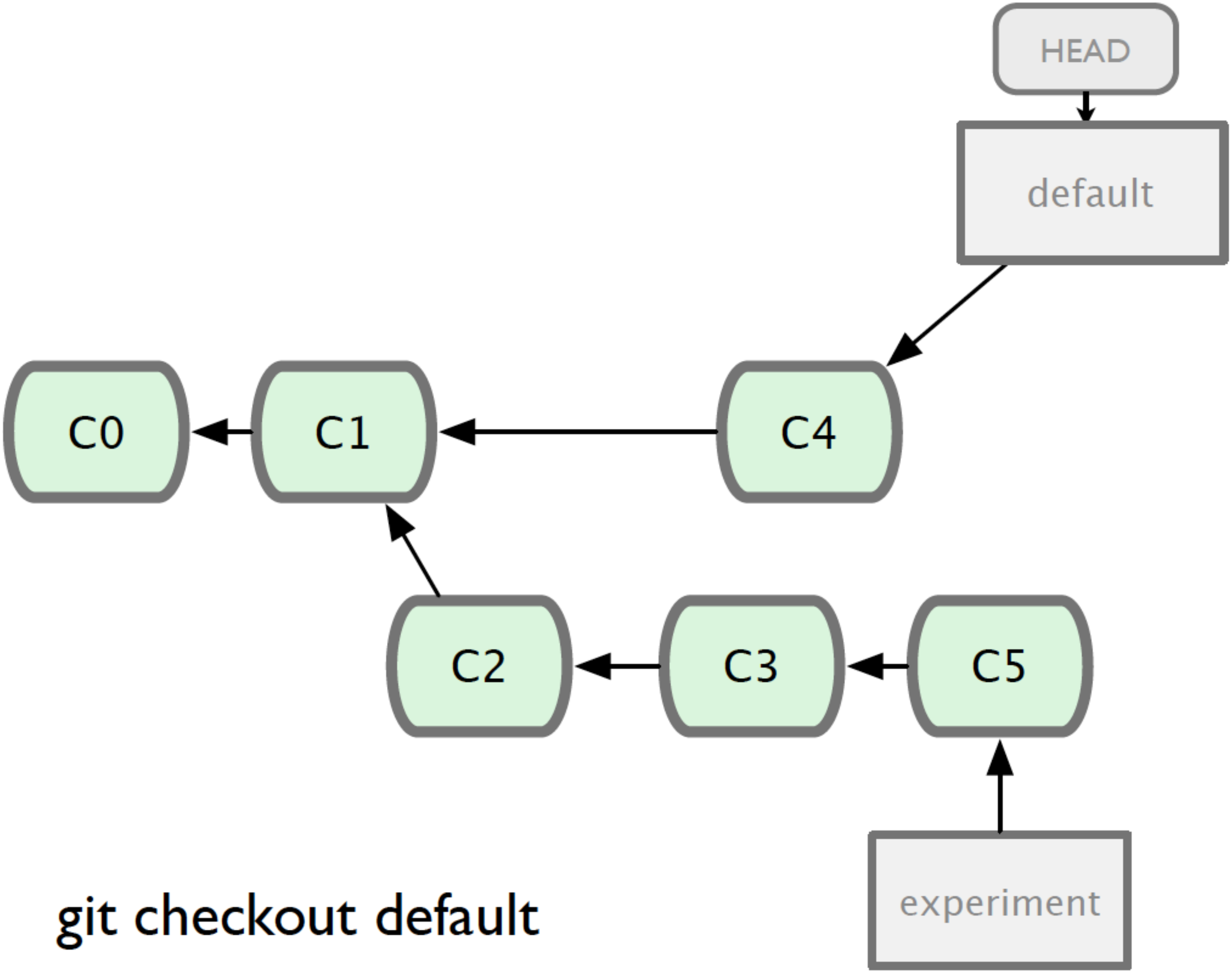
git merge

apply edits on one branch to another branch

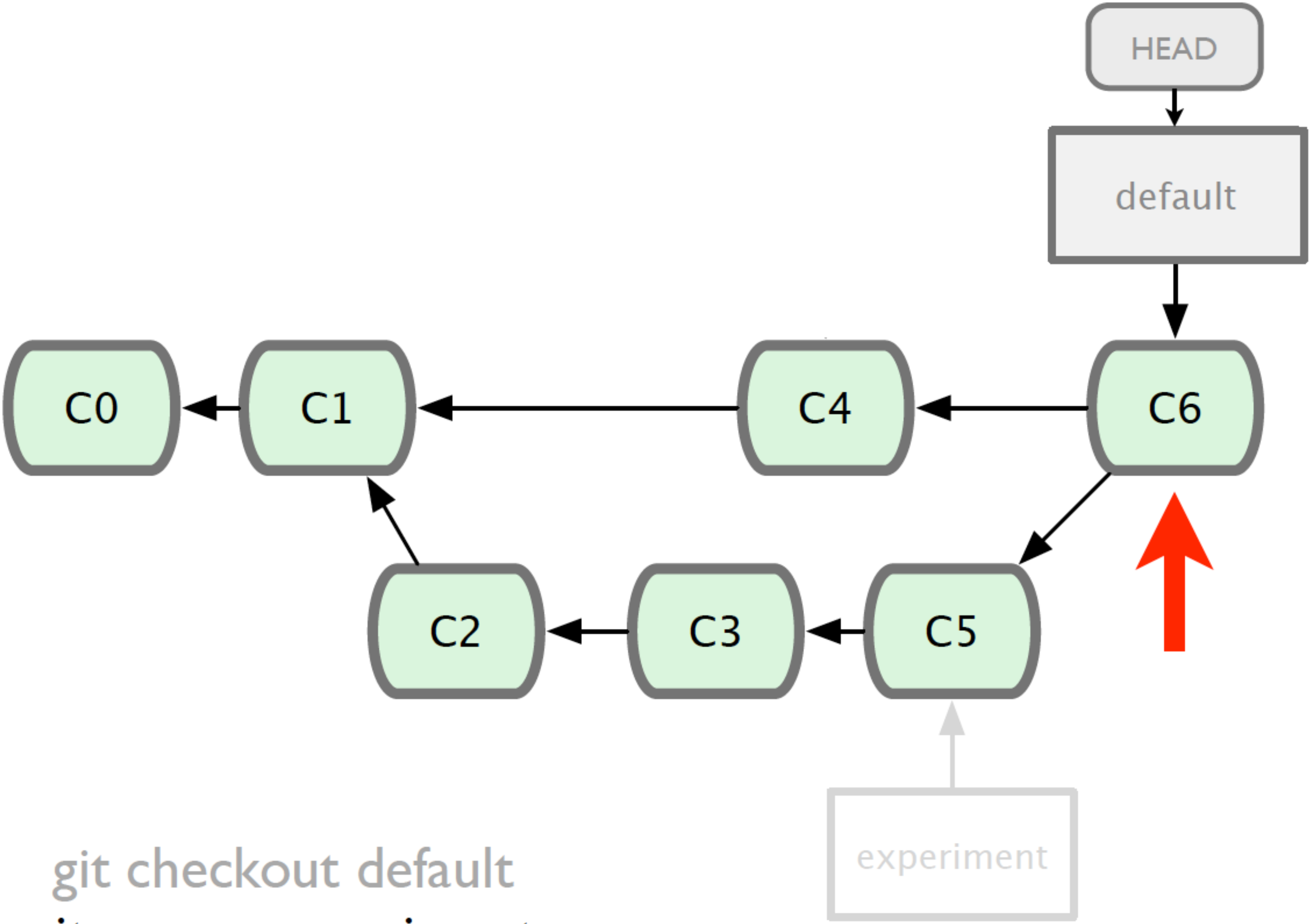

default
C0
C1
C4
C2
C3
C5
experiment
HEAD



git checkout default

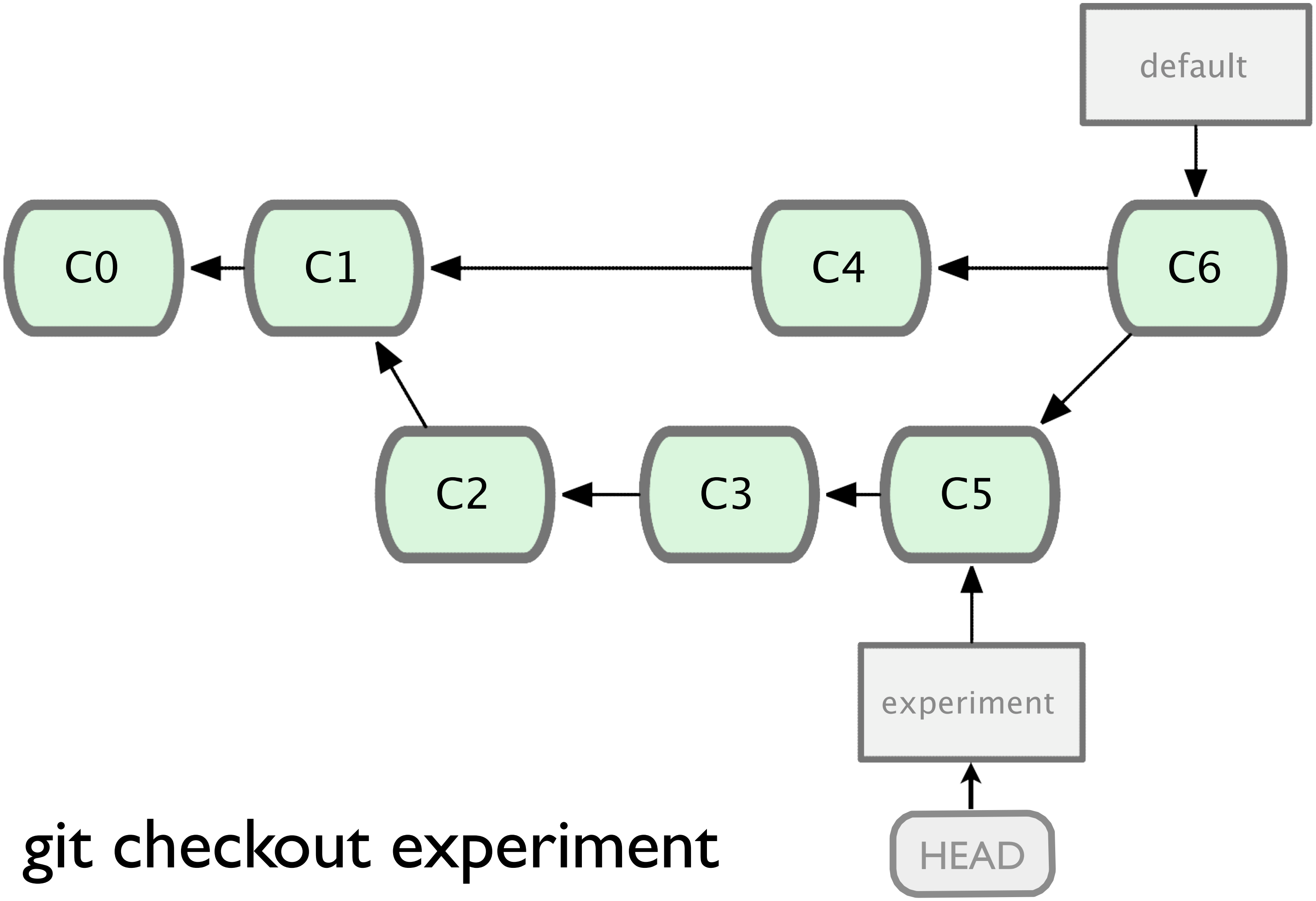


git checkout default

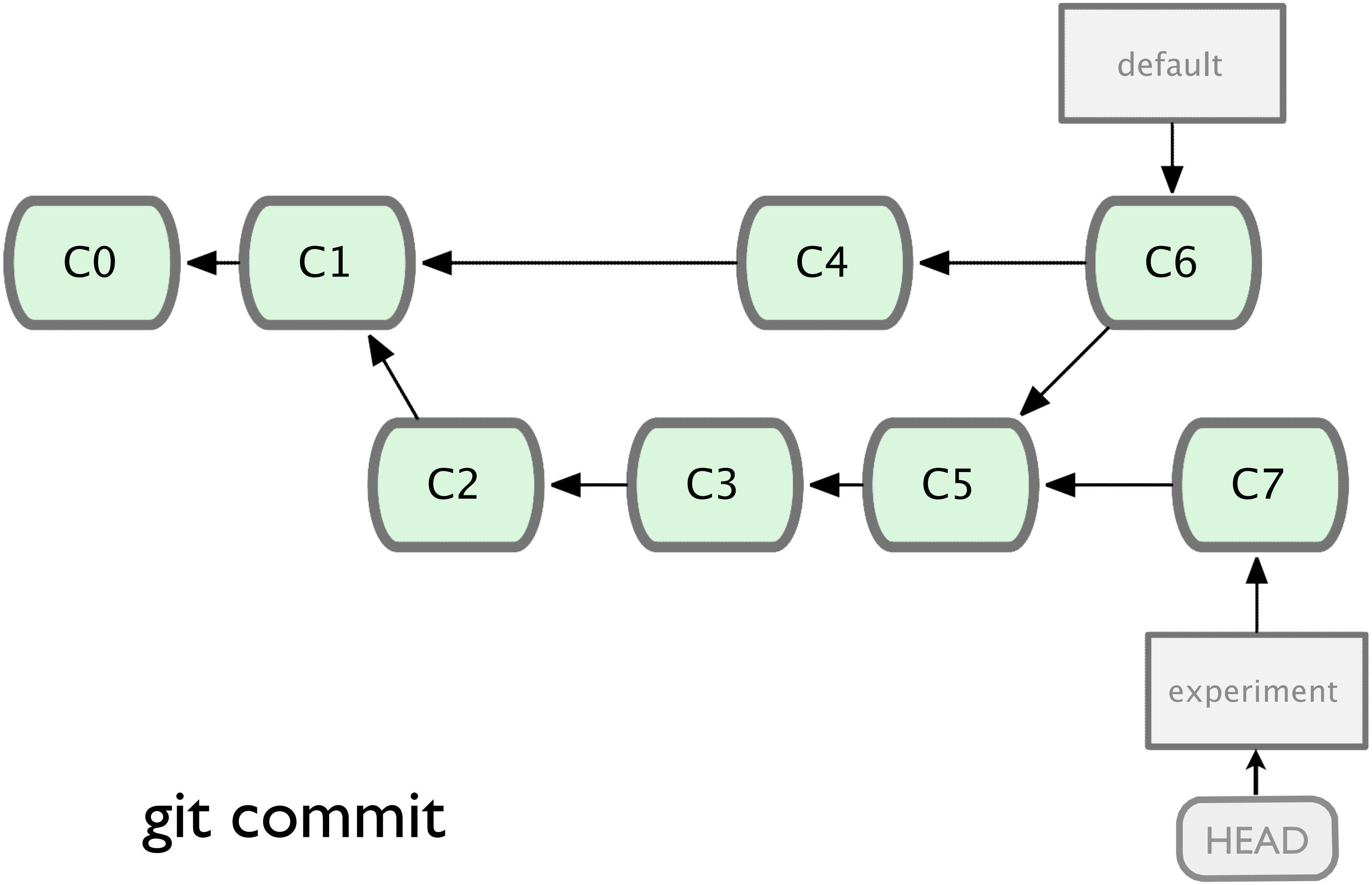


git checkout default

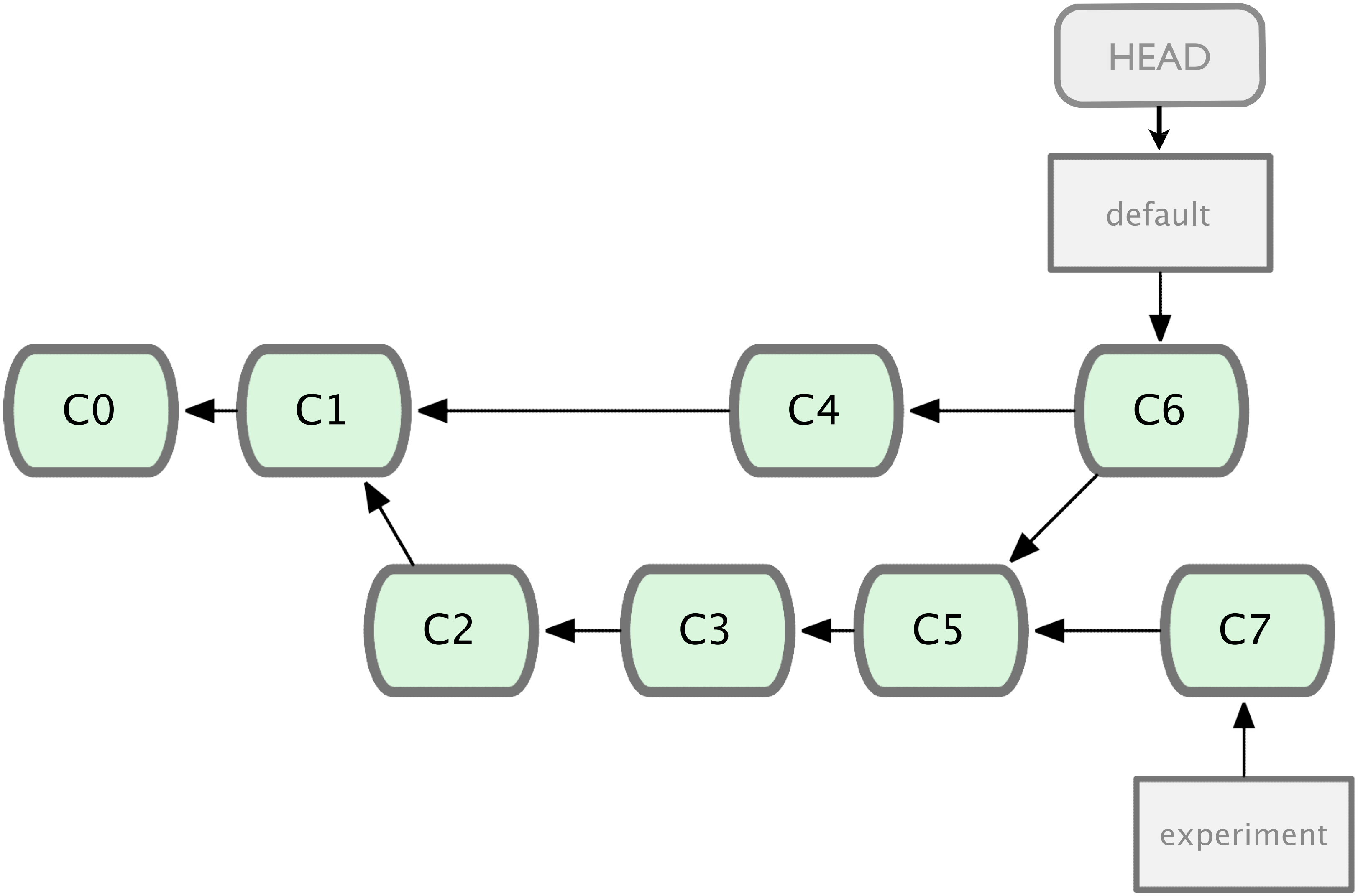
git merge experiment



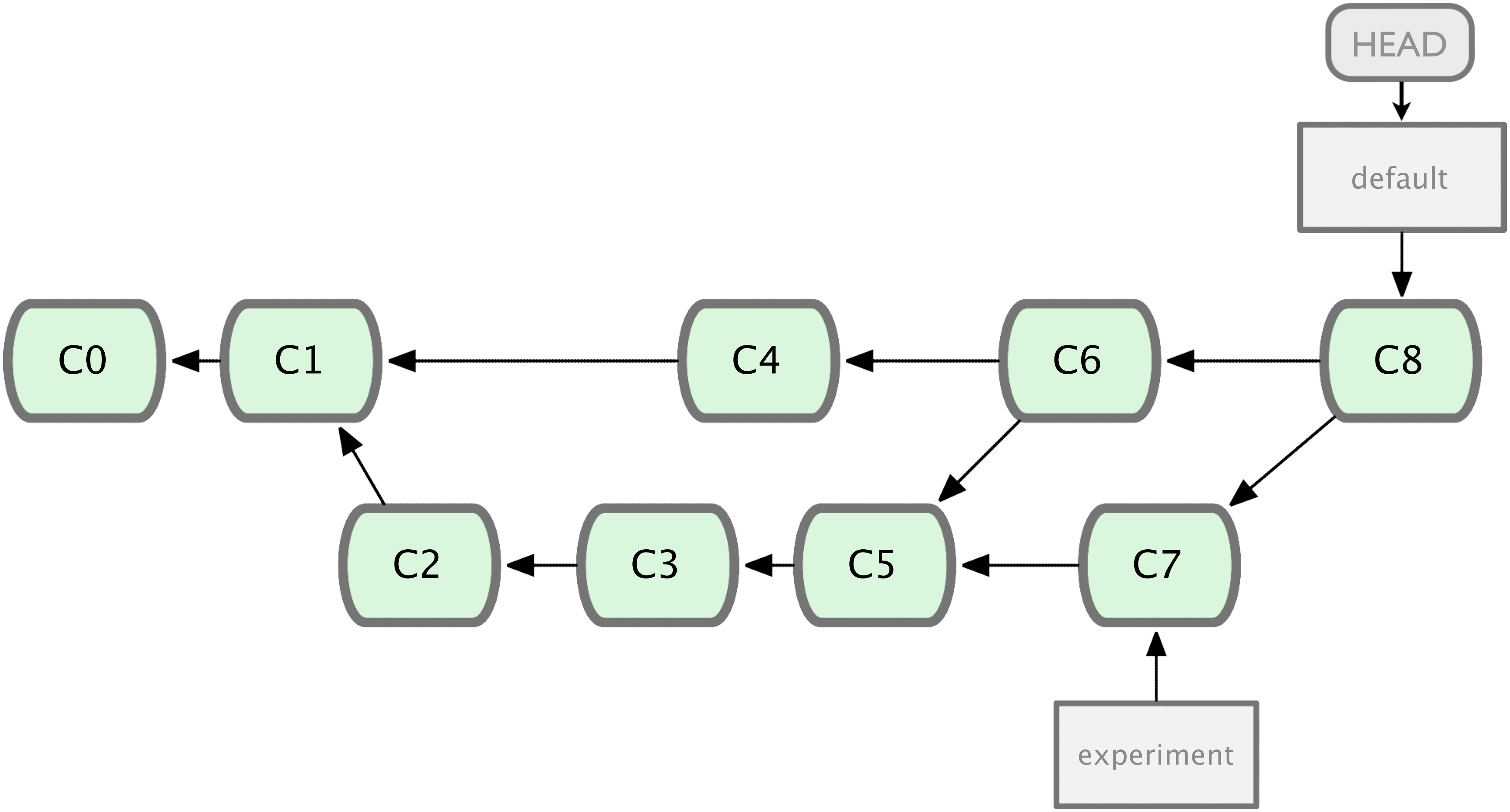
git checkout experiment



git commit



git checkout default



git merge experiment

Lots more to merging

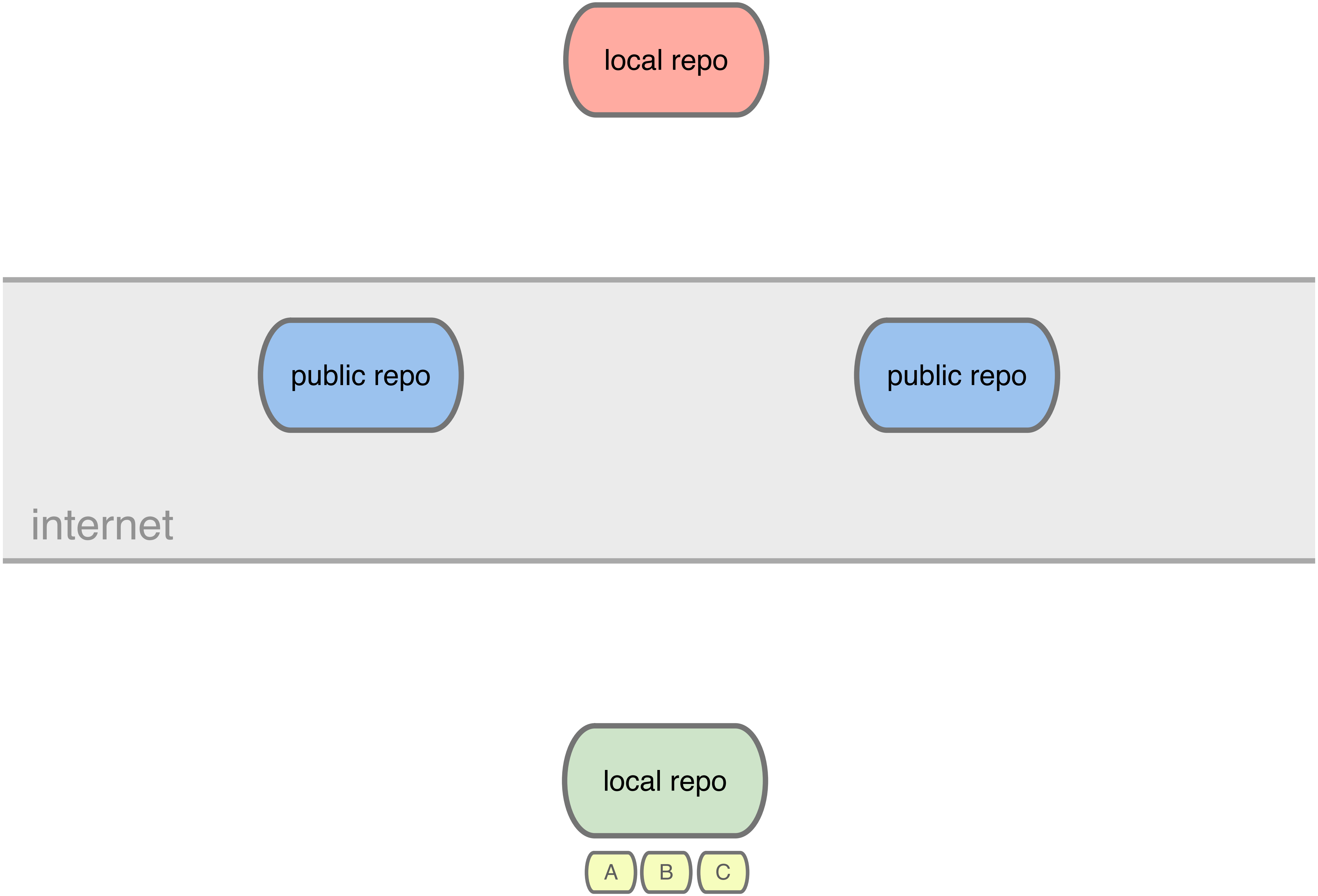
AND…

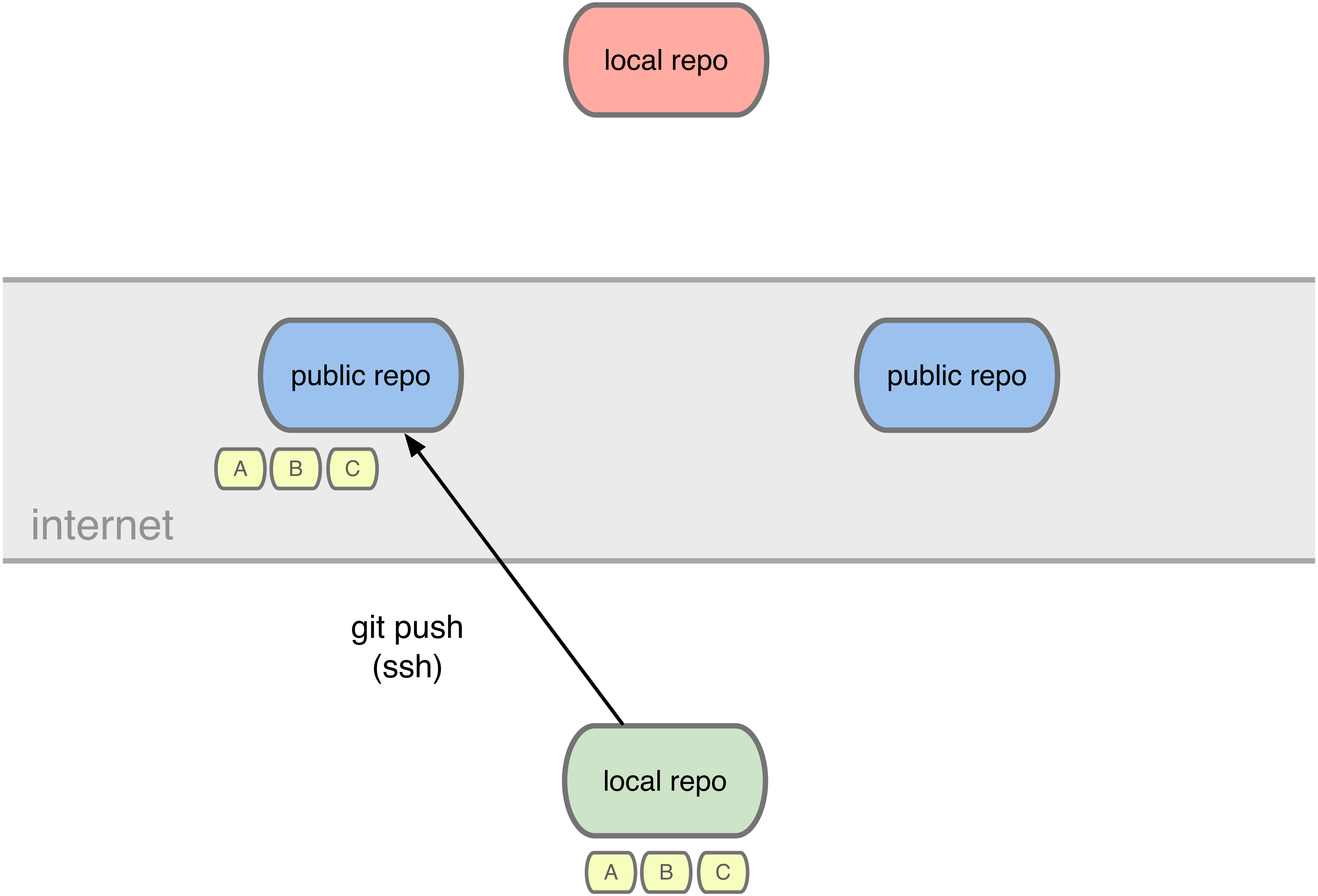
Merging cannot always be done cleanly and may need your interaction to succeed.

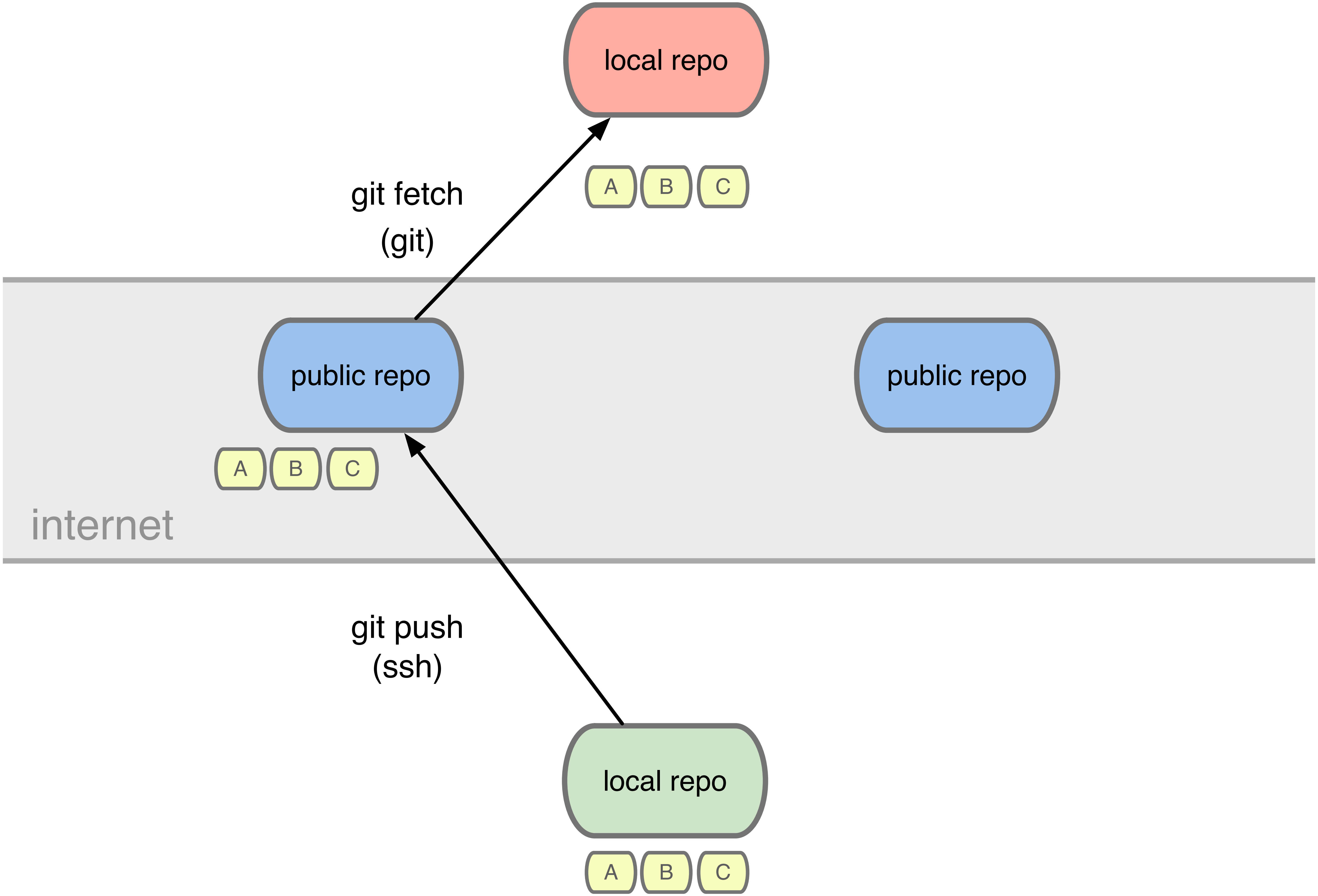
See the git resources for details.

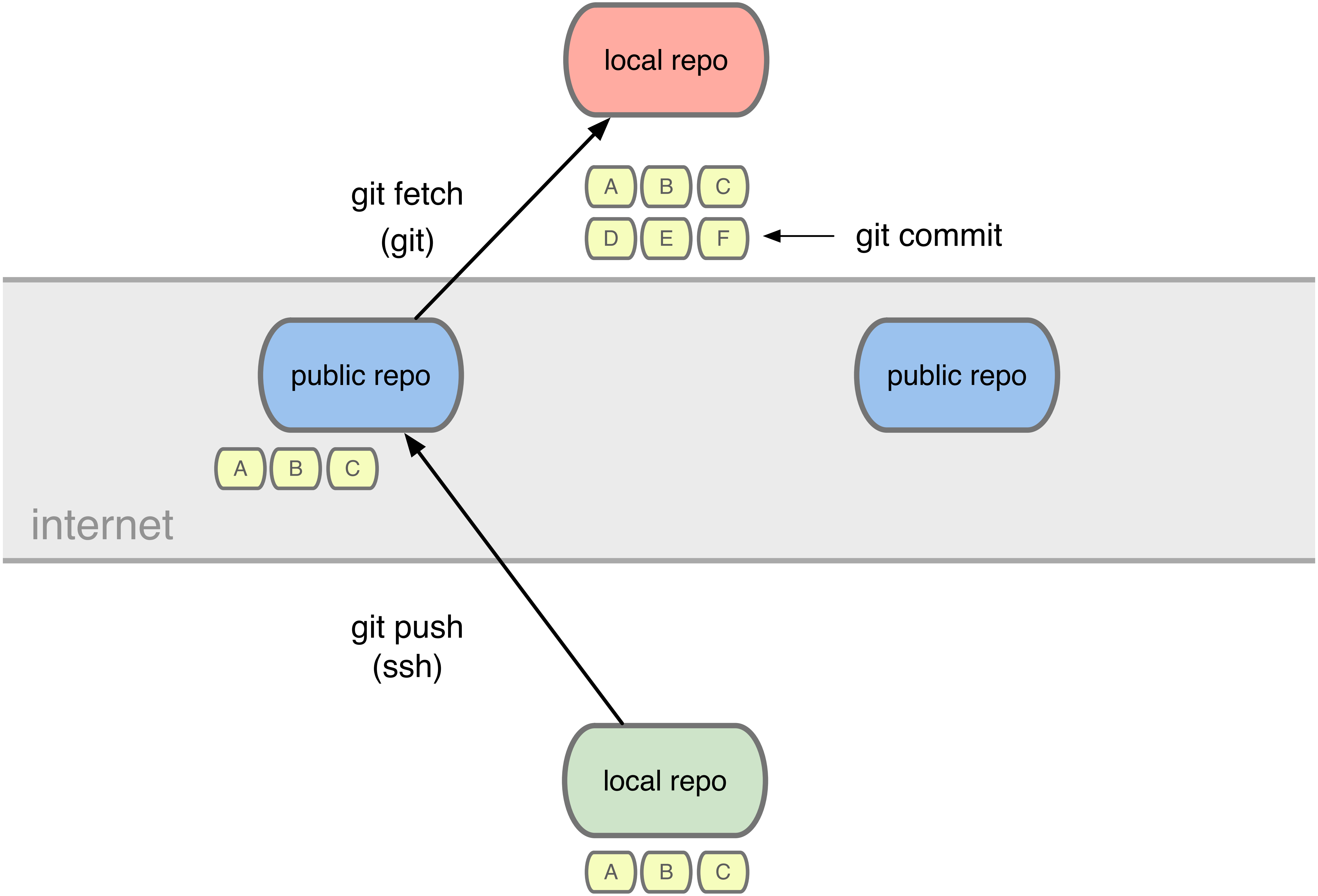
Remotes

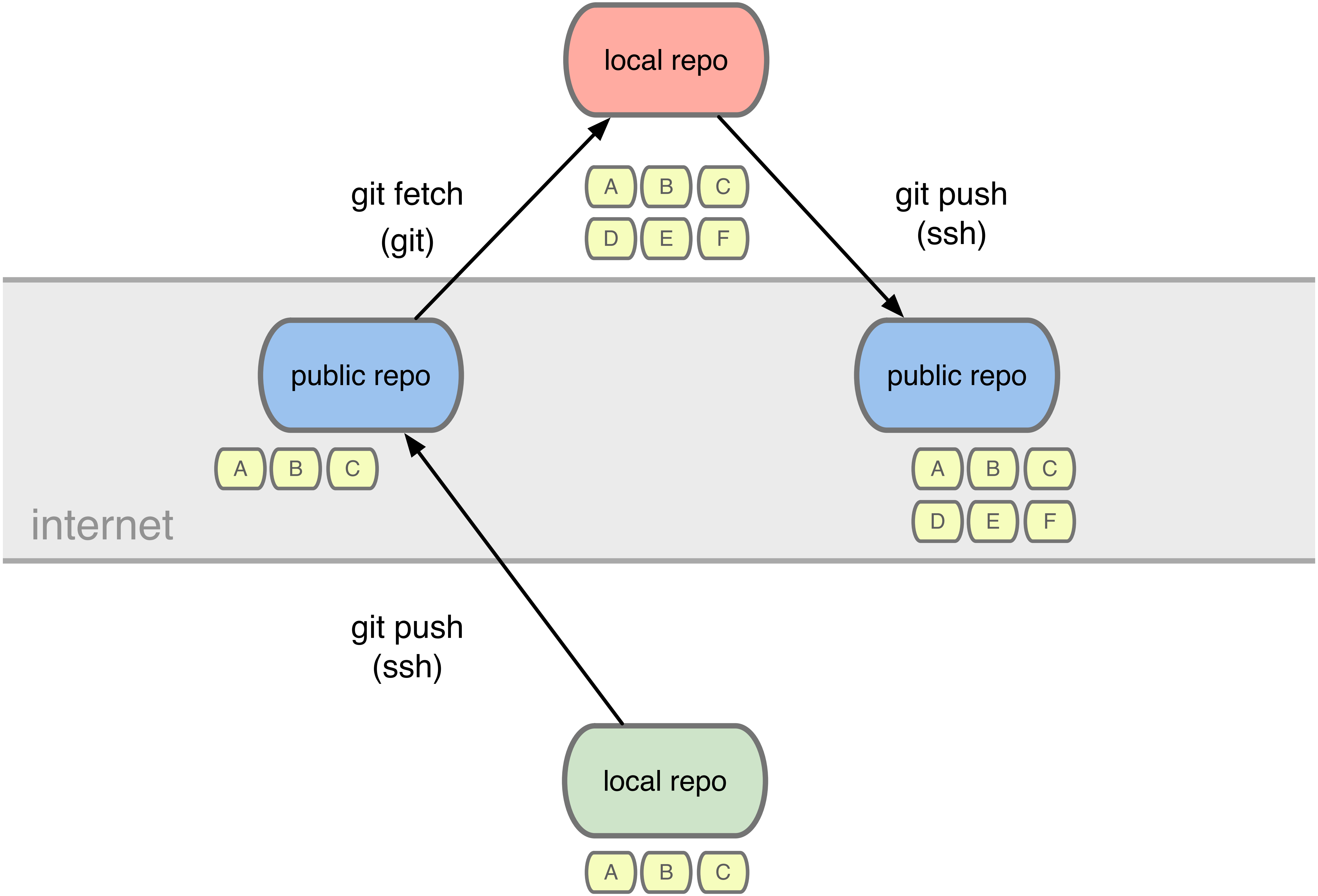
Distributed Workflow

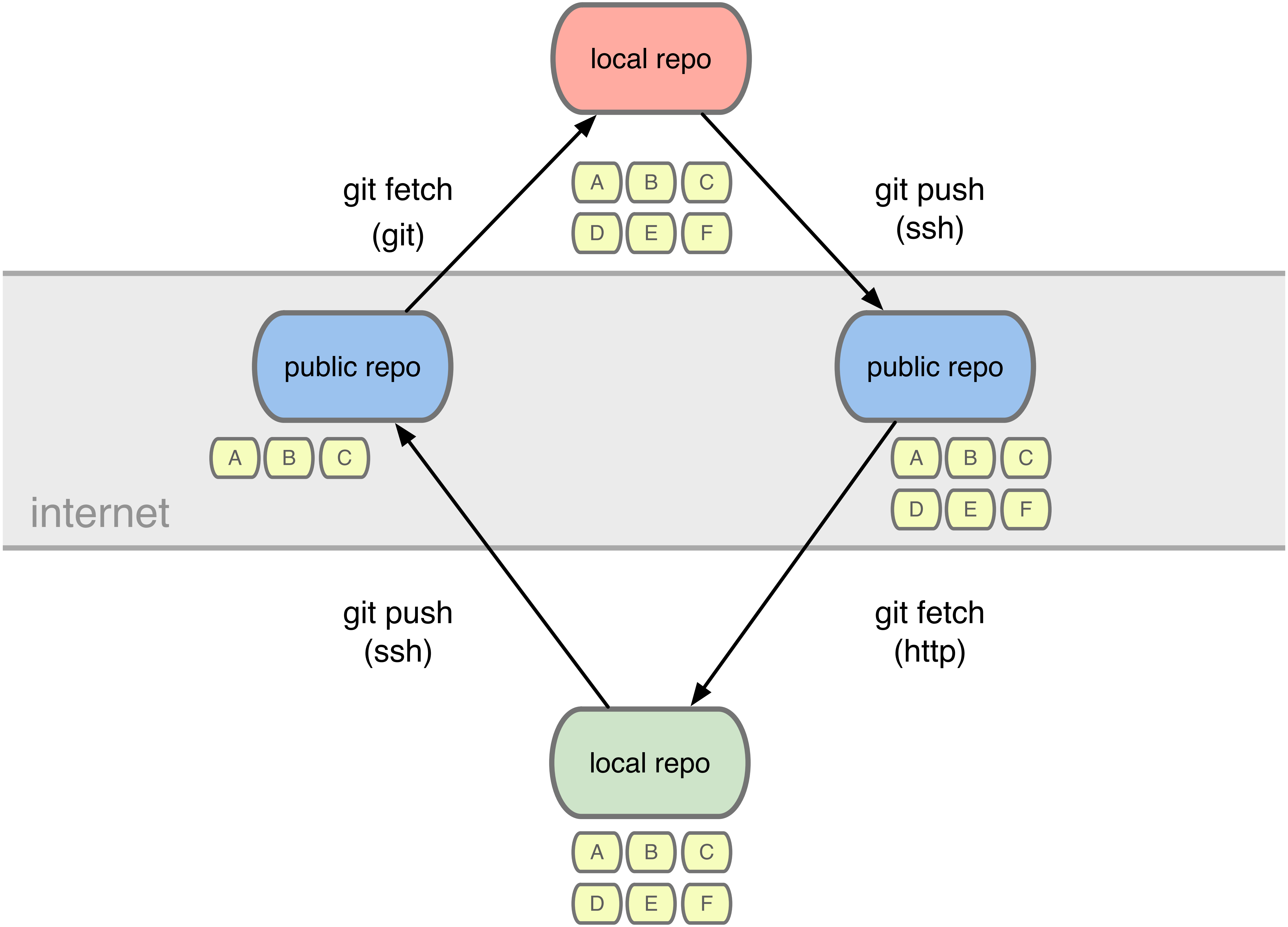

local repo
public repo
public repo
internet
local repo
A
B
C

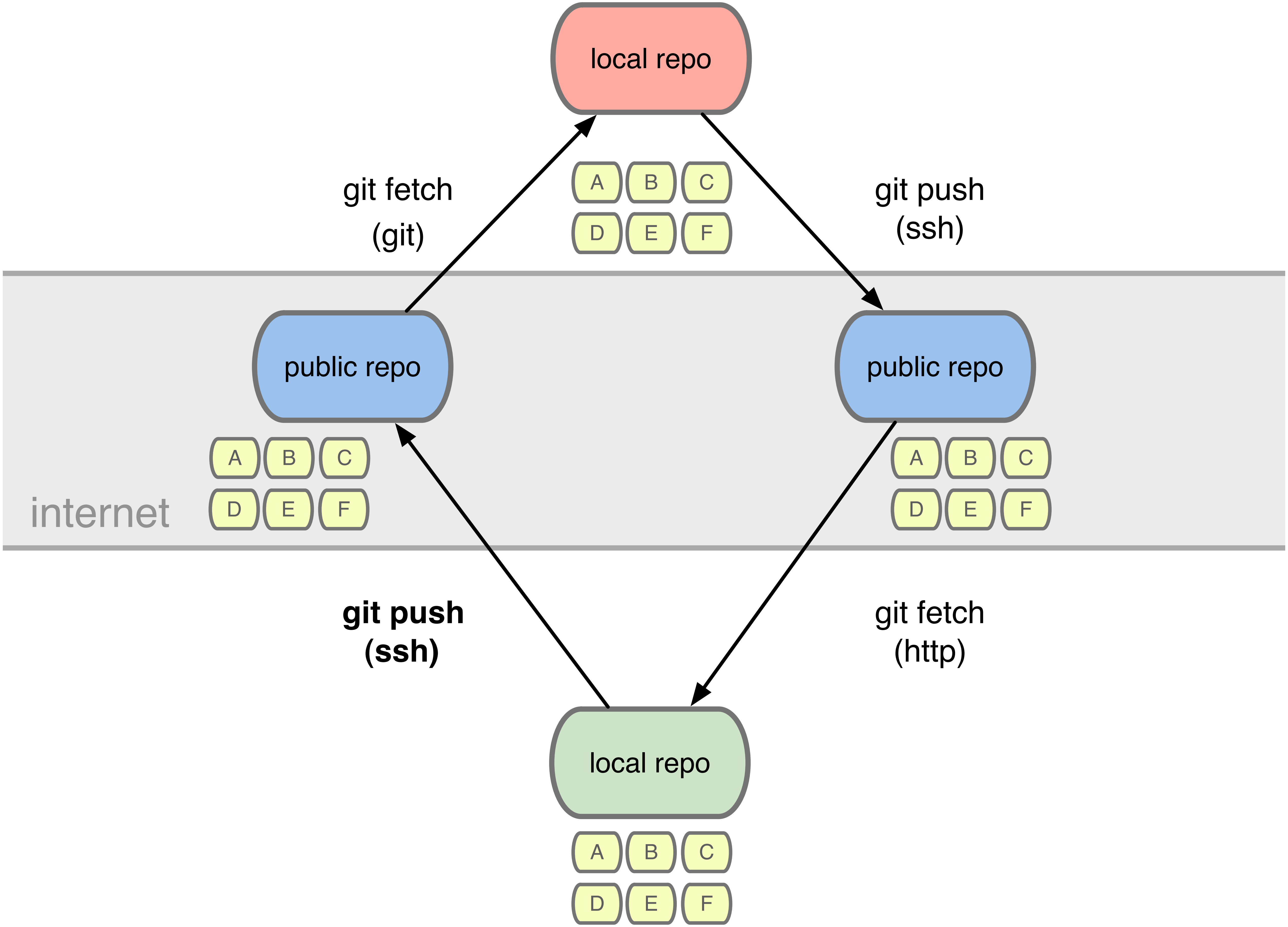

local repo
internet
public repo
public repo
A
B
C
git push
(ssh)
local repo
A
B
C


local repo
A
B
C
git fetch
(git)
public repo
public repo
A
B
C
internet
git push
(ssh)
local repo
A
B
C

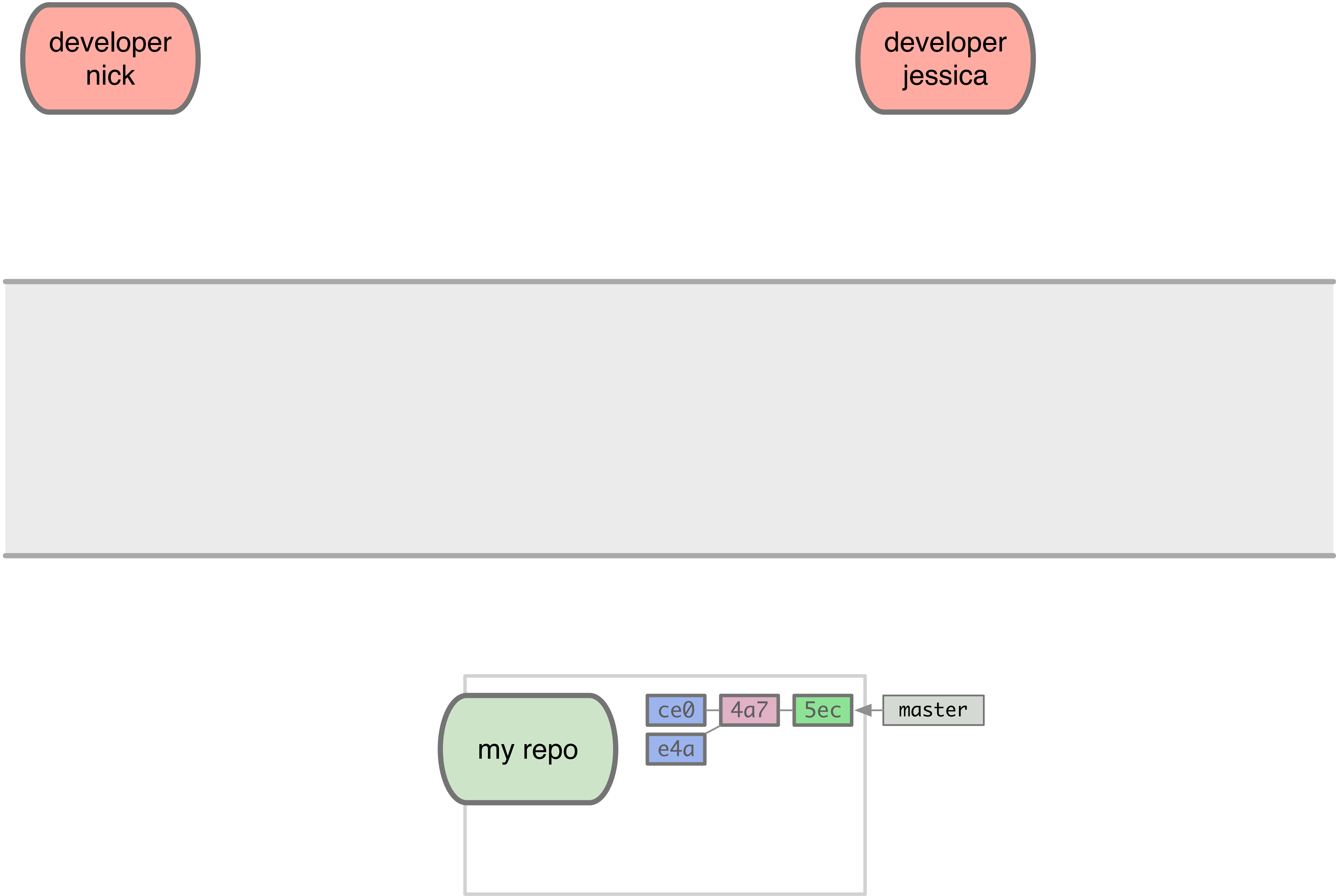

local repo
git fetch
(git)
A B C
D E F
git commit
public repo
public repo
A B C
internet
git push
(ssh)
local repo
A B C

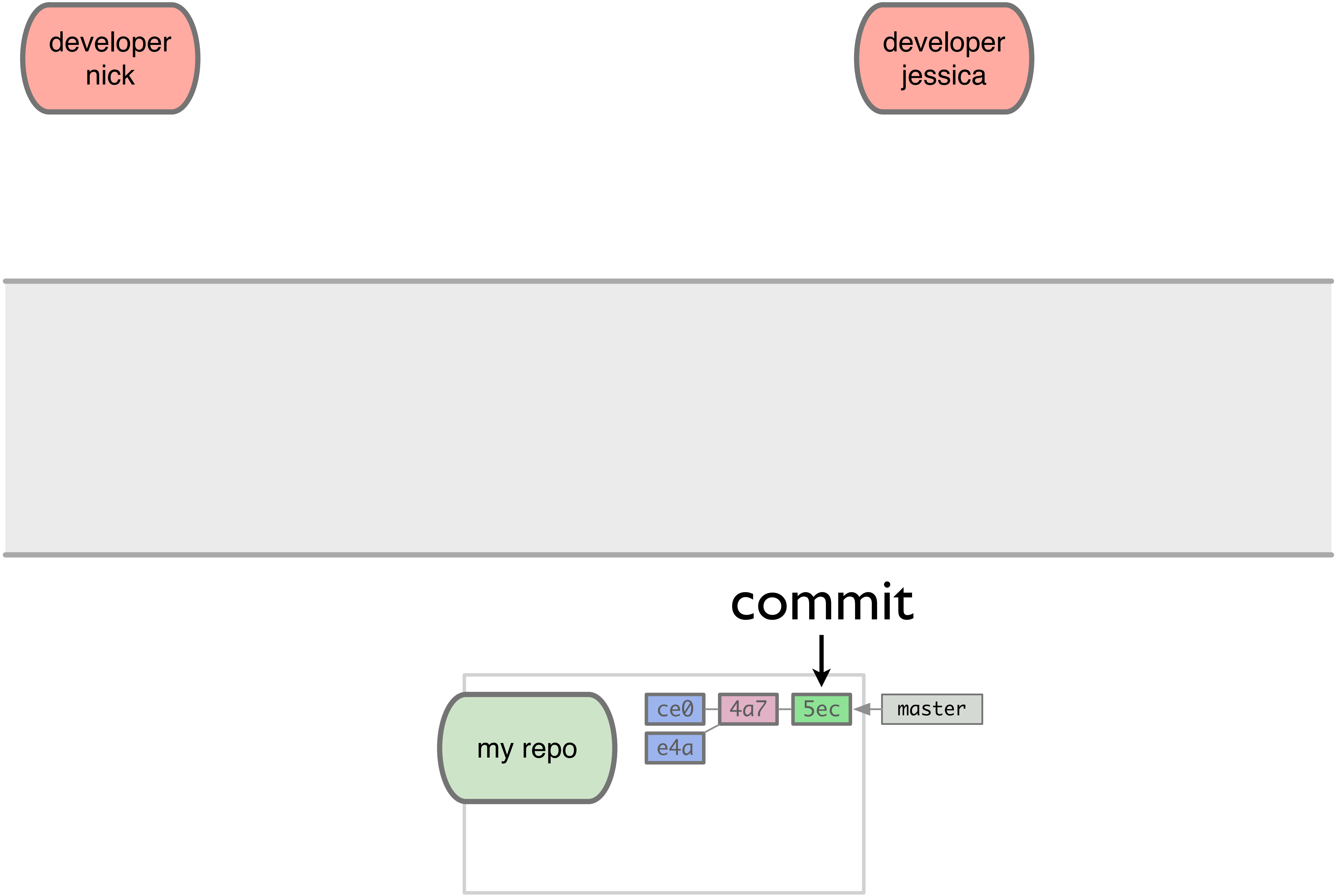

local repo
A B C
D E F
git fetch
(git)
git push
(ssh)
public repo
A B C
public repo
A B C
D E F
internet
git push
(ssh)
local repo
A B C


local repo
git fetch
(git)
A B C
D E F
git push
(ssh)
public repo
public repo
A B C
A B C
D E F
internet
git push
(ssh)
git fetch
(http)
local repo
A B C
D E F


local repo
git fetch
(git)
A B C
D E F
git push
(ssh)
public repo
public repo
A B C
D E F
internet
A B C
D E F
git push
(ssh)
git fetch
(http)
local repo
A B C
D E F

Multiple Remotes


developer nick
developer jessica
my repo
ce0
4a7
5ec
master
e4a



developer nick

developer jessica

commit

my repo

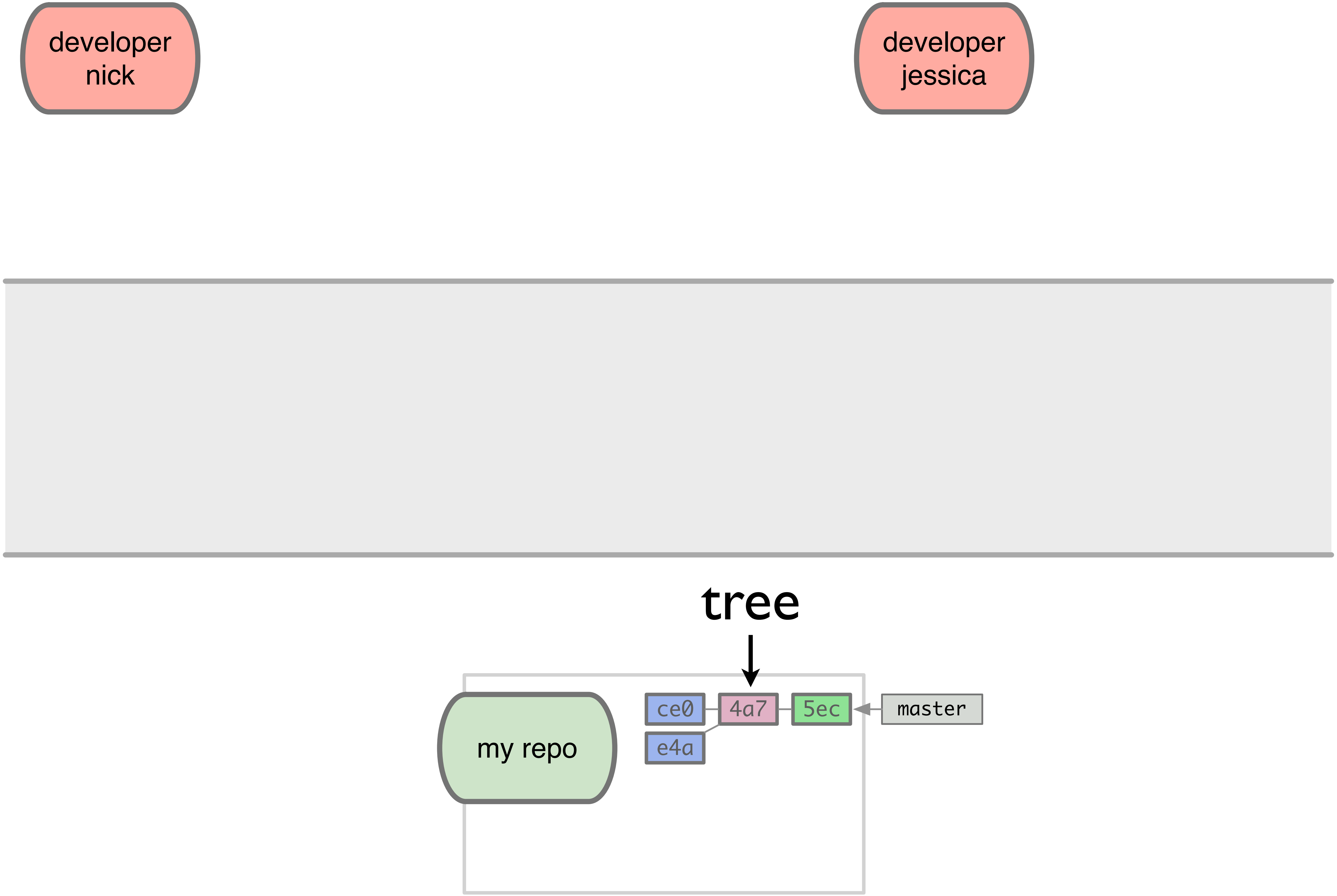
ce0

4a7

5ec

e4a

master



developer nick

developer jessica

tree

my repo

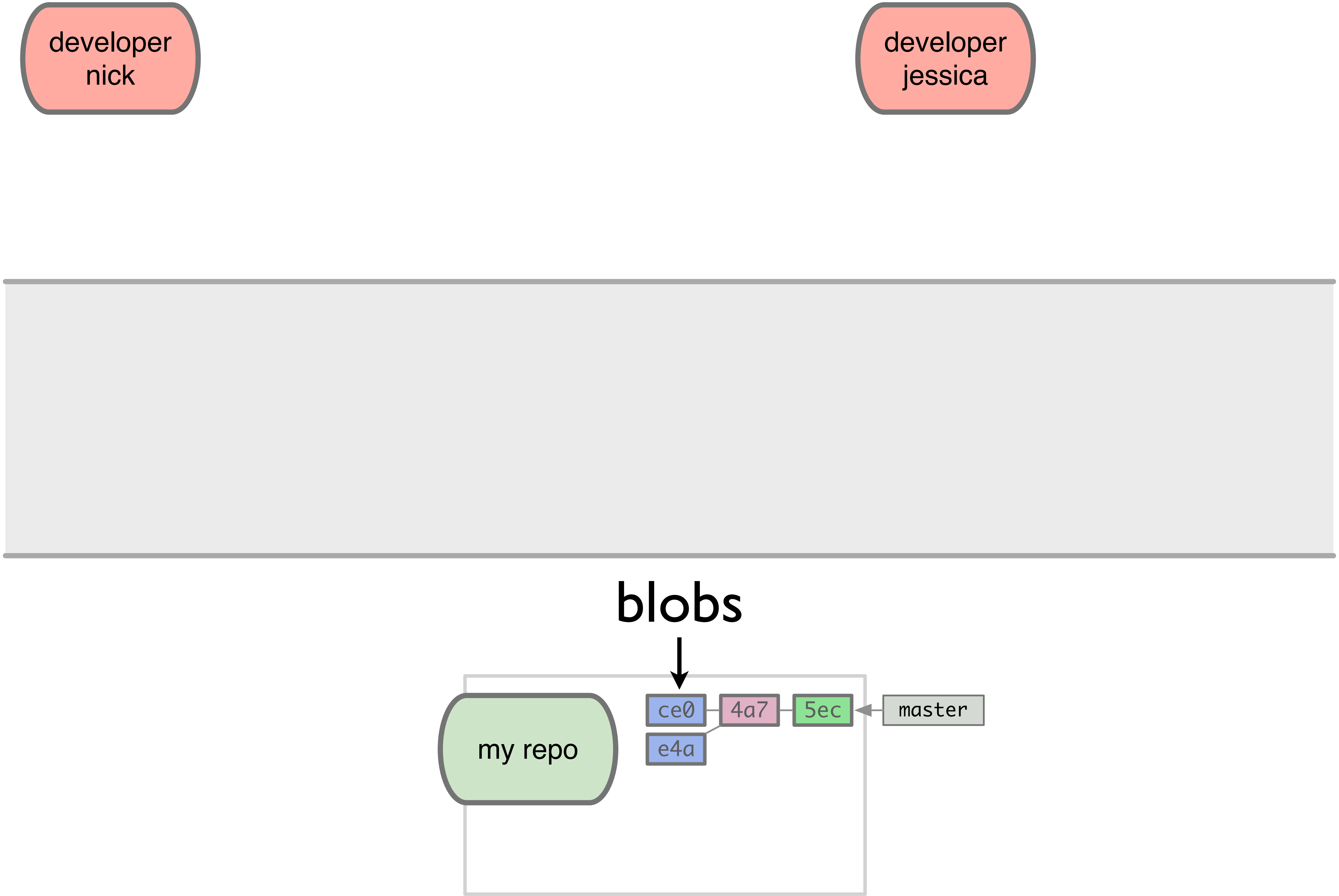
ce0

4a7

5ec

e4a

master



developer nick

developer jessica

blobs

my repo

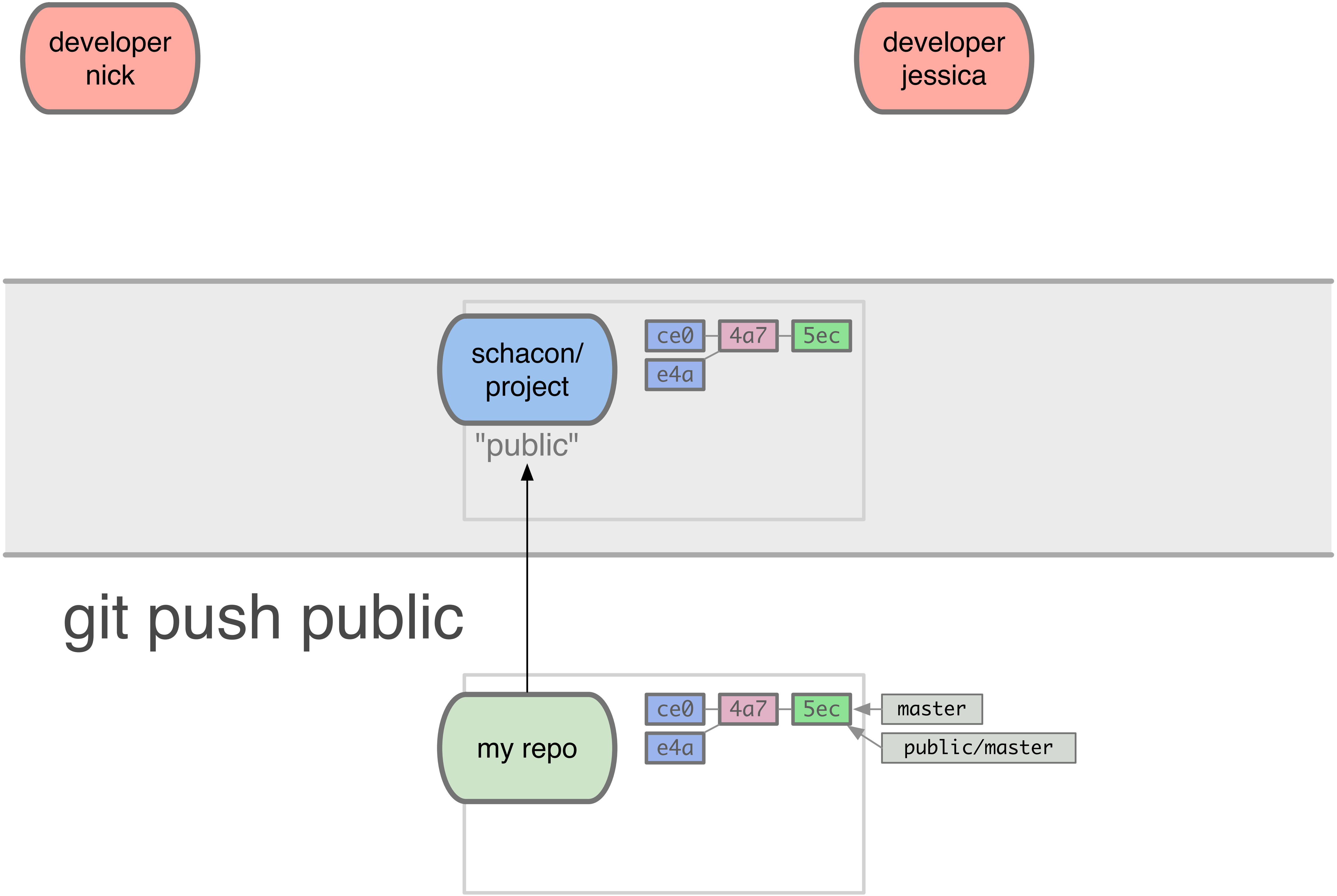
ce0

4a7

5ec

e4a

master



developer nick

developer jessica

schacon/ project

"public"

ce0 4a7 5ec

e4a

git push public

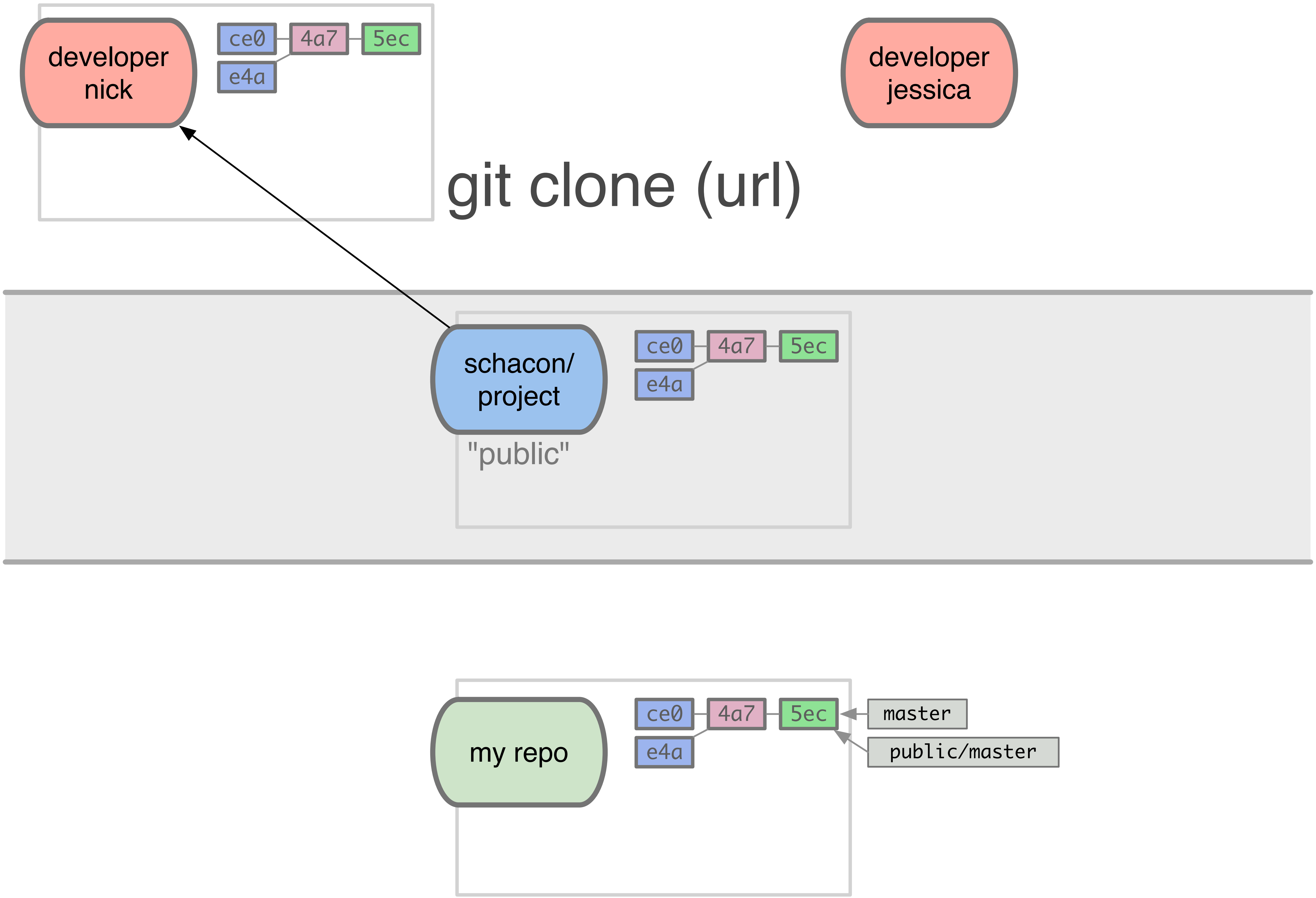
my repo

ce0 4a7 5ec

e4a

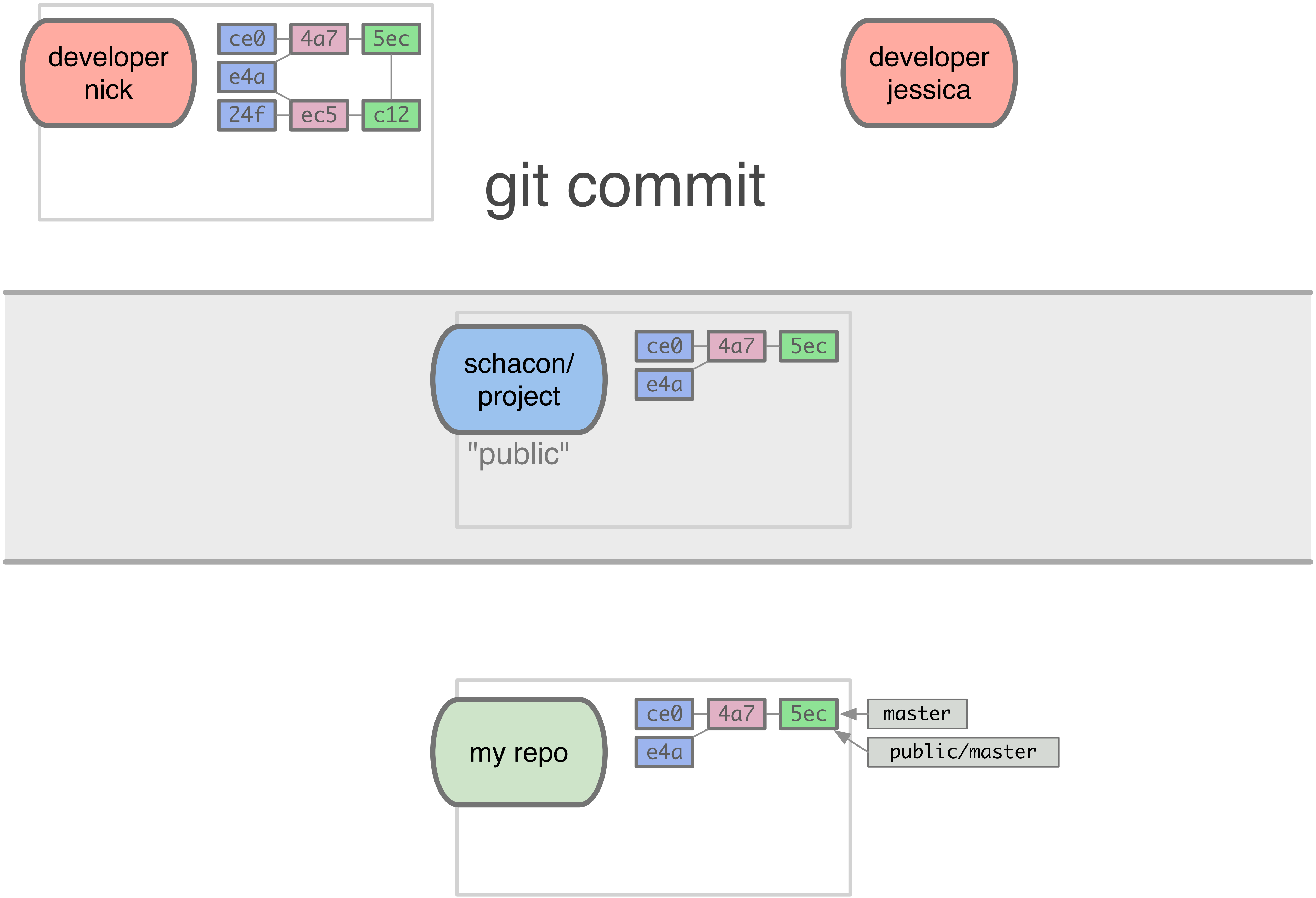
master

public/master

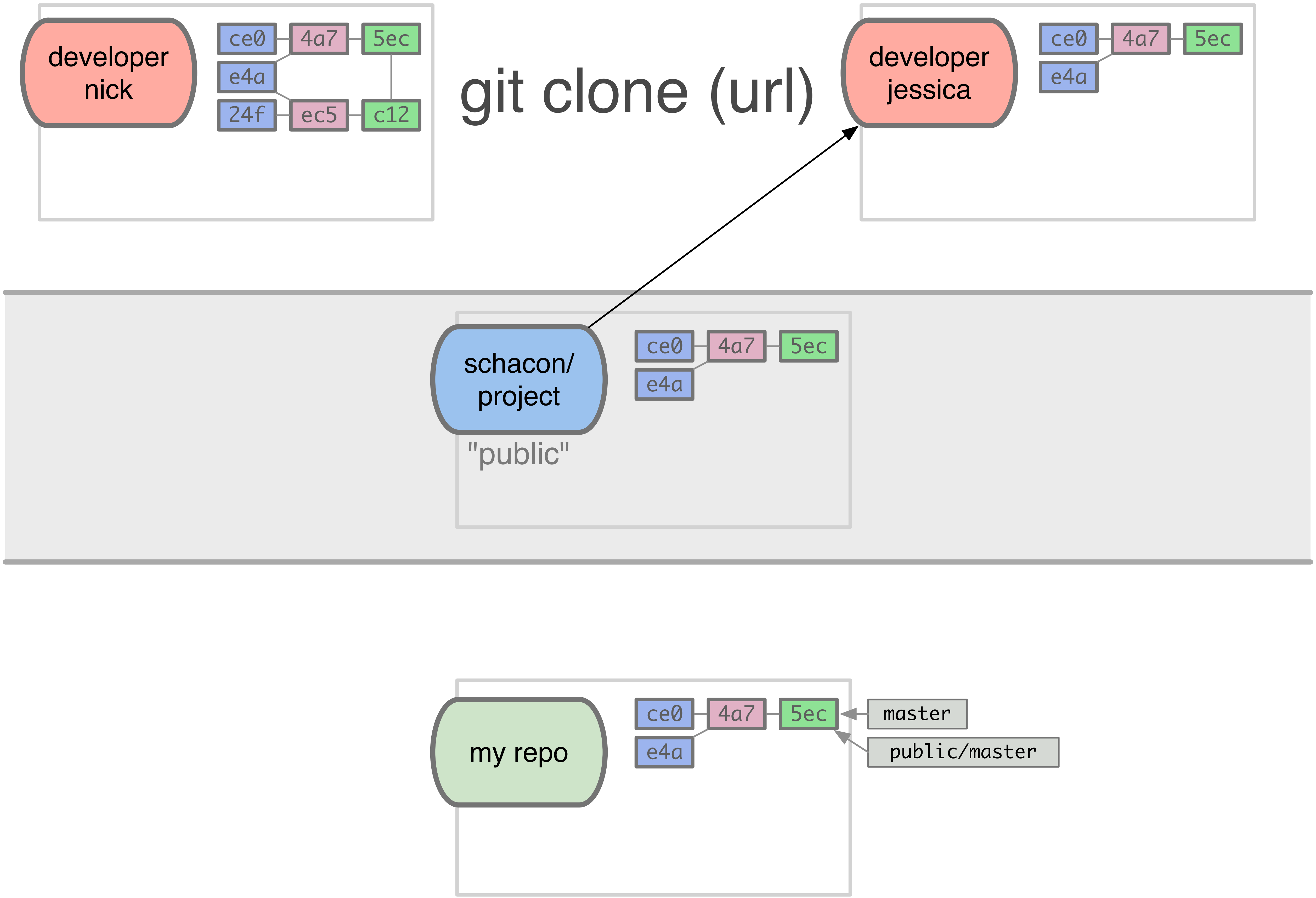


git clone (url)

git commit


developer nick
ce0
4a7
5ec
e4a
24f
ec5
c12
developer jessica
schacon/ project
ce0
4a7
5ec
e4a
"public"
my repo
ce0
4a7
5ec
e4a
master
public/master

git clone (url)



developer nick

ce0 4a7 5ec

e4a

24f ec5 c12

developer jessica

ce0 4a7 5ec

e4a

schacon/ project

"public"

ce0 4a7 5ec

e4a

my repo

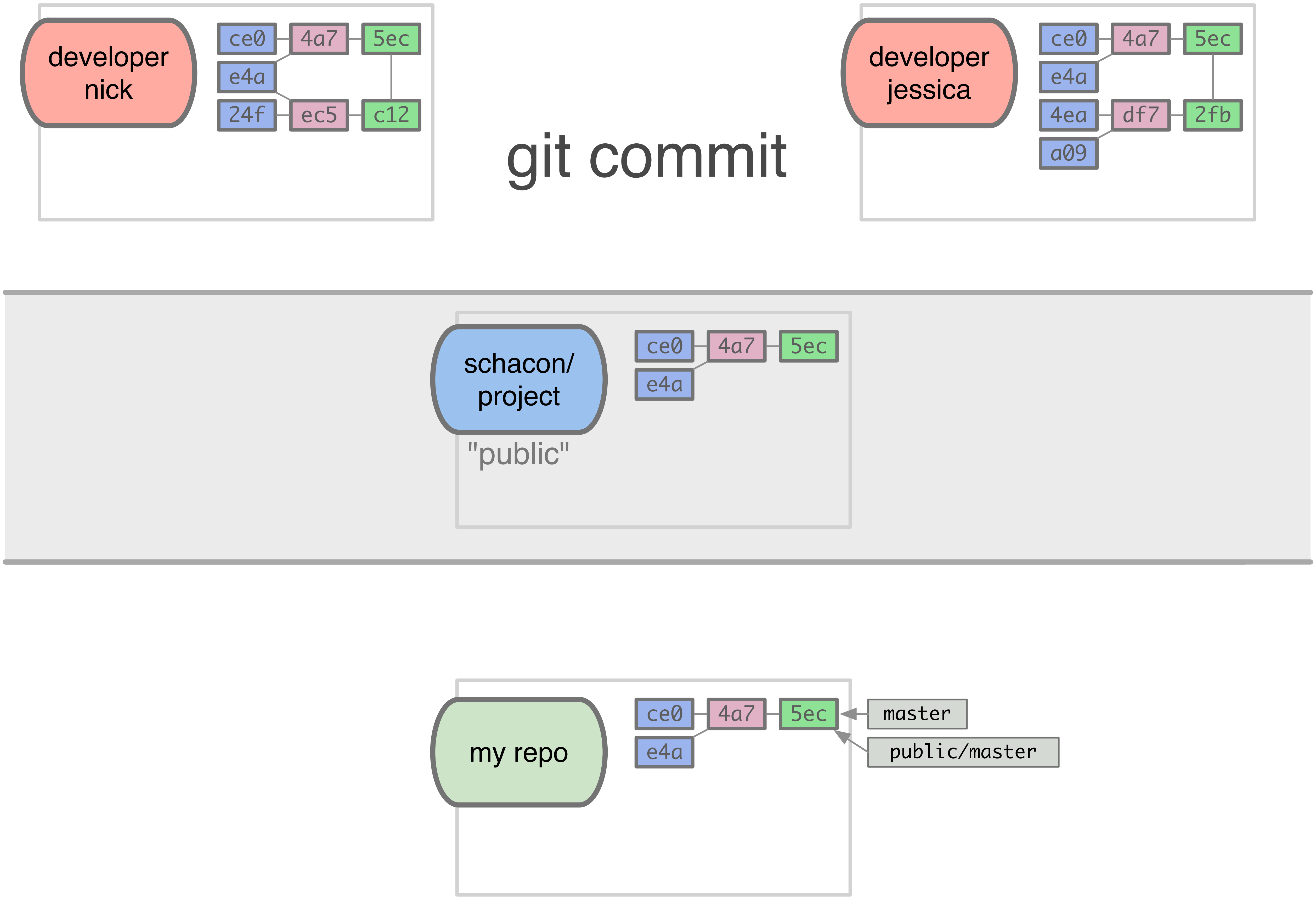
ce0 4a7 5ec

e4a

master

public/master

git commit



developer nick

ce0 4a7 5ec

e4a

24f ec5 c12

developer jessica

ce0 4a7 5ec

e4a

4ea df7 2fb

a09

schacon/ project

"public"

ce0 4a7 5ec

e4a

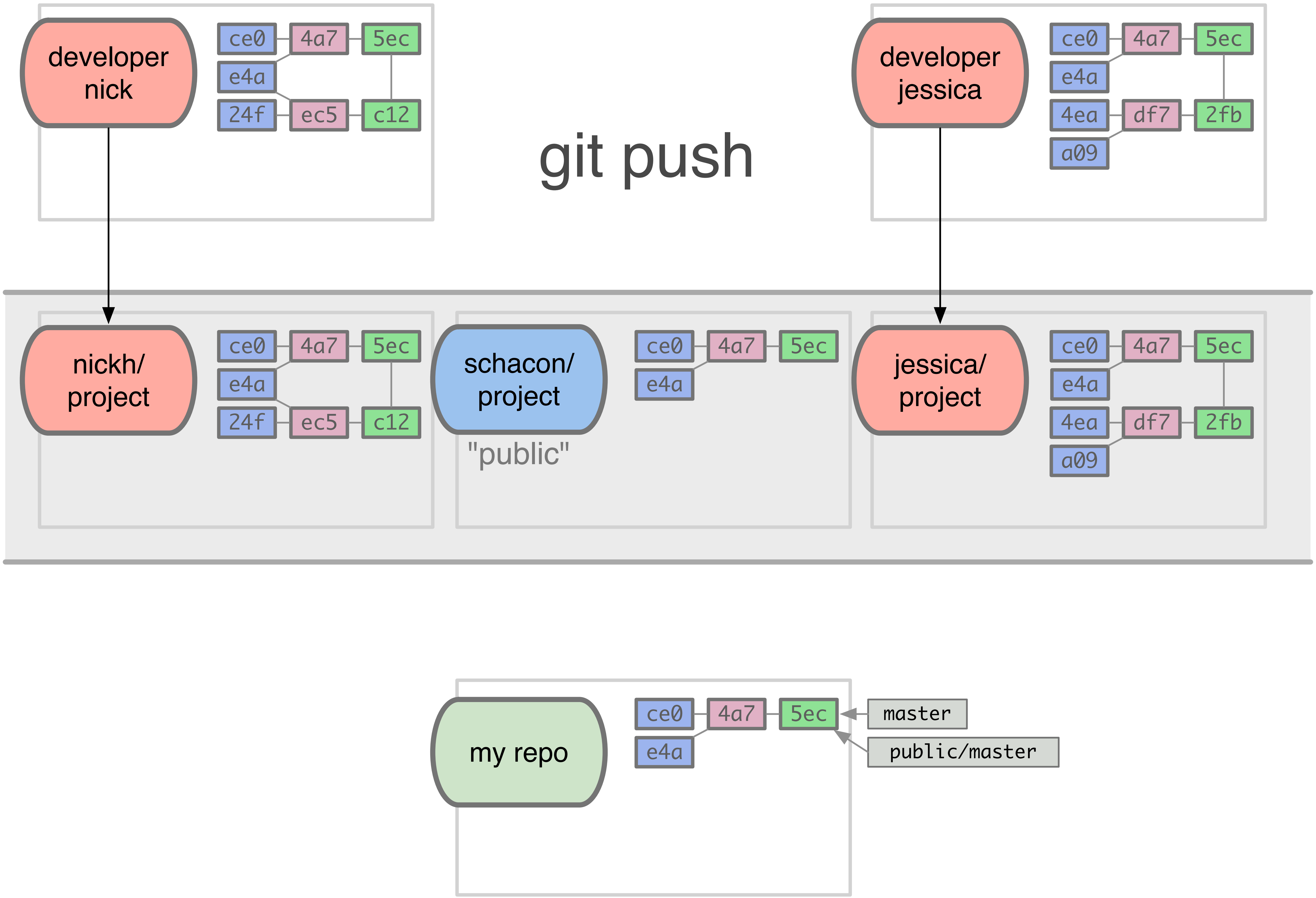
my repo

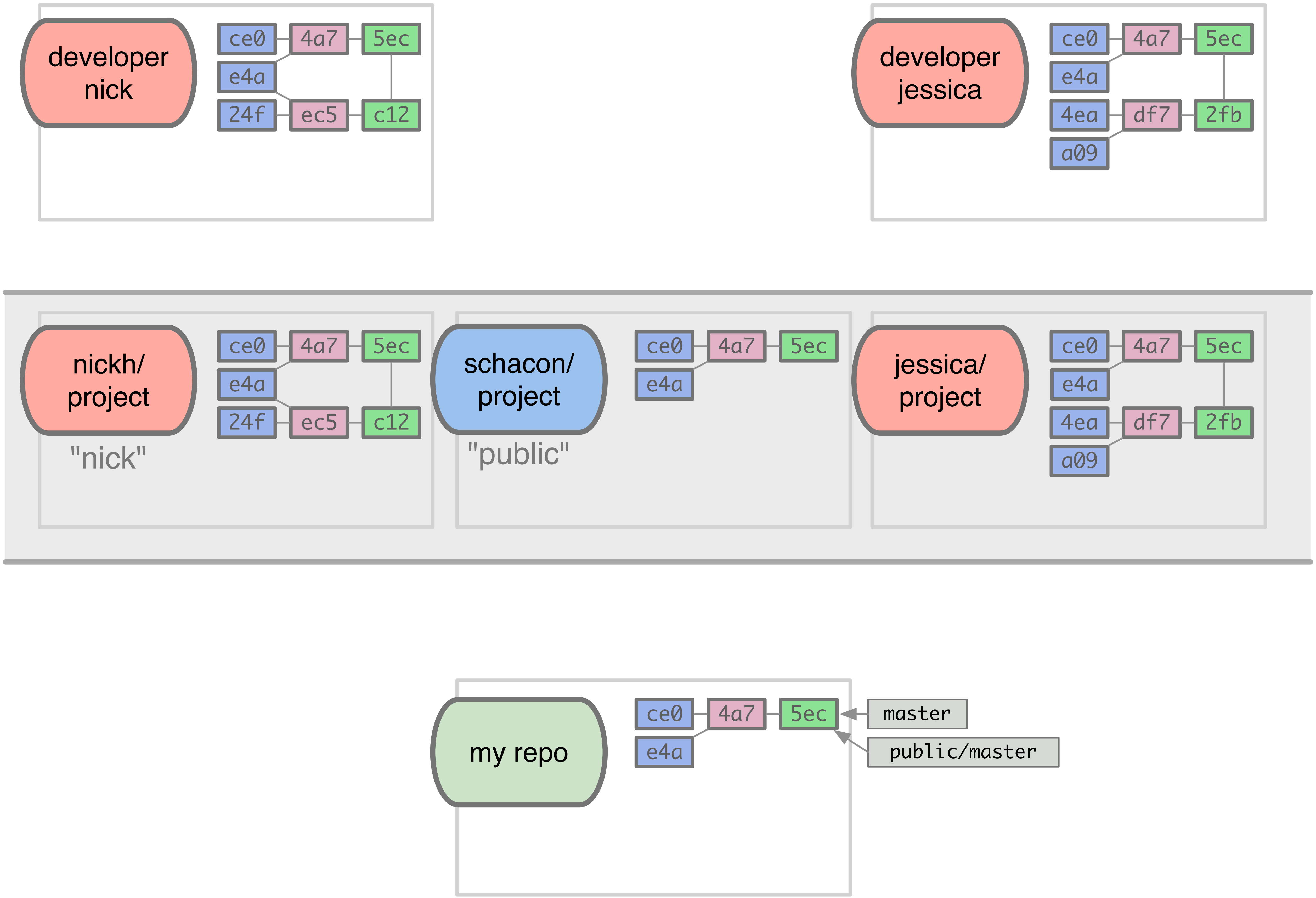
ce0 4a7 5ec

e4a

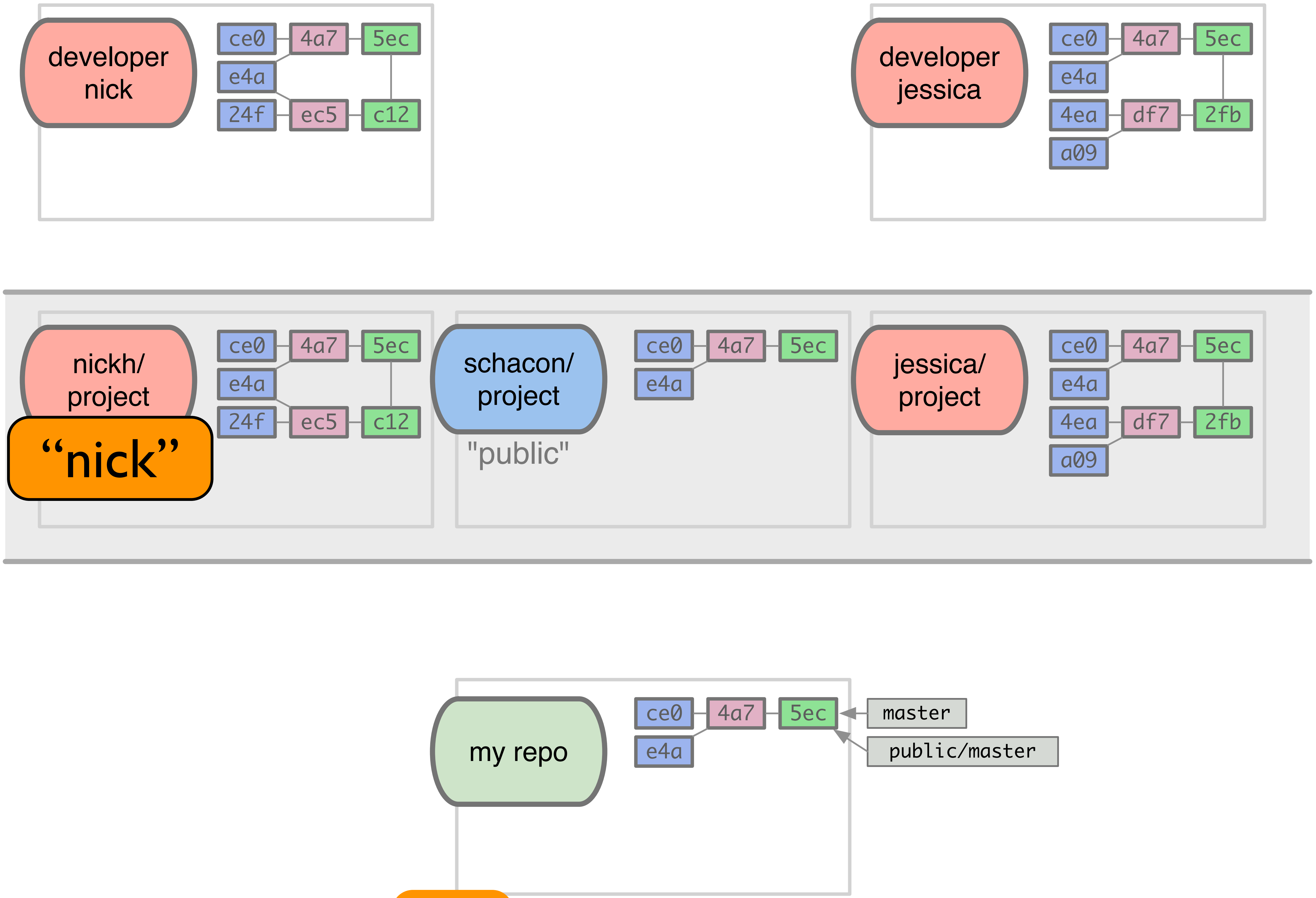
master

public/master

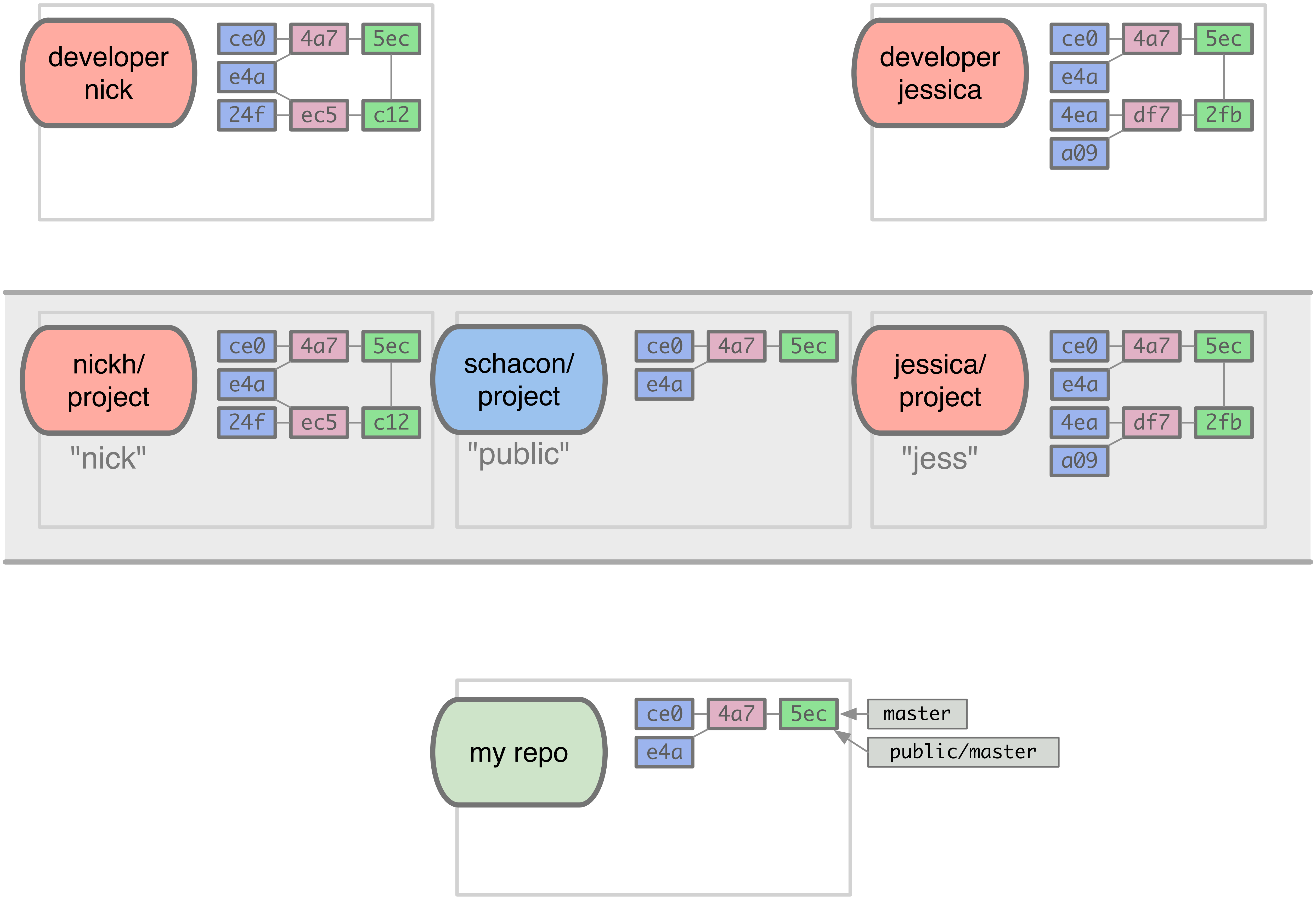

git push
developer nick
ce0
4a7
5ec
e4a
24f
ec5
c12
developer jessica
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
nickh/ project
ce0
4a7
5ec
e4a
24f
ec5
c12
schacon/ project
"public"
ce0
4a7
5ec
e4a
jessica/ project
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
my repo
ce0
4a7
5ec
e4a
master
public/master



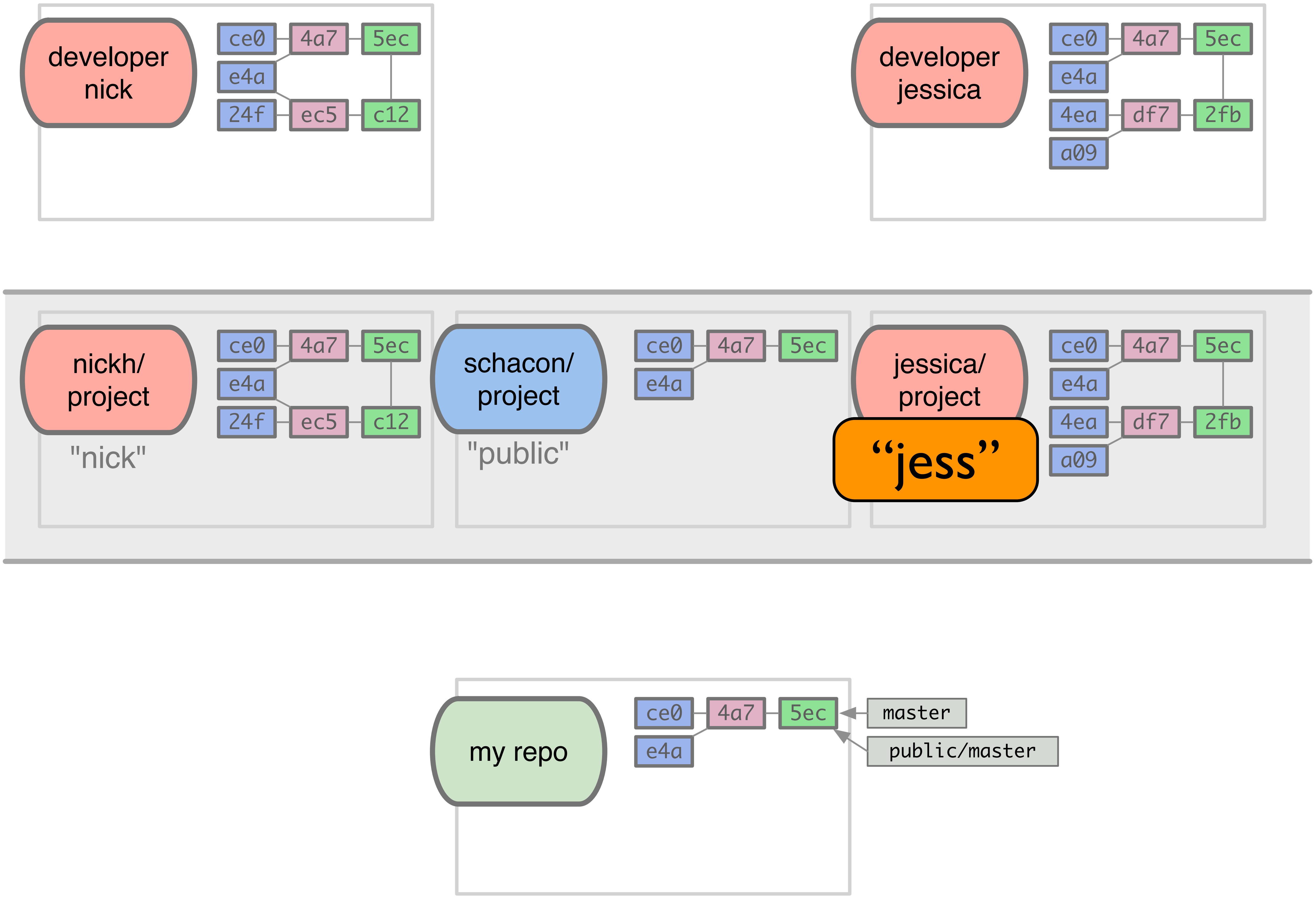
```
git remote add nick git://github.com/nickh/project.git
```



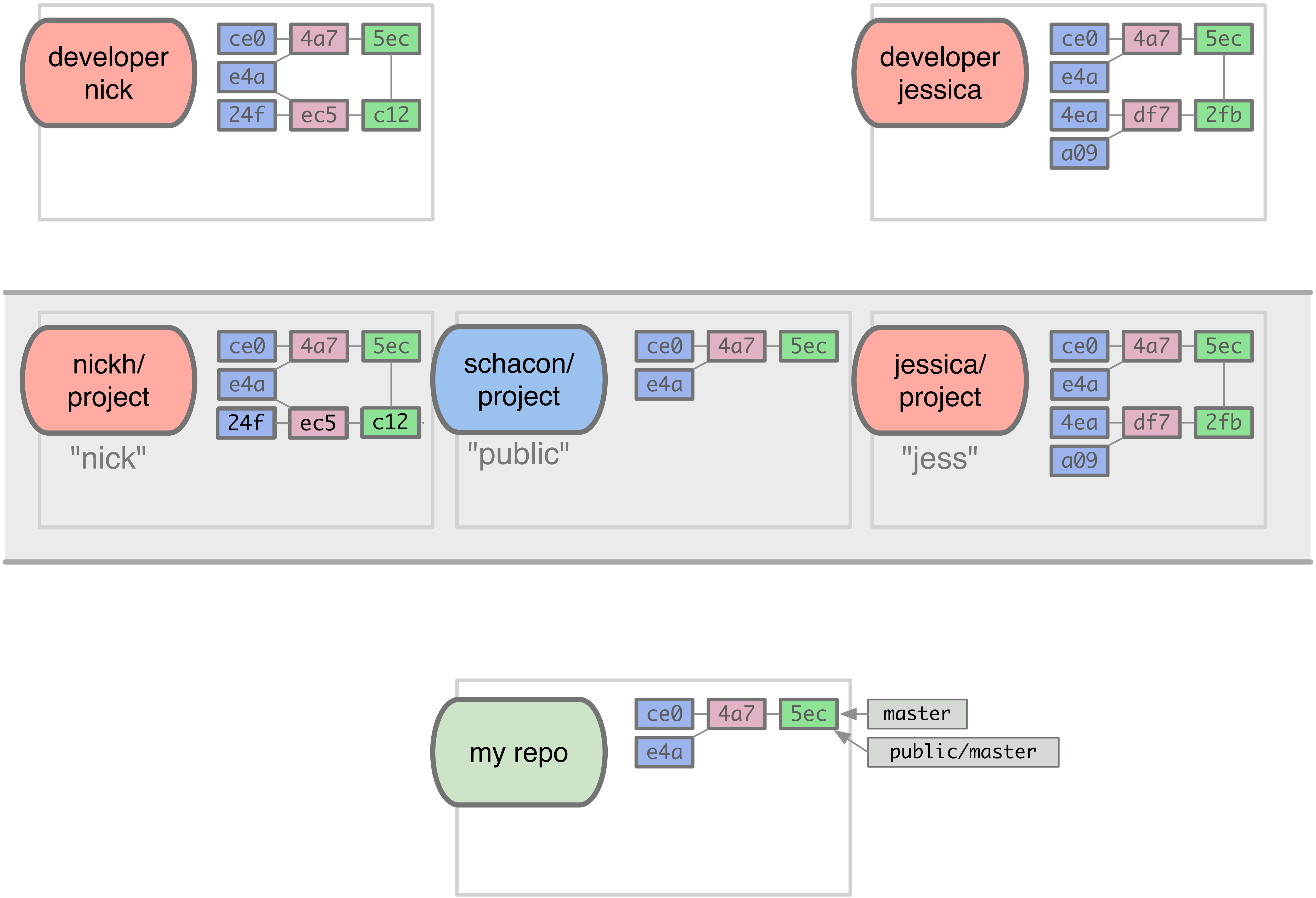
```
git remote add nick git://github.com/nickh/project.git
```



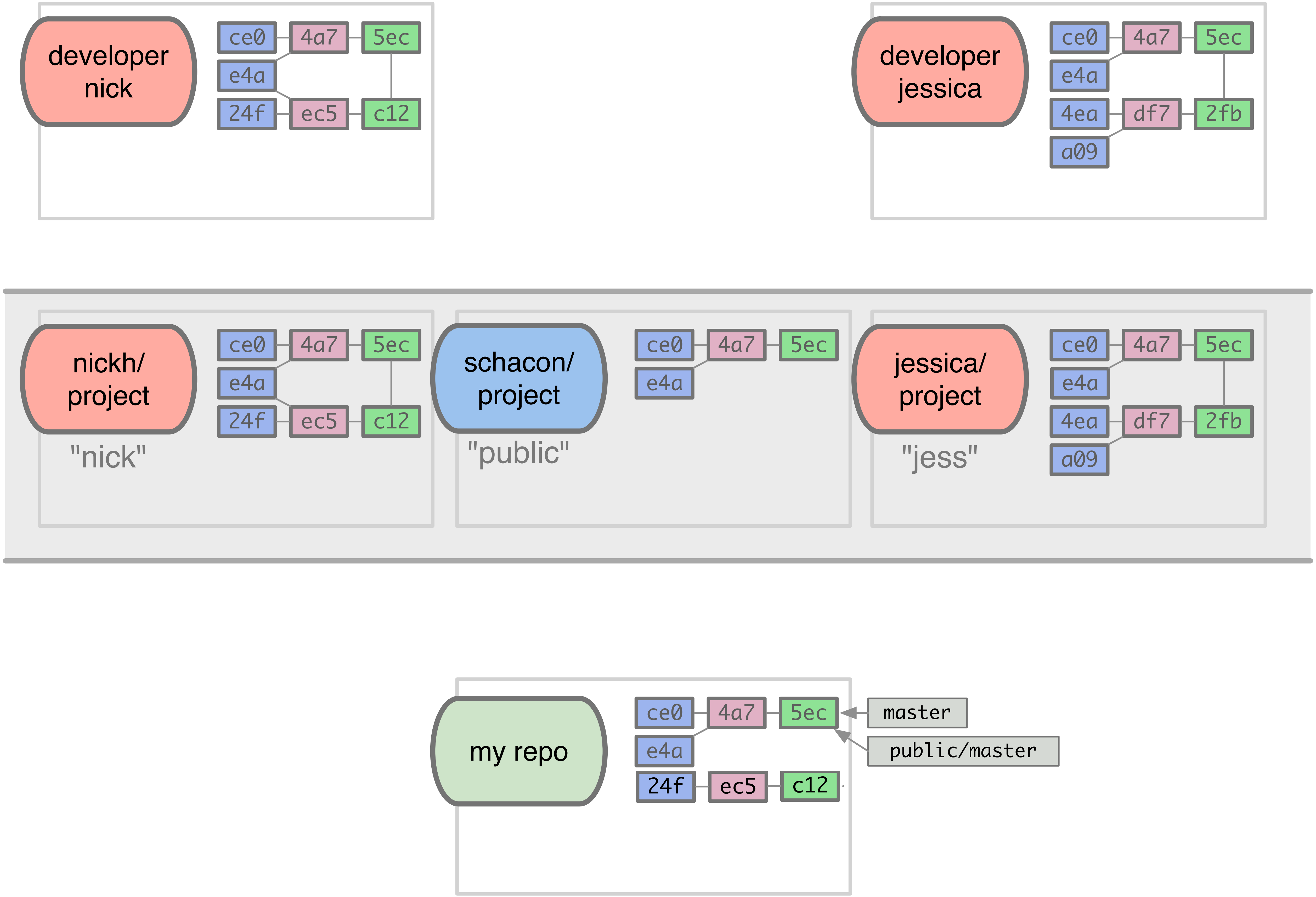
```
git remote add jess git://github.com/jessica/project.git
```



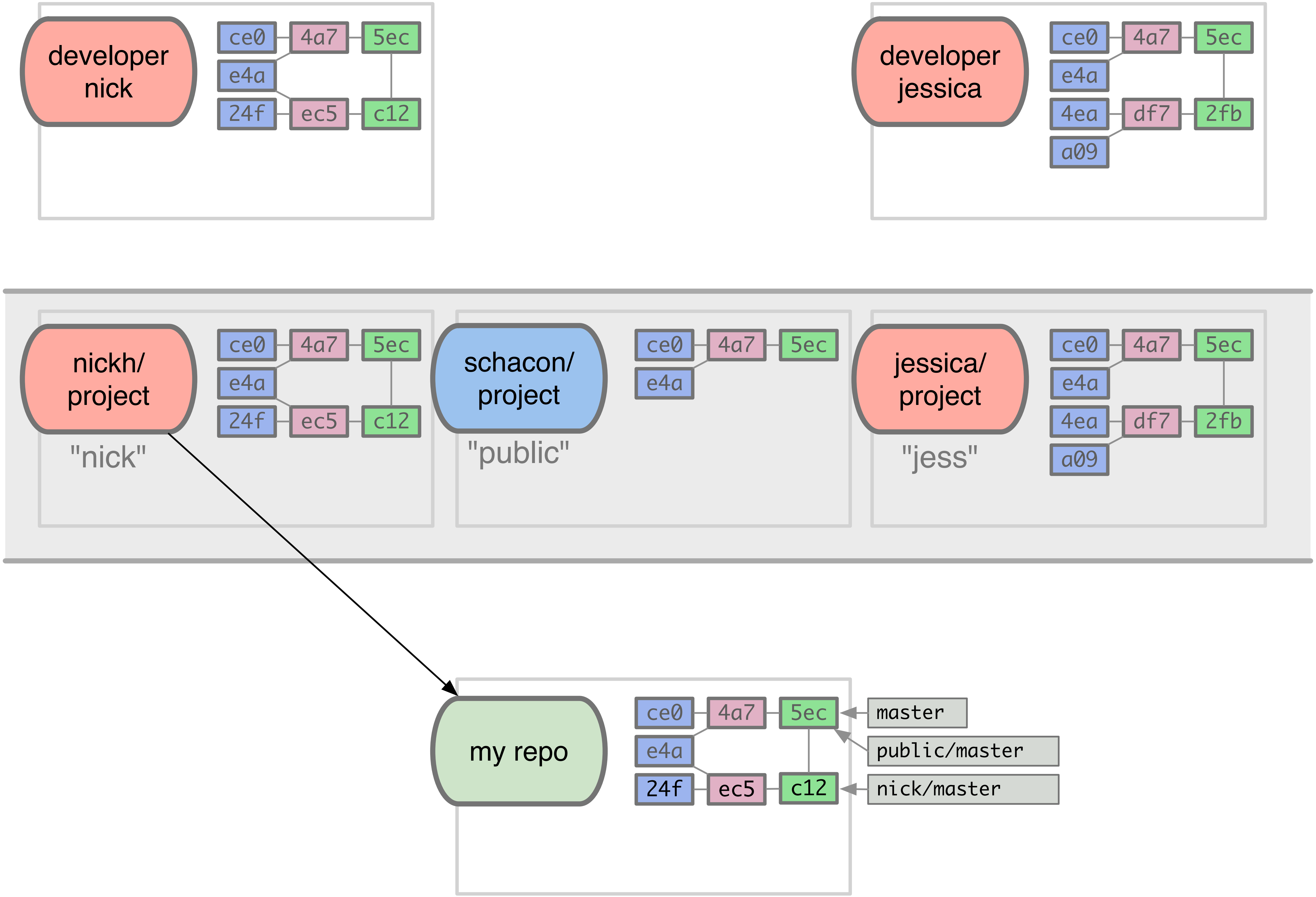
```
git remote add jess git://github.com/jessica/project.git
```

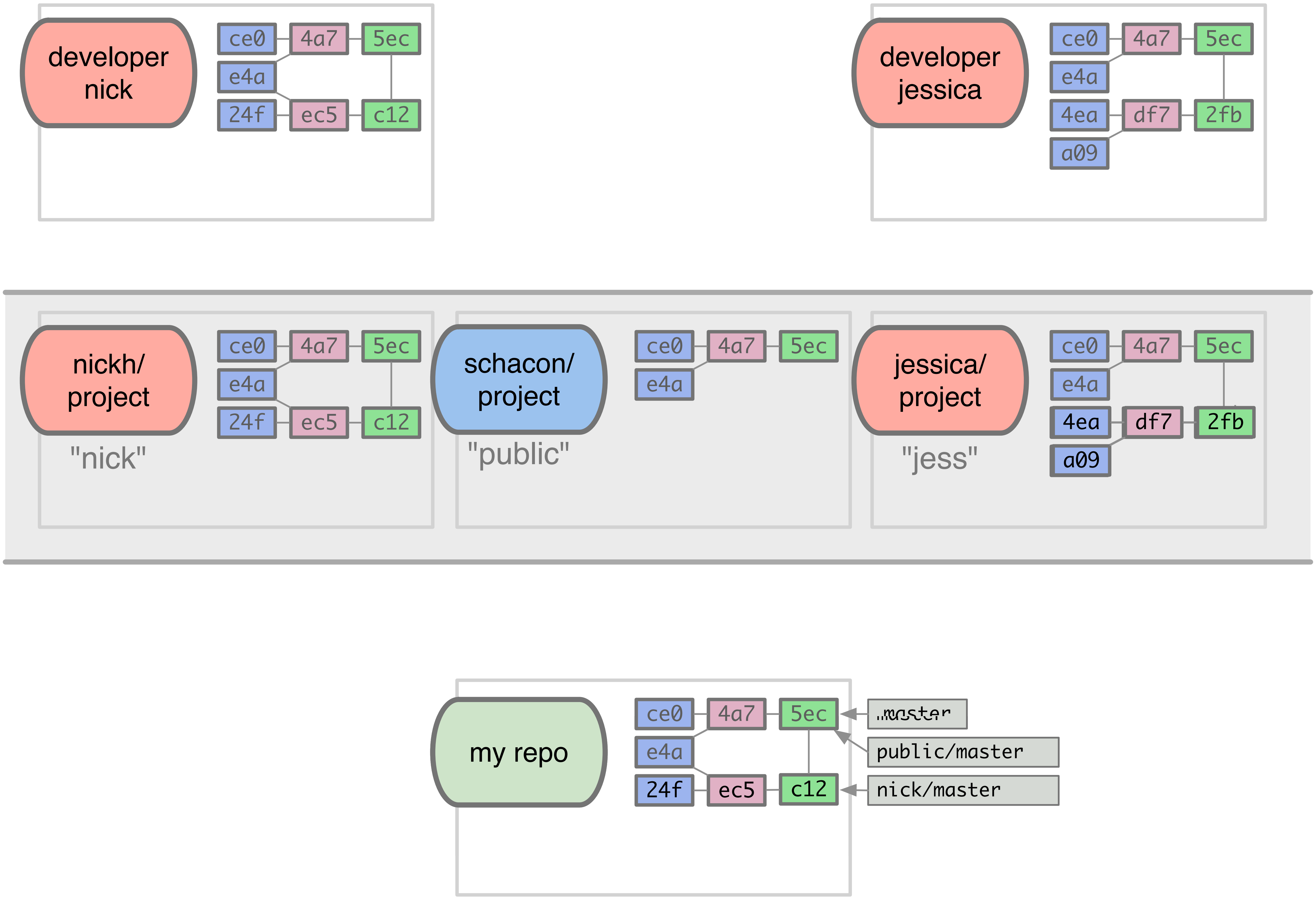


git fetch nick

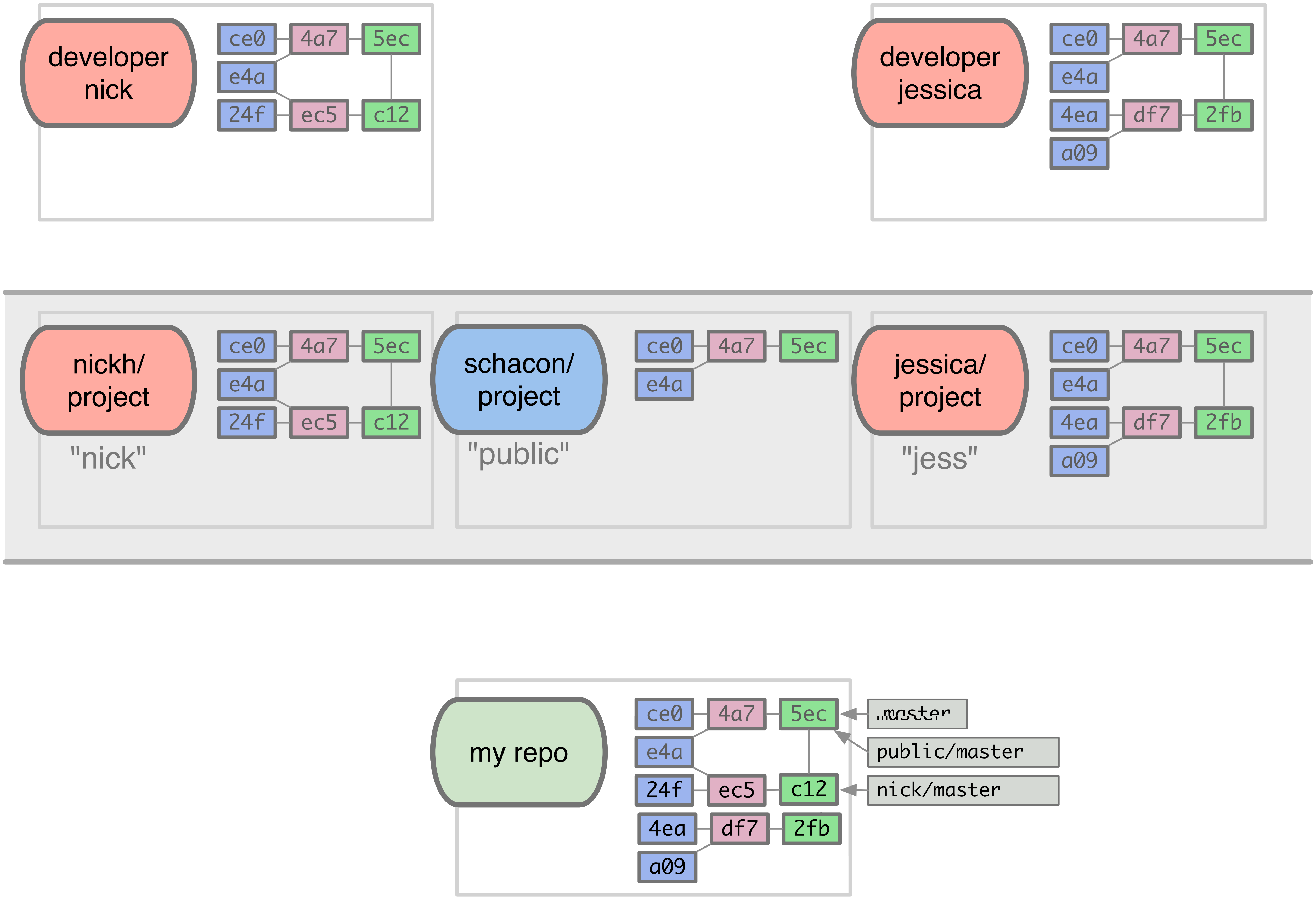


git fetch nick

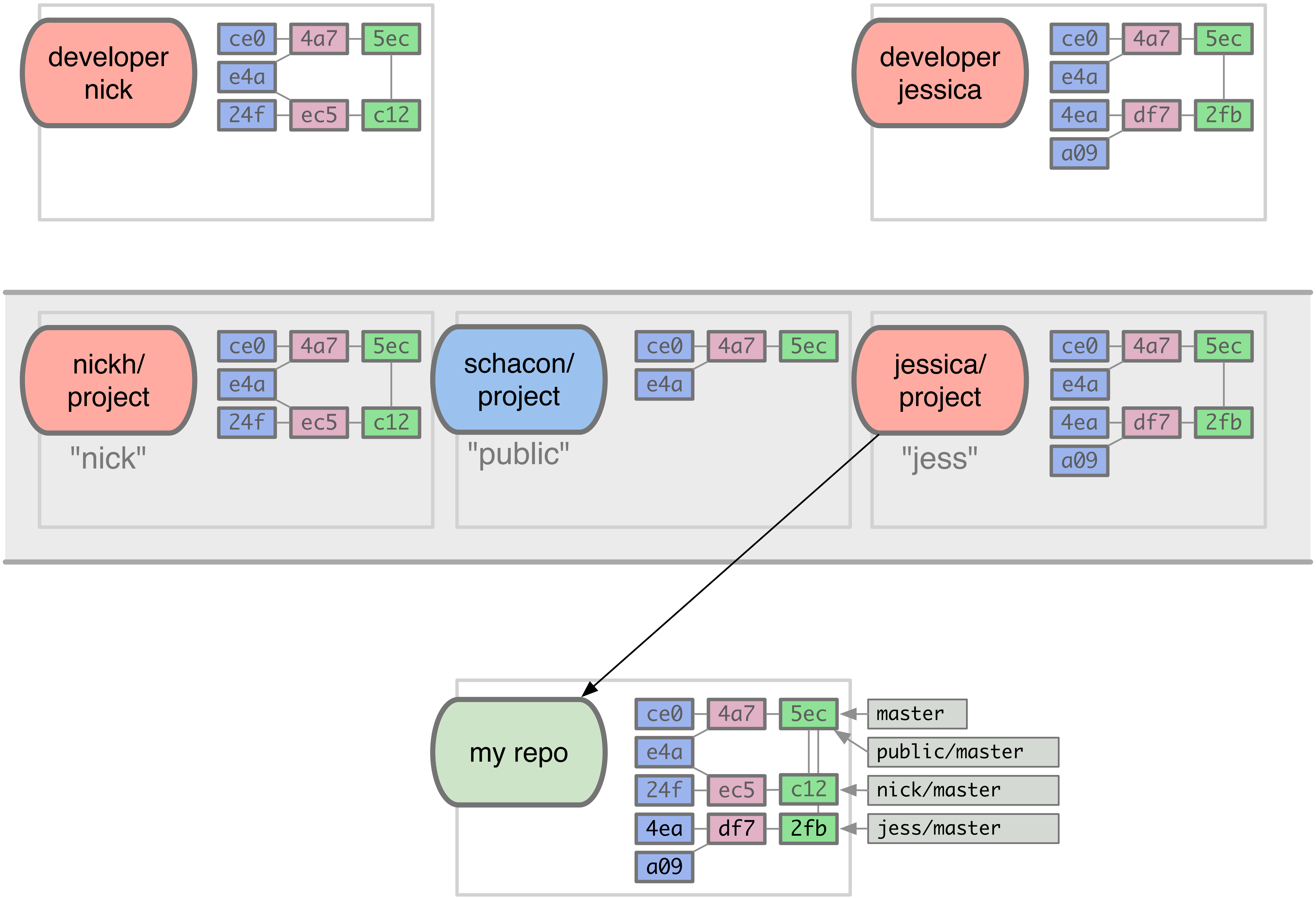

developer nick
ce0
4a7
5ec
e4a
24f
ec5
c12
developer jessica
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
nickh/ project
"nick"
ce0
4a7
5ec
e4a
24f
ec5
c12
schacon/ project
"public"
ce0
4a7
5ec
e4a
jessica/ project
"jess"
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
my repo
ce0
4a7
5ec
e4a
24f
ec5
c12
master
public/master
nick/master

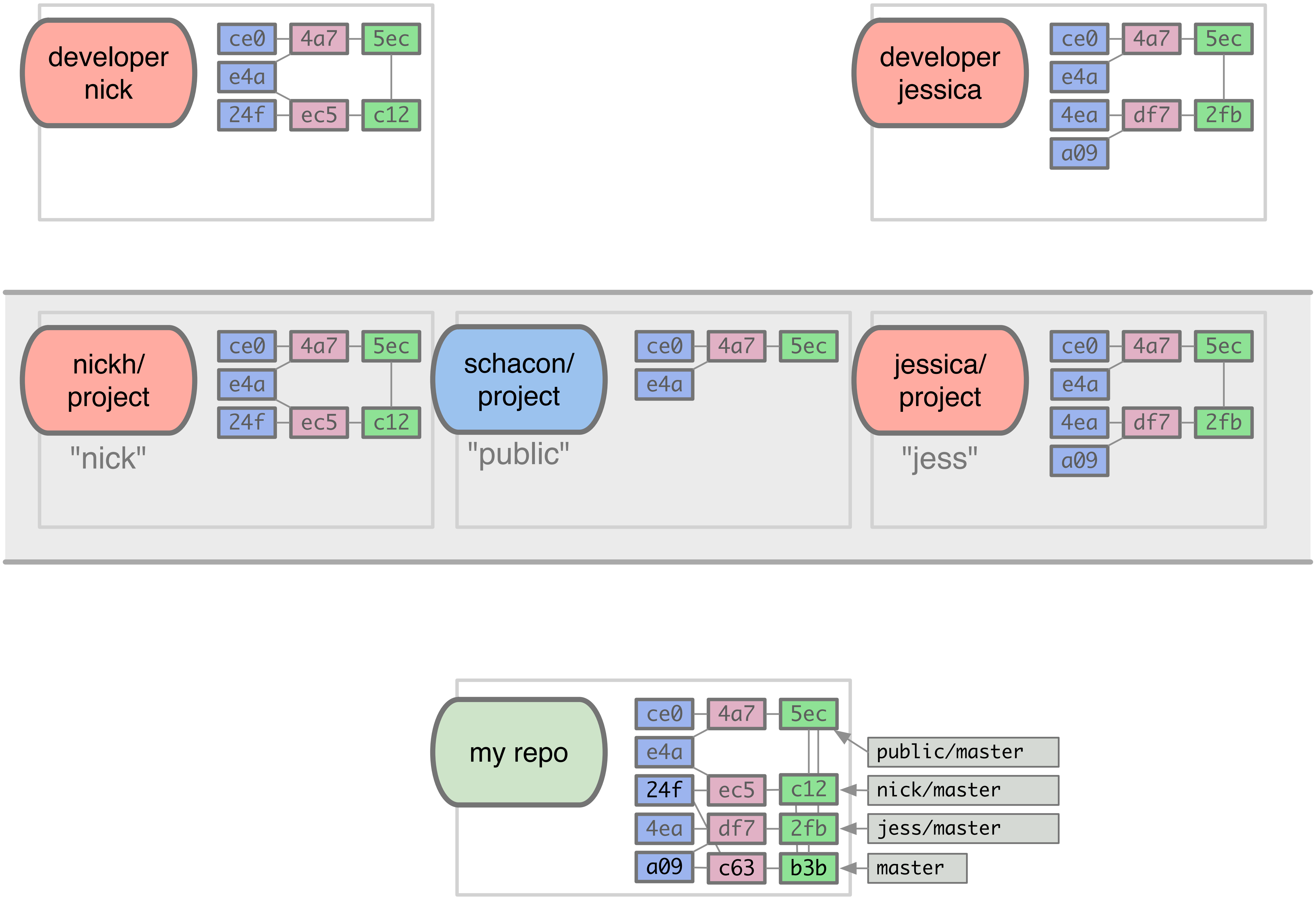


git fetch jess

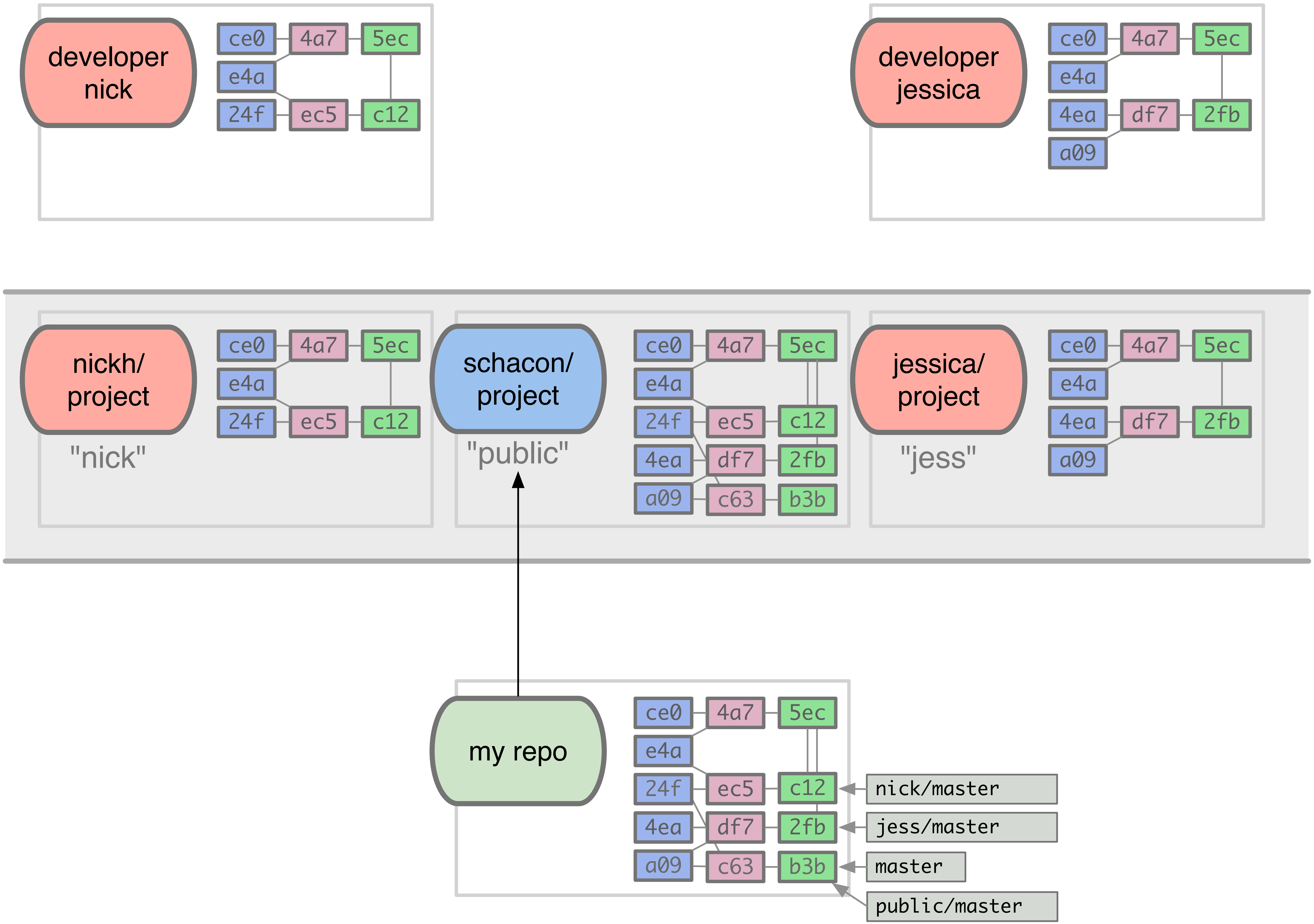


git fetch jess

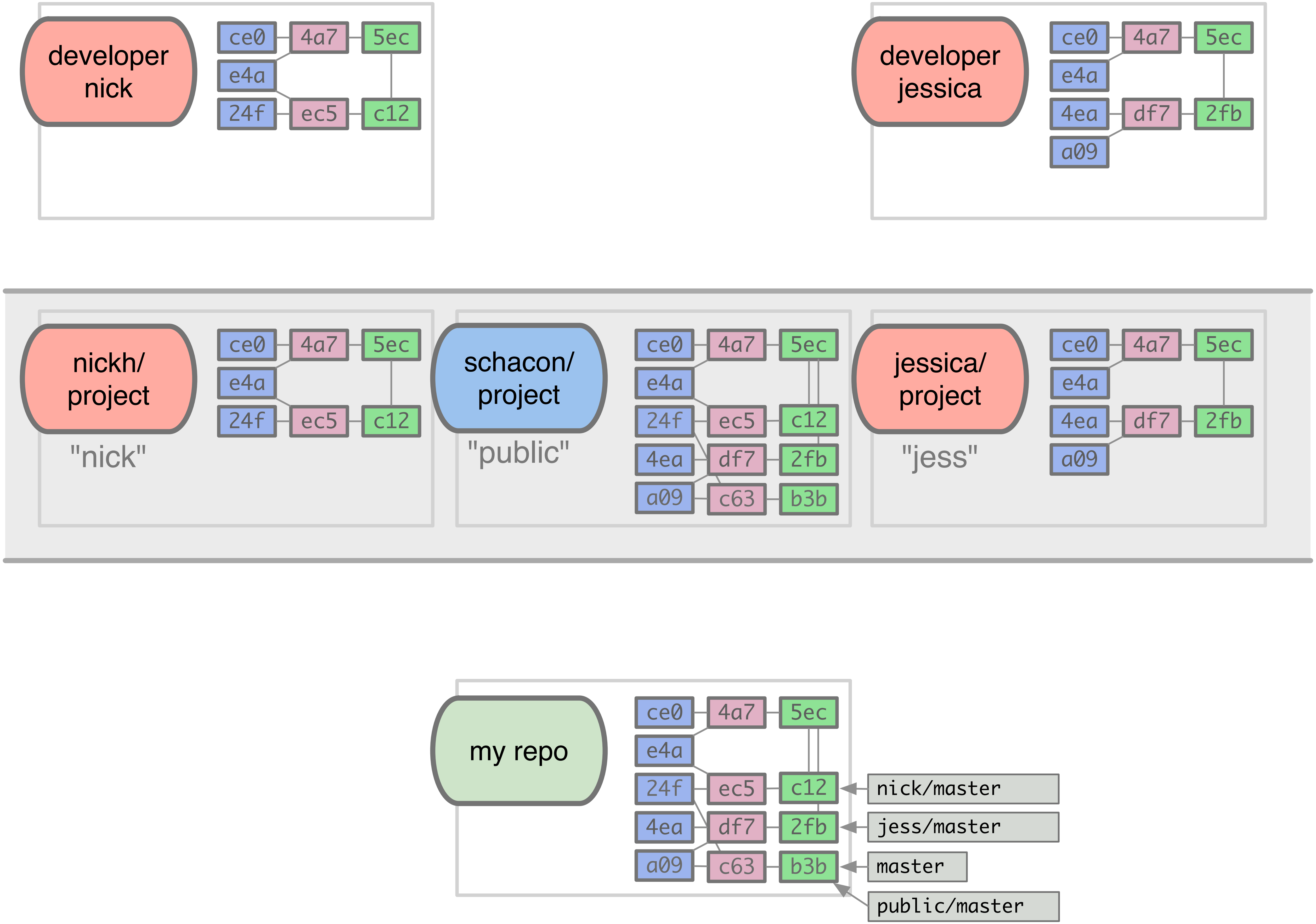

developer nick
ce0
4a7
5ec
e4a
24f
ec5
c12
developer jessica
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
nickh/ project
"nick"
ce0
4a7
5ec
e4a
24f
ec5
c12
schacon/ project
"public"
ce0
4a7
5ec
e4a
jessica/ project
"jess"
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
my repo
ce0
4a7
5ec
e4a
24f
ec5
c12
4ea
df7
2fb
a09
master
public/master
nick/master
jess/master

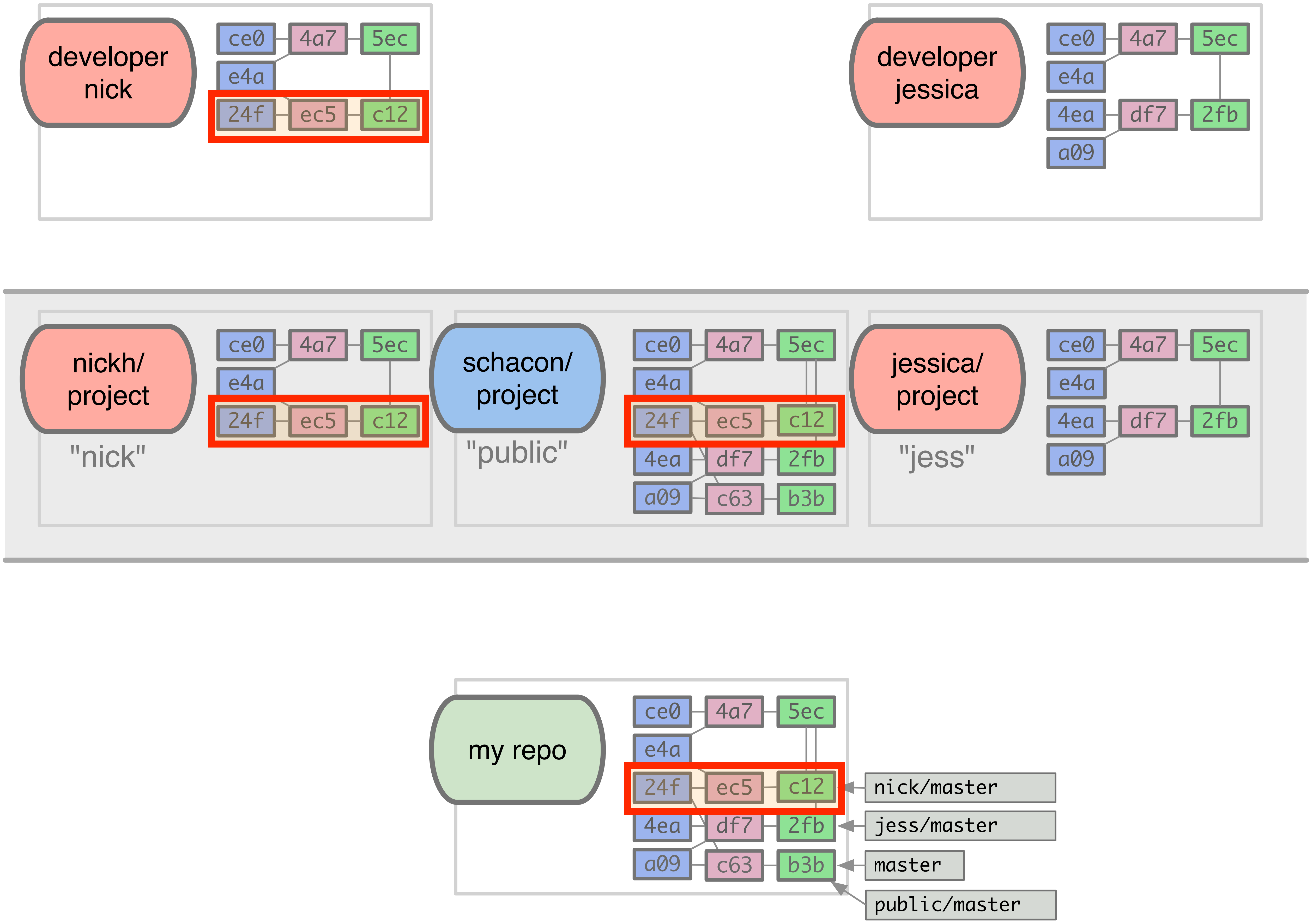


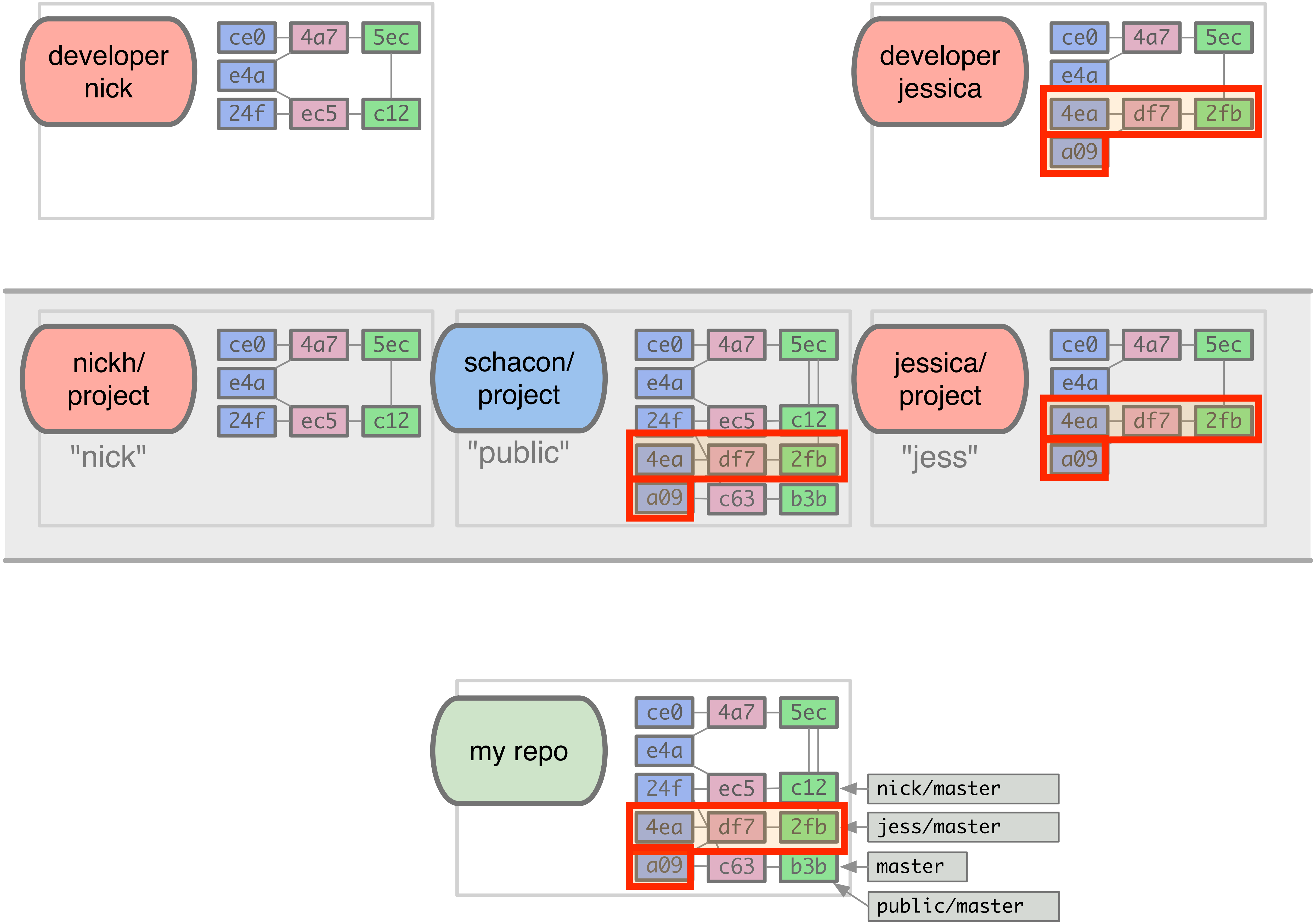
```
git merge nick jess
```

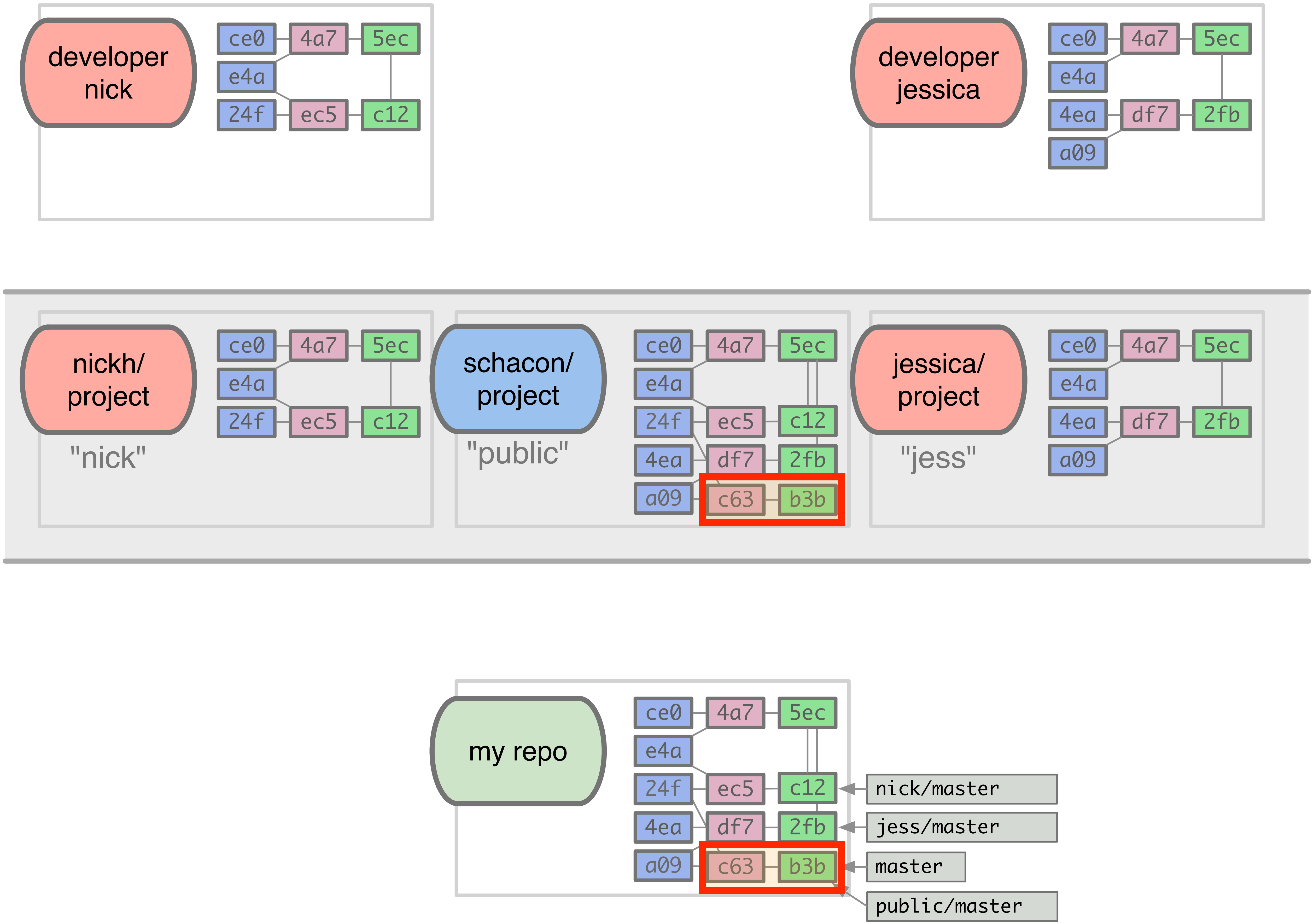


git push public

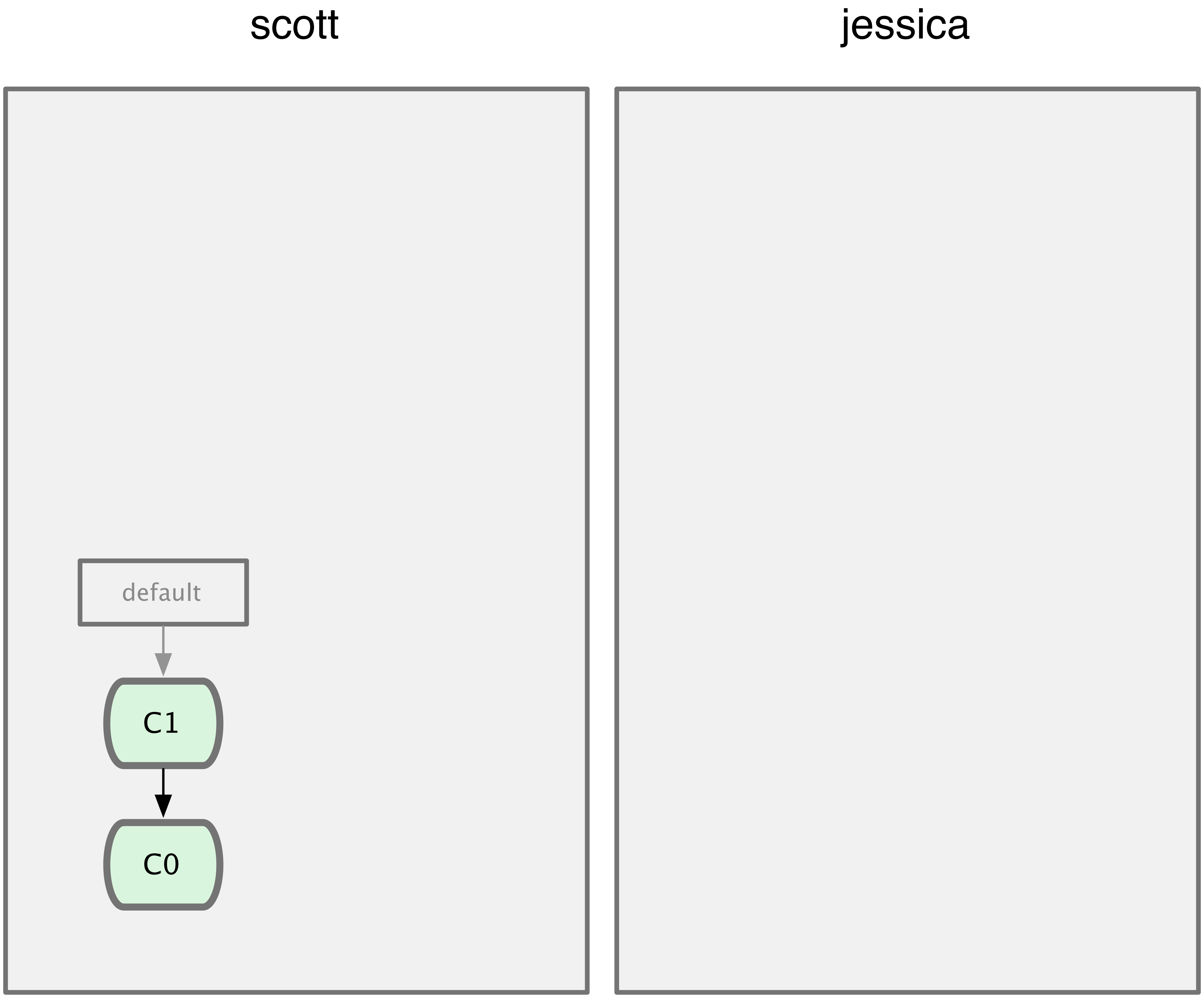

developer nick
ce0
4a7
5ec
e4a
24f
ec5
c12
developer jessica
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
nickh/ project
"nick"
ce0
4a7
5ec
e4a
24f
ec5
c12
schacon/ project
"public"
ce0
4a7
5ec
e4a
24f
ec5
c12
4ea
df7
2fb
a09
c63
b3b
jessica/ project
"jess"
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
my repo
ce0
4a7
5ec
e4a
24f
ec5
c12
4ea
df7
2fb
a09
c63
b3b
nick/master
jess/master
master
public/master

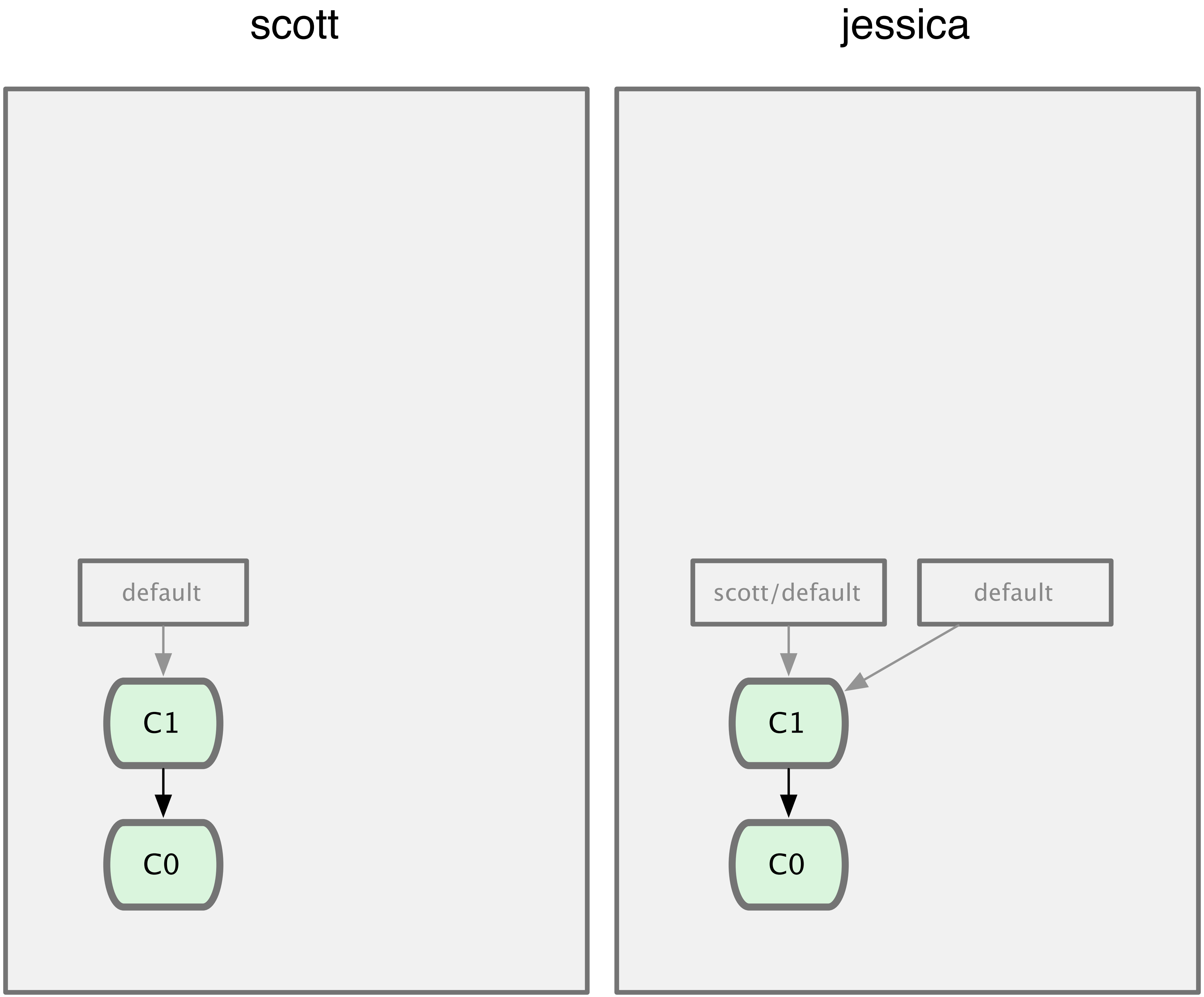

developer nick
ce0
4a7
5ec
e4a
24f
ec5
c12
developer jessica
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
nickh/ project
"nick"
ce0
4a7
5ec
e4a
24f
ec5
c12
schacon/ project
"public"
ce0
4a7
5ec
e4a
24f
ec5
c12
4ea
df7
2fb
a09
c63
b3b
jessica/ project
"jess"
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
my repo
ce0
4a7
5ec
e4a
24f
ec5
c12
4ea
df7
2fb
a09
c63
b3b
nick/master
jess/master
master
public/master

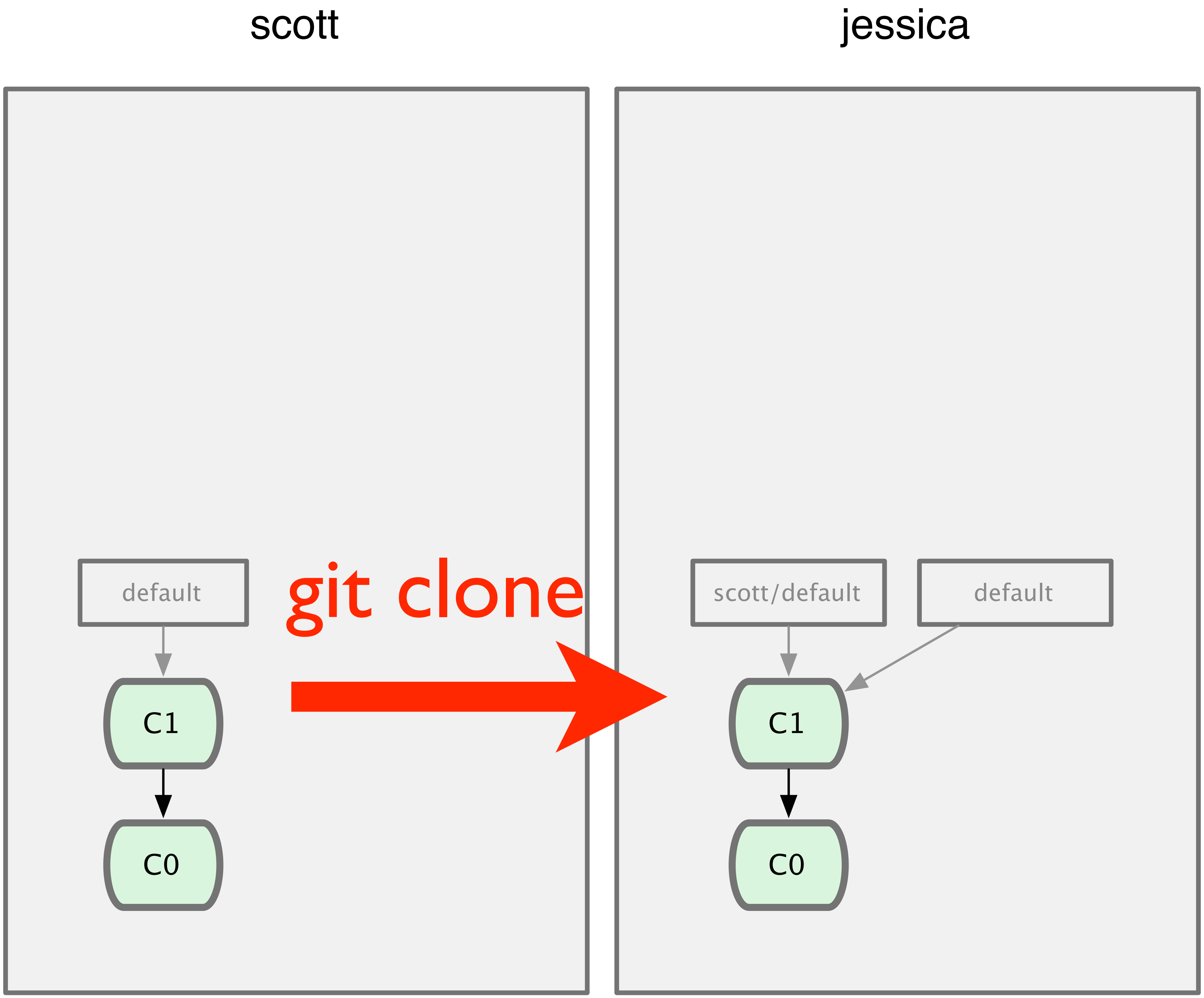

developer nick
ce0
4a7
5ec
e4a
24f
ec5
c12
developer jessica
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
nickh/ project
"nick"
ce0
4a7
5ec
e4a
24f
ec5
c12
schacon/ project
"public"
ce0
4a7
5ec
e4a
24f
ec5
c12
4ea
df7
2fb
a09
c63
b3b
jessica/ project
"jess"
ce0
4a7
5ec
e4a
4ea
df7
2fb
a09
my repo
ce0
4a7
5ec
e4a
24f
ec5
c12
4ea
df7
2fb
a09
c63
b3b
nick/master
jess/master
master
public/master

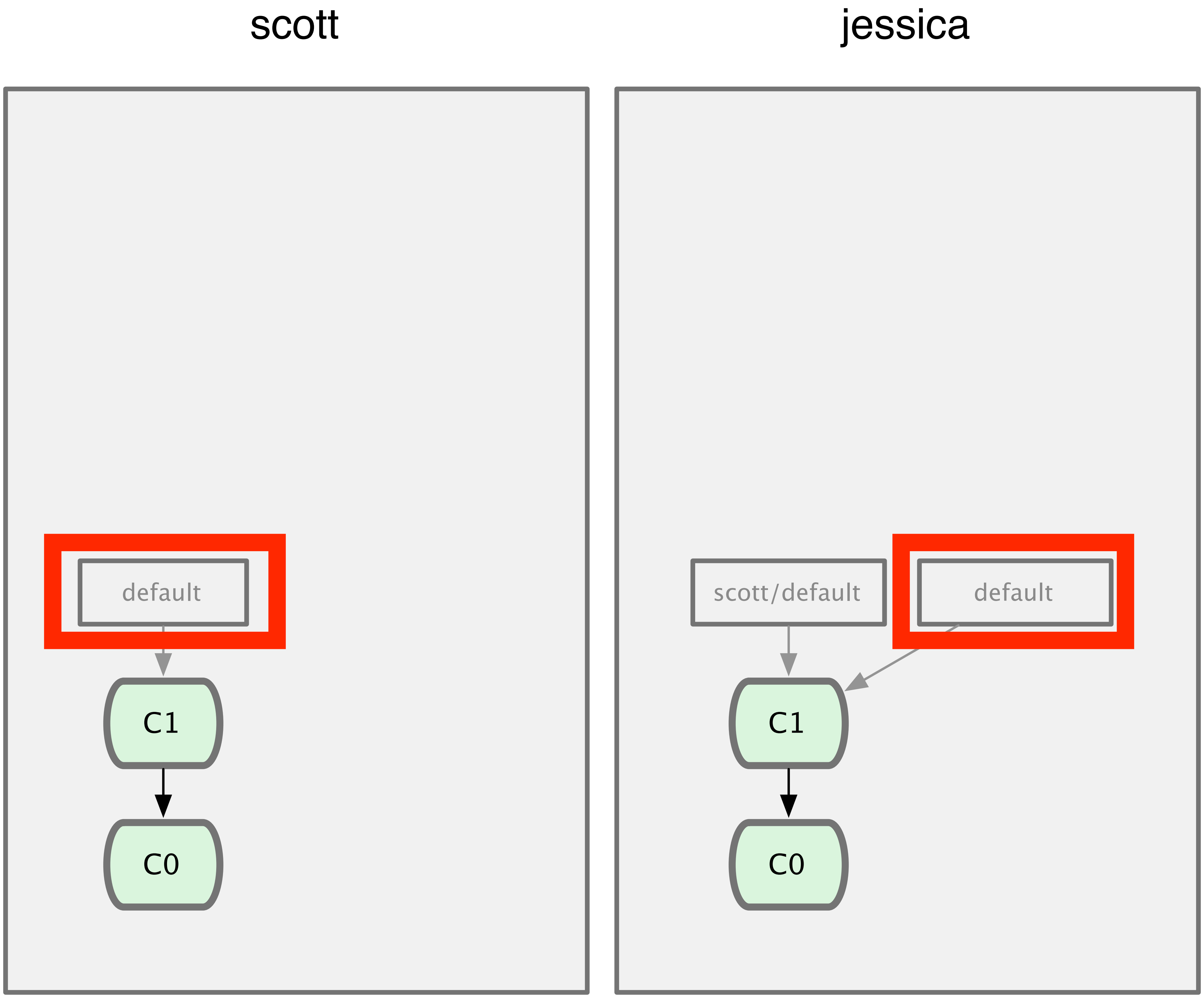

developer nick
ce0 4a7 5ec
e4a
24f ec5 c12
developer jessica
ce0 4a7 5ec
e4a
4ea df7 2fb
a09
nickh/ project
"nick"
ce0 4a7 5ec
e4a
24f ec5 c12
schacon/ project
"public"
ce0 4a7 5ec
e4a
24f ec5 c12
4ea df7 2fb
a09 c63 b3b
jessica/ project
"jess"
ce0 4a7 5ec
e4a
4ea df7 2fb
a09
my repo
ce0 4a7 5ec
e4a
24f ec5 c12
4ea df7 2fb
a09 c63 b3b
nick/master
jess/master
master
public/master

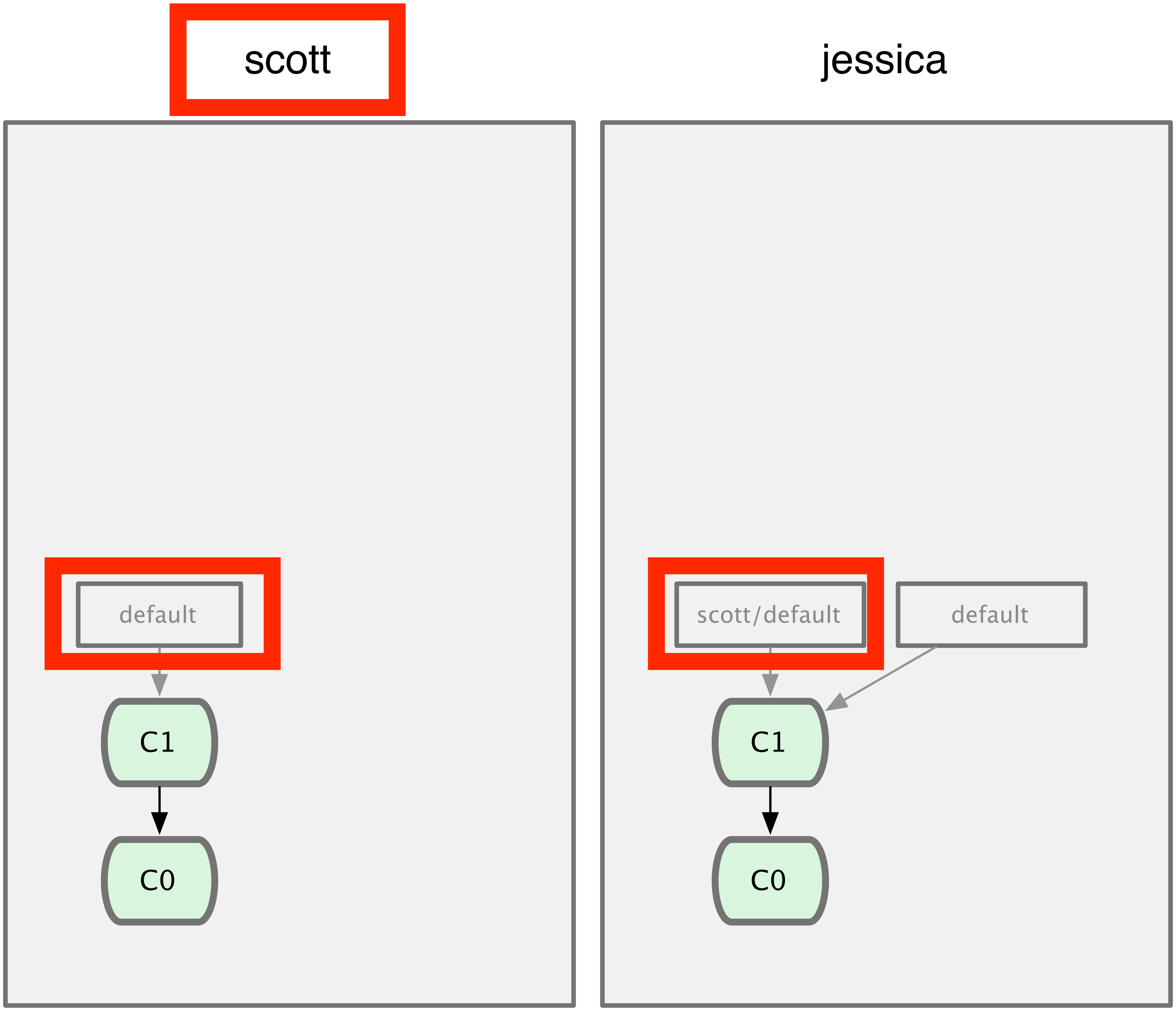
Remotes Are Branches

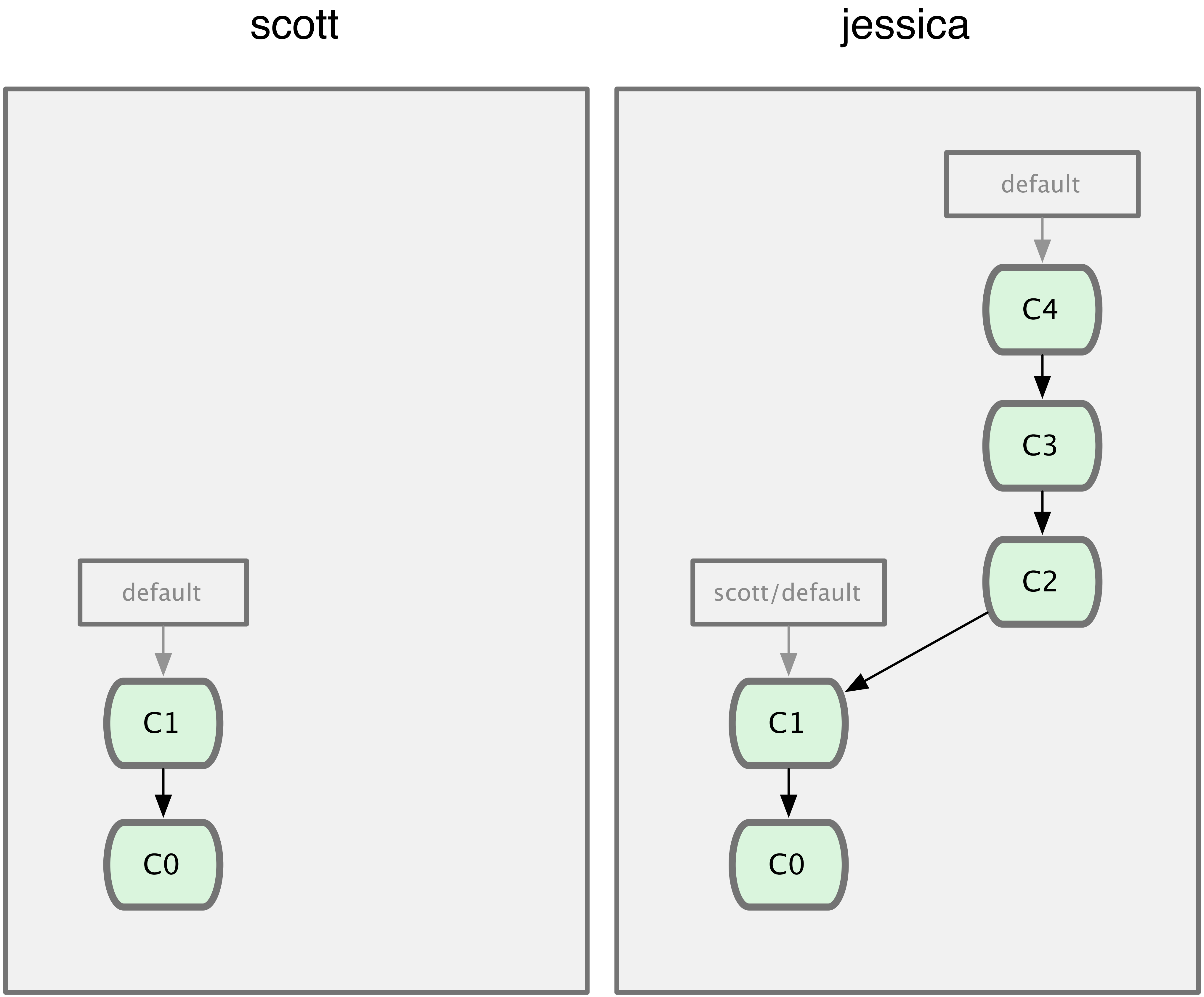

scott
jessica
default
C1
C0

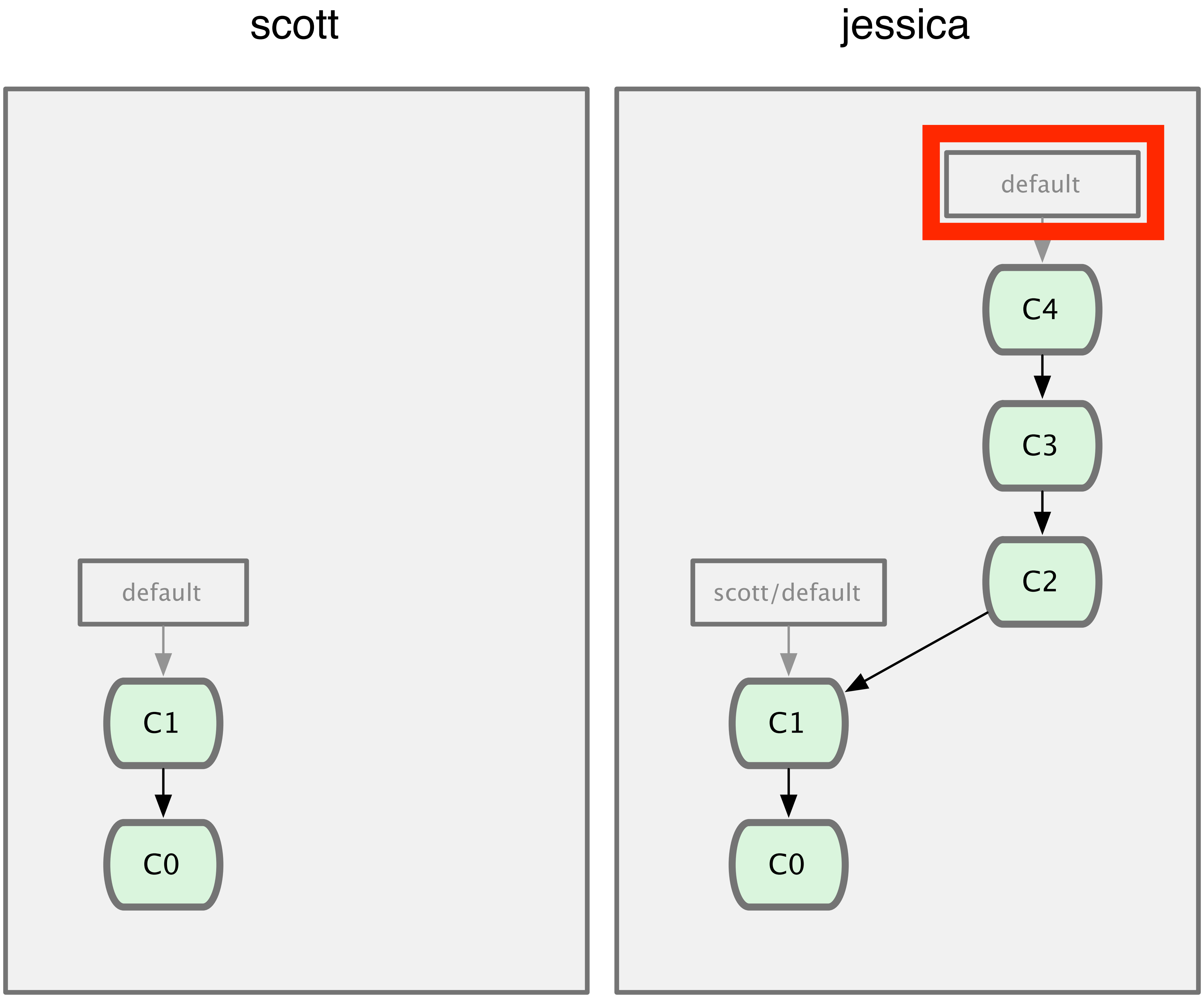

scott
jessica
default
C1
C0
scott/default
default
C1
C0

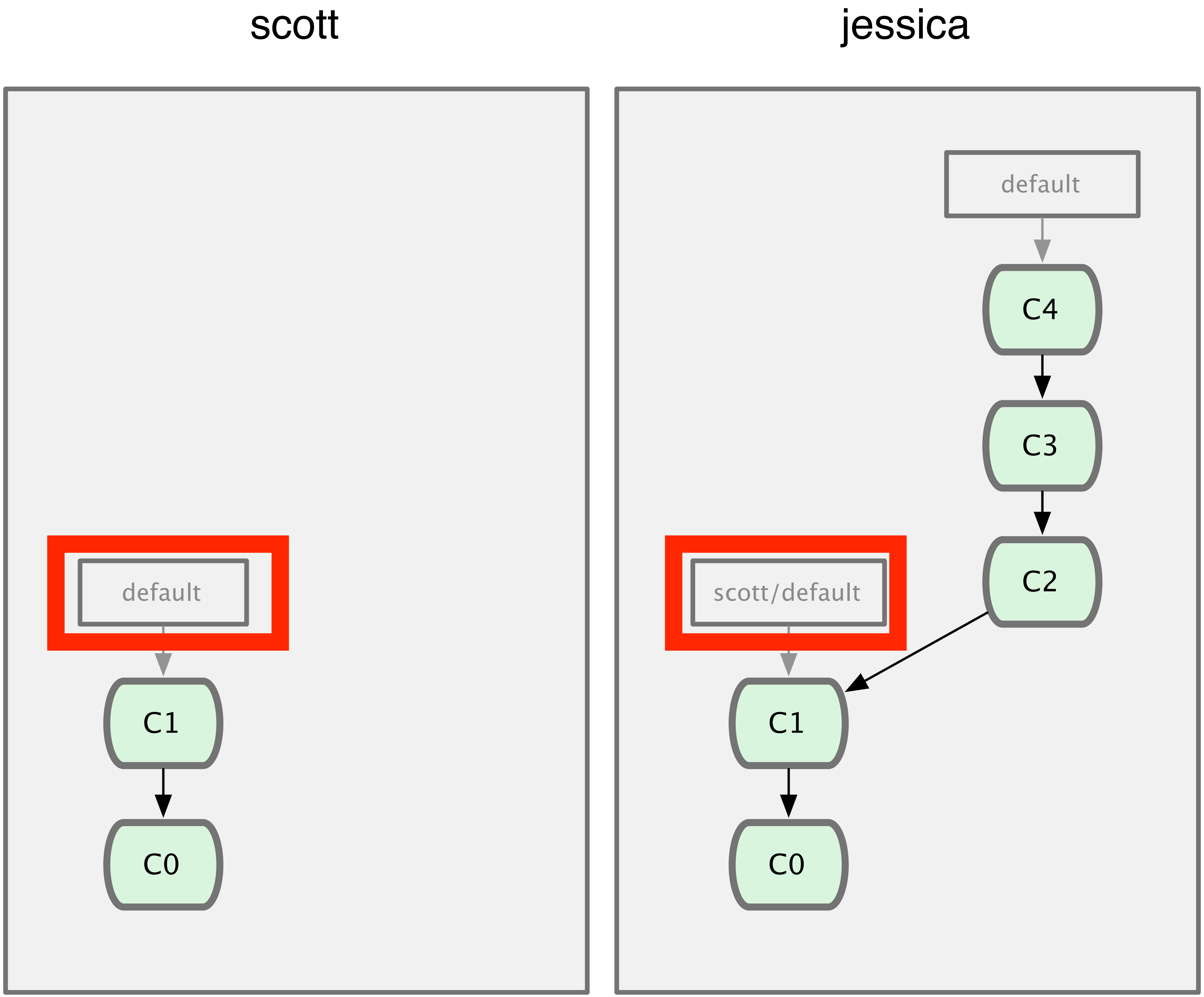

scott
jessica
default
git clone
scott/default
default
C1
C0
C1
C0

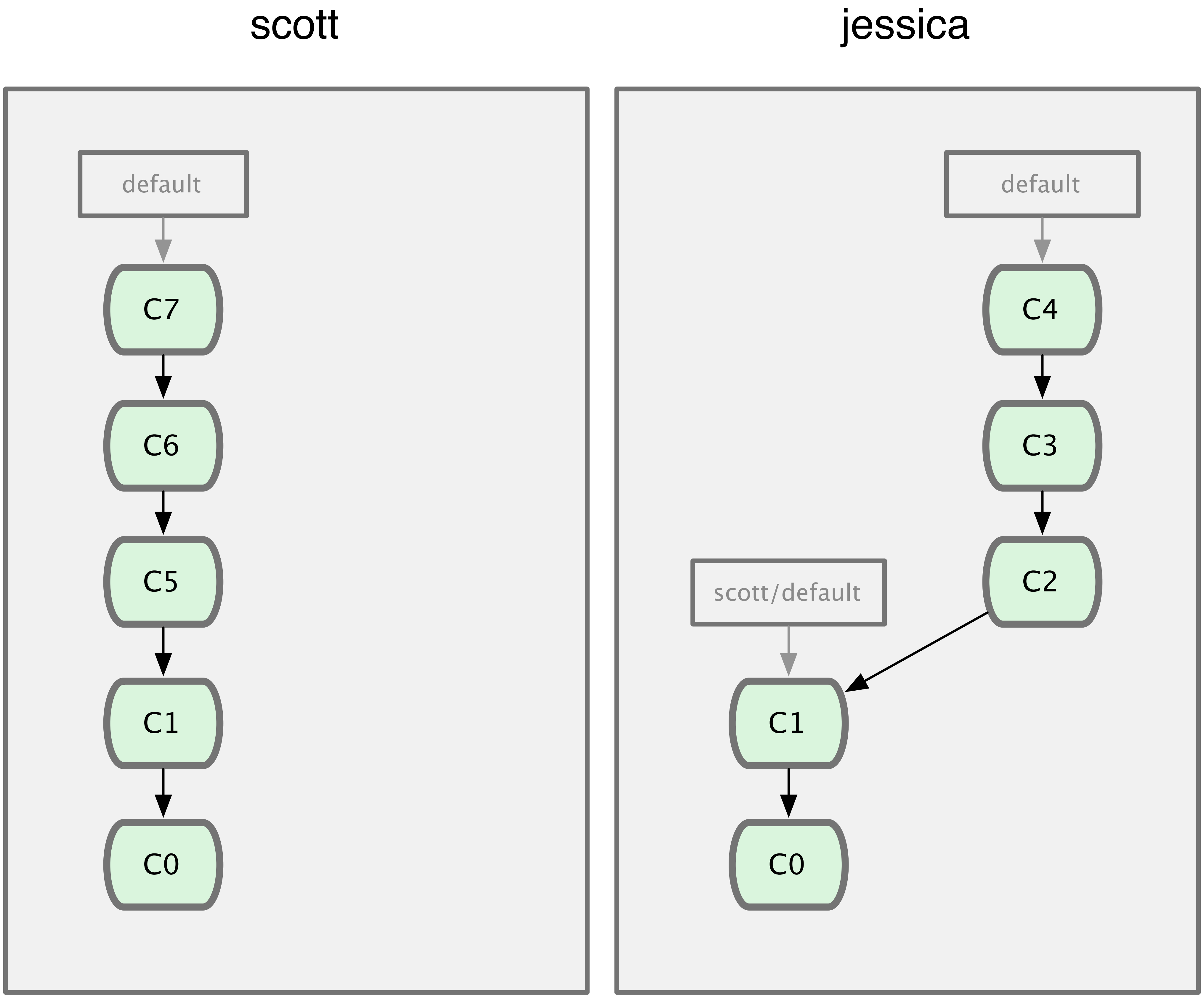

scott
jessica
default
C1
C0
scott/default
default
C1
C0


scott
jessica
default
C1
C0
scott/default
default
C1
C0

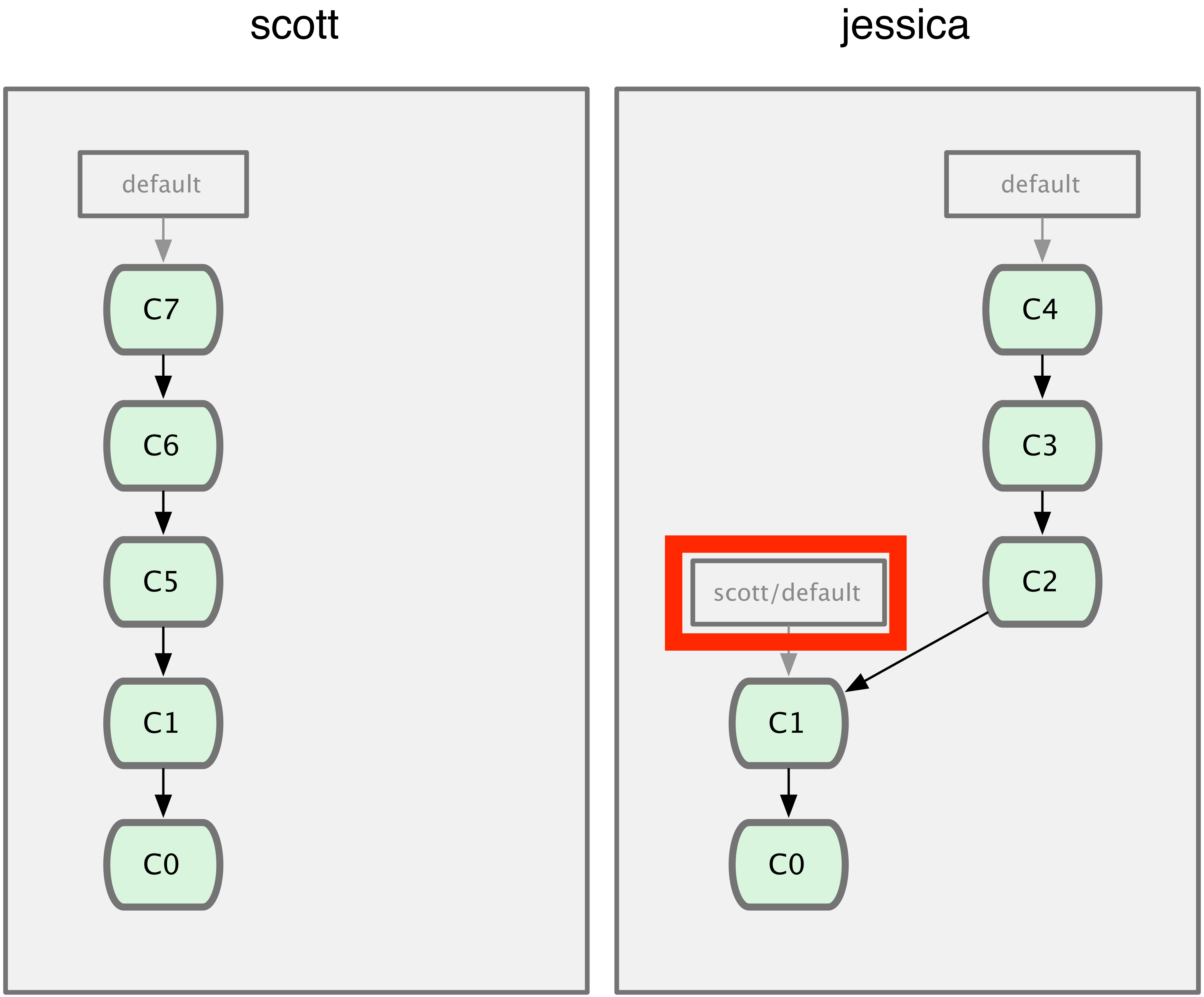

scott
jessica
default
C1
C0
default
C4
C3
C2
scott/default
C1
C0

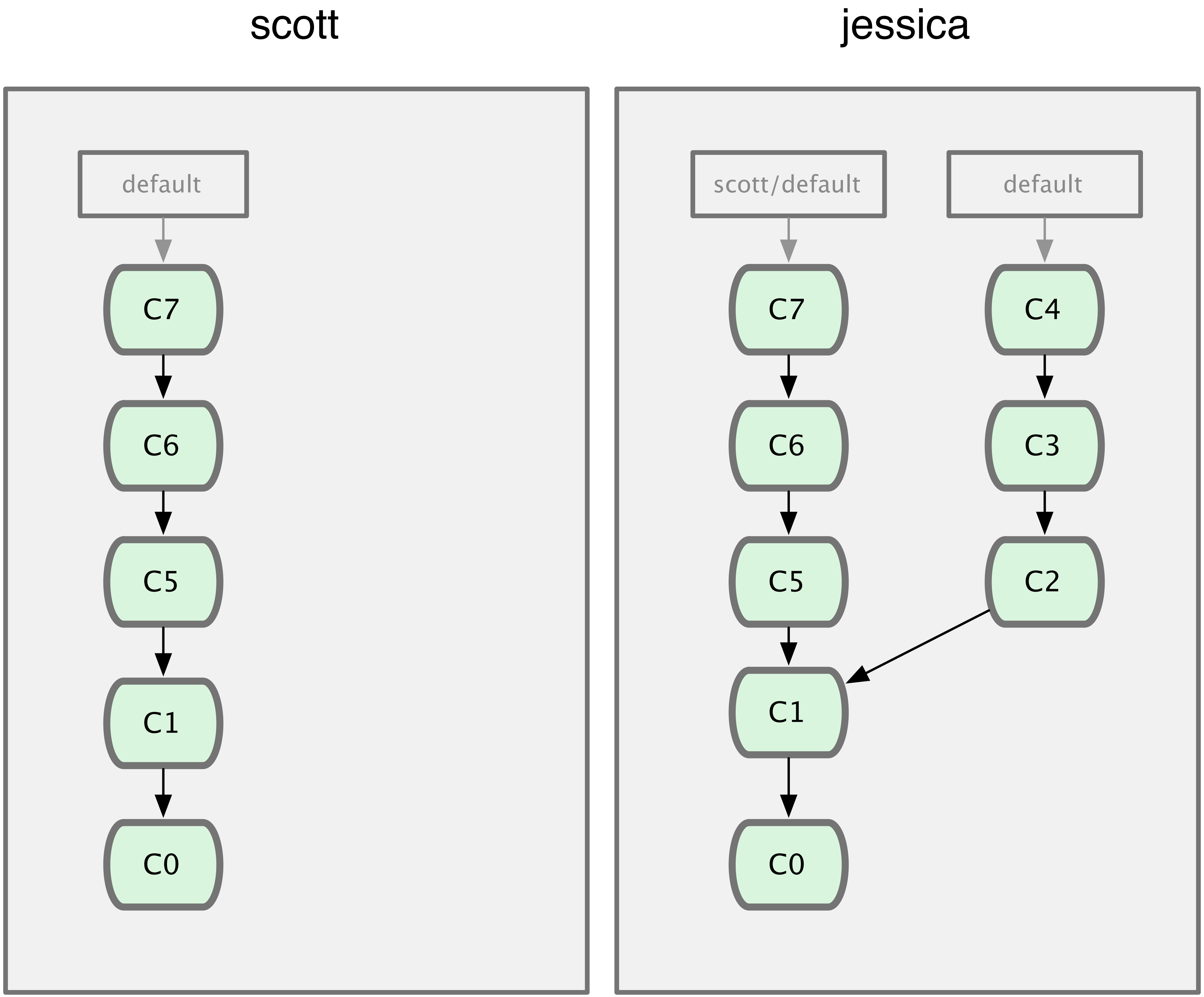

scott
jessica
default
C1
C0
default
C4
C3
C2
scott/default
C1
C0

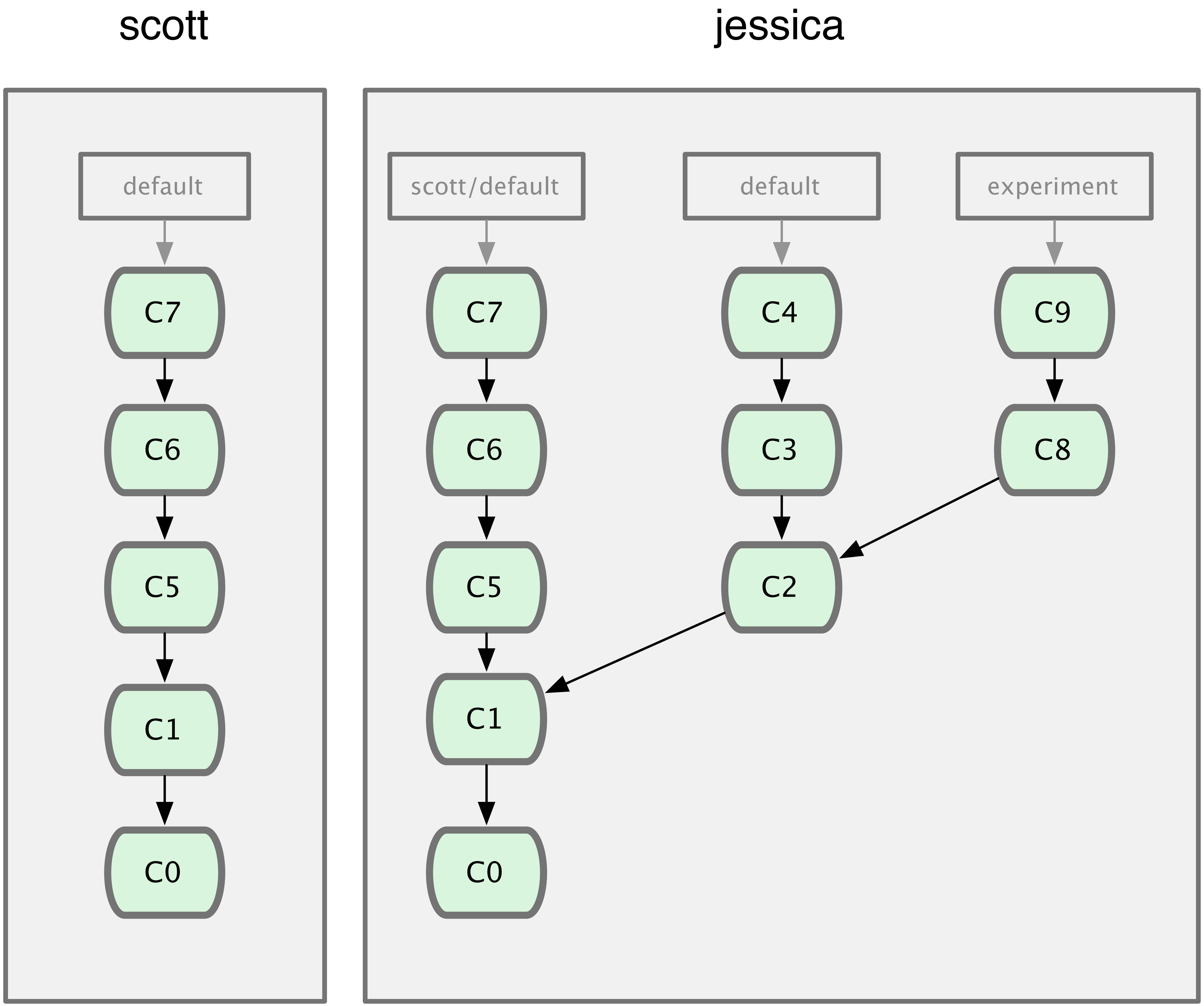

scott
jessica
default
C1
C0
default
C4
C3
C2
scott/default
C1
C0

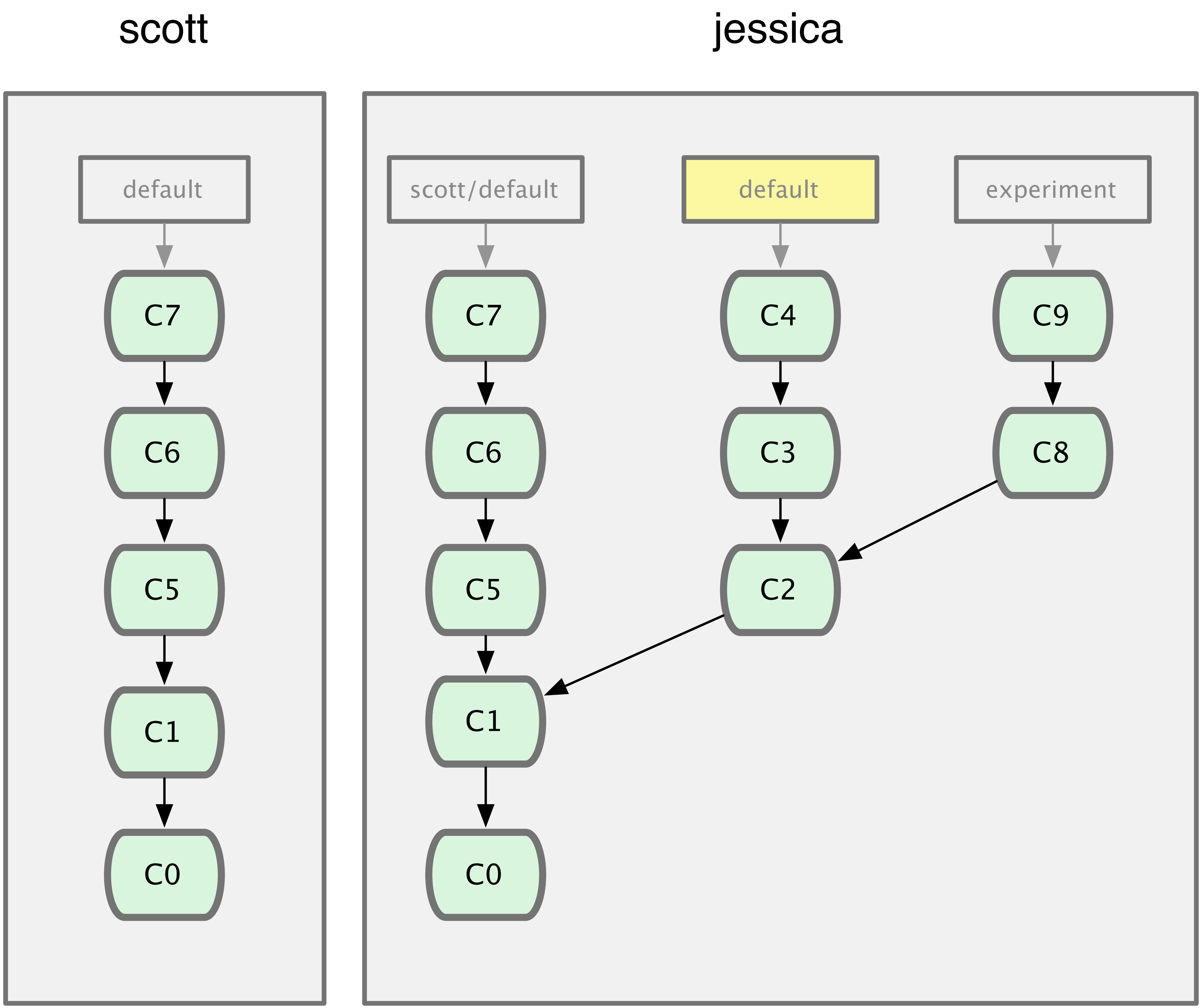


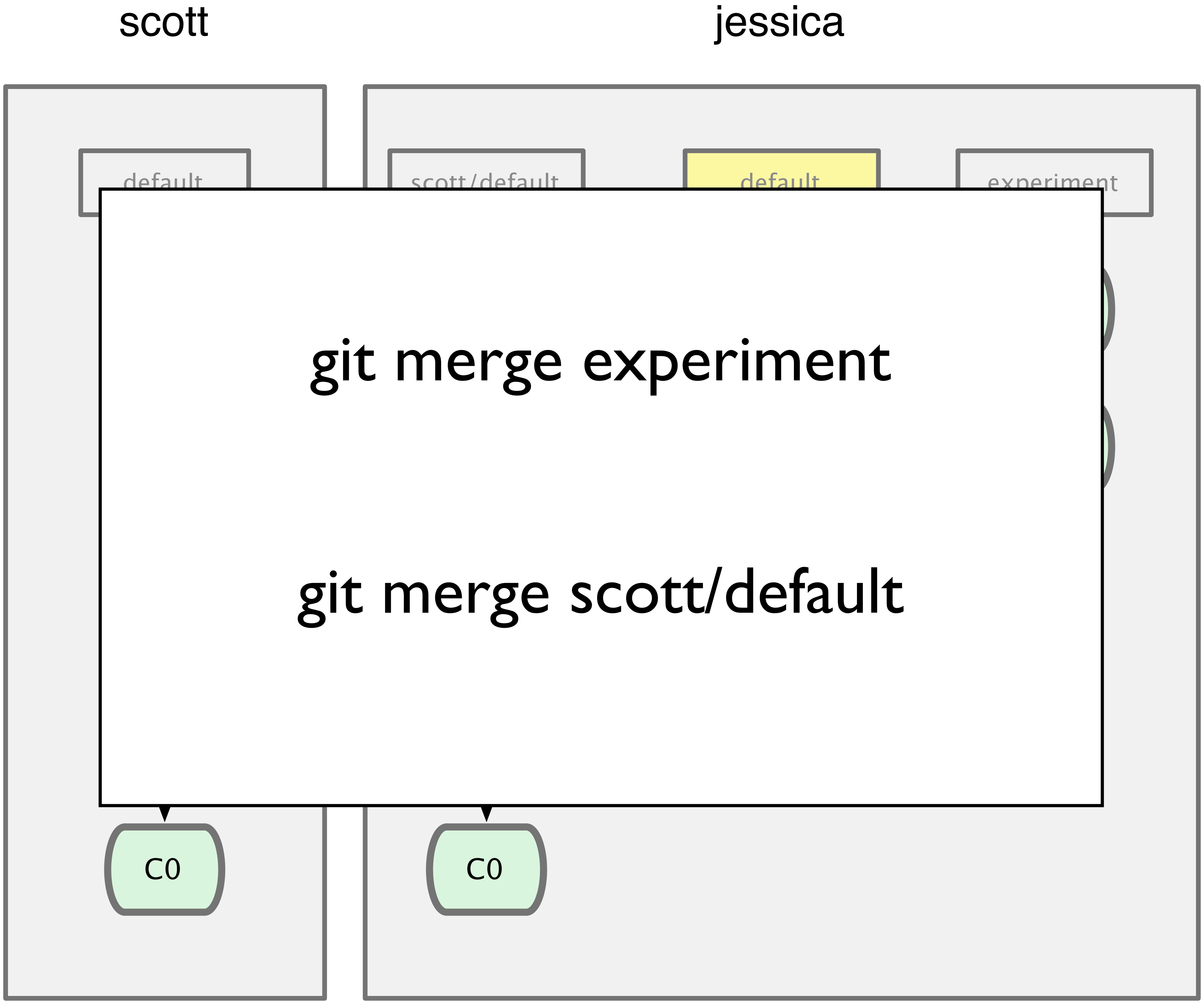
scott
jessica
default
C7
C6
C5
C1
C0
default
C4
C3
C2
scott/default
C1
C0


scott
jessica
default
C7
C6
C5
C1
C0
default
C4
C3
C2
scott/default
C1
C0

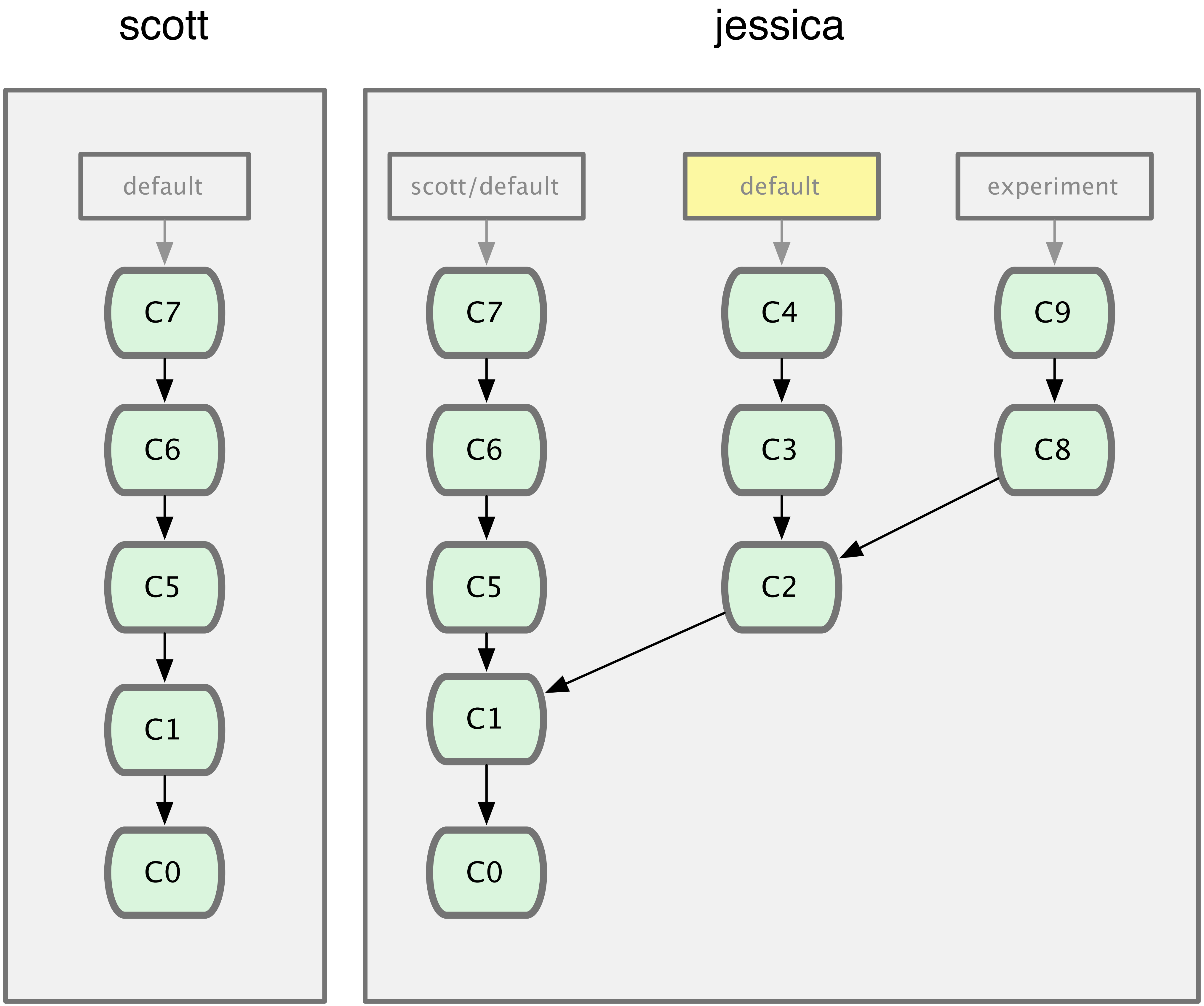

scott
jessica
default
C7
C6
C5
C1
C0
scott/default
default
C7
C6
C5
C1
C0
C4
C3
C2

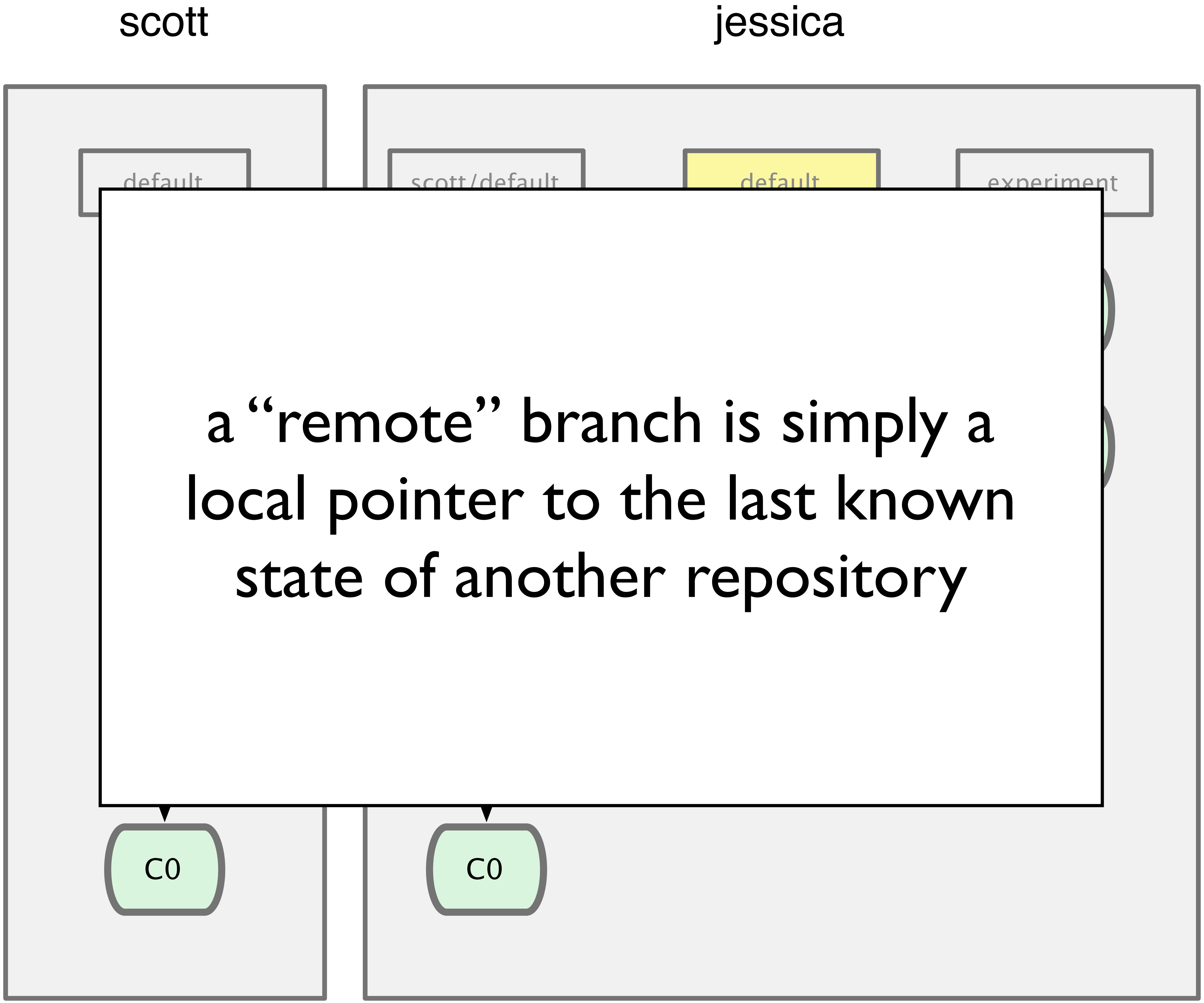

scott
jessica
default
C7
C6
C5
C1
C0
scott/default
C7
C6
C5
C1
C0
default
C4
C3
C2
experiment
C9
C8


scott
jessica
default
C7
C6
C5
C1
C0
scott/default
C7
C6
C5
C1
C0
default
C4
C3
C2
experiment
C9
C8



```
git merge experiment

git merge scott/default
```


scott
jessica
default
C7
C6
C5
C1
C0
scott/default
C7
C6
C5
C1
C0
default
C4
C3
C2
experiment
C9
C8

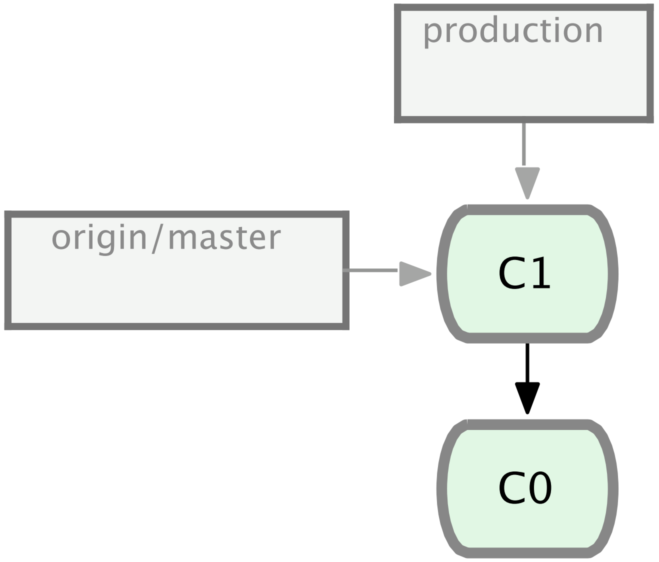


a “remote” branch is simply a local pointer to the last known state of another repository

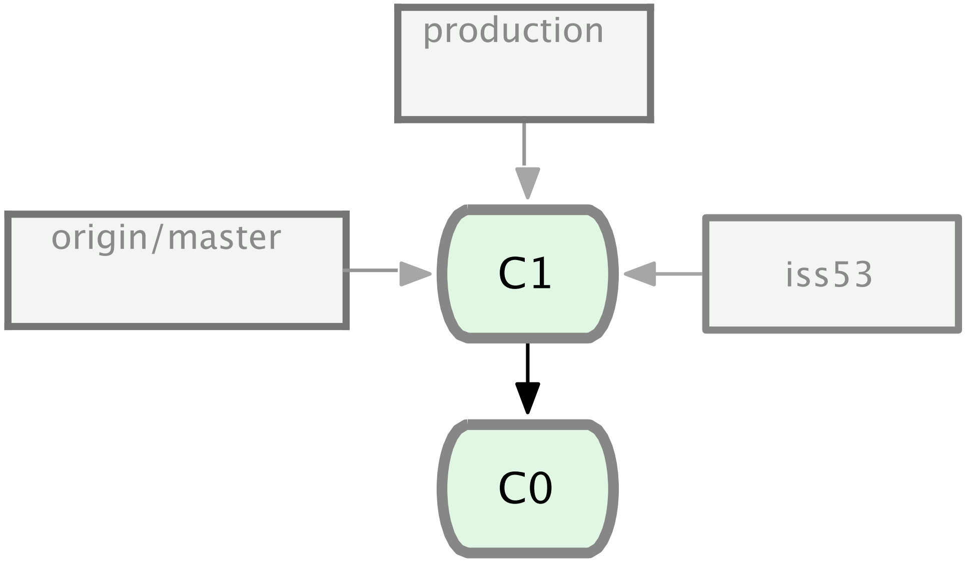
Why is this cool?

non-linear development

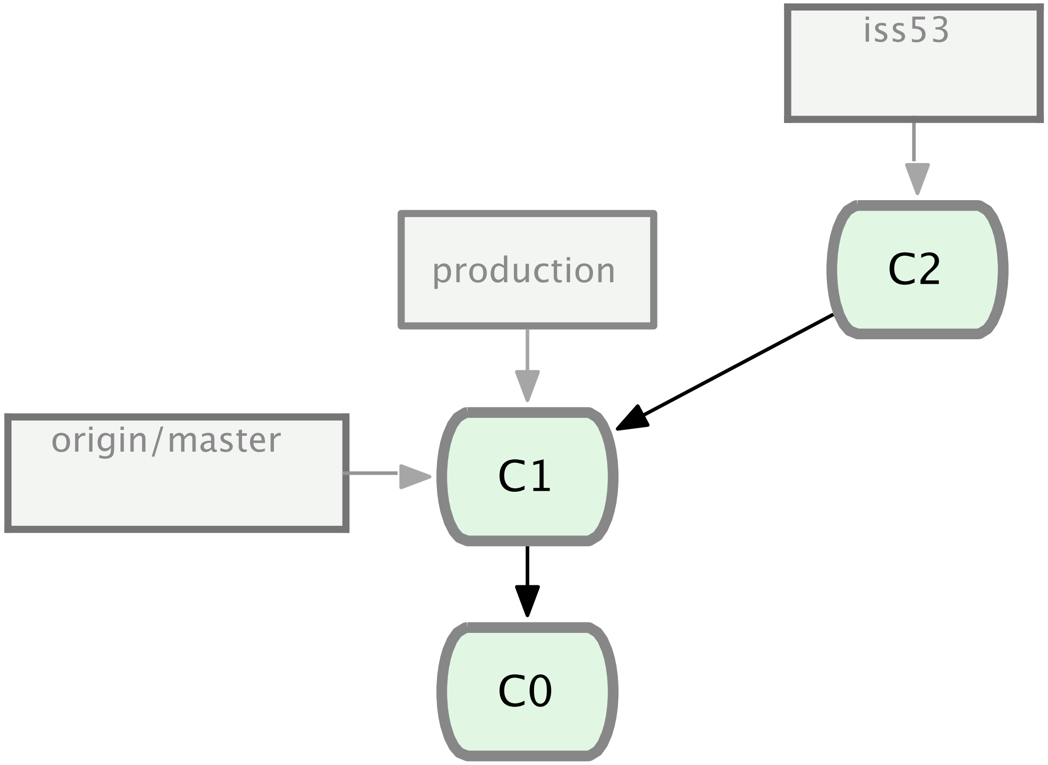
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout 'production'
- create a branch (iss102)
- fix the issue
- checkout 'production', merge 'iss102'
- push 'production'
- etc



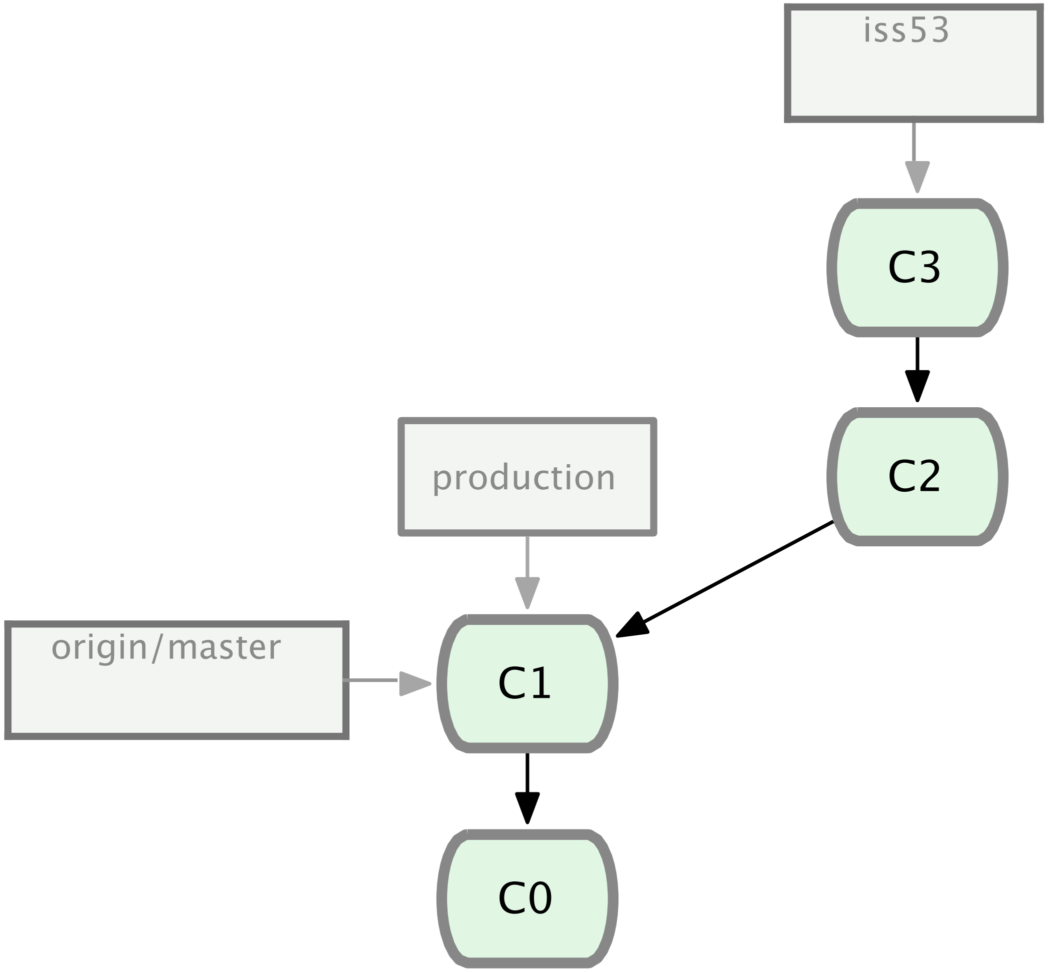
git clone



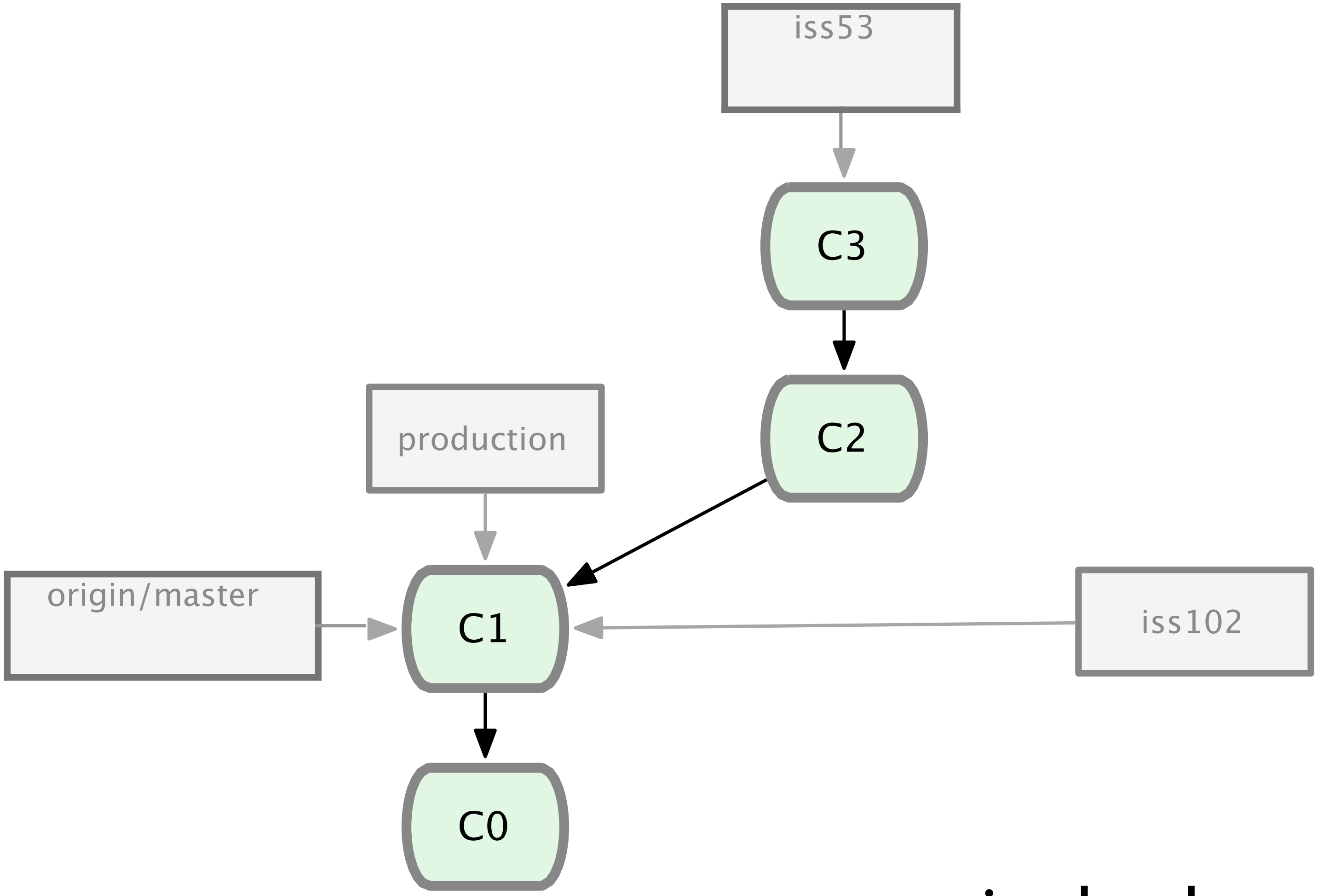
git checkout -b iss53



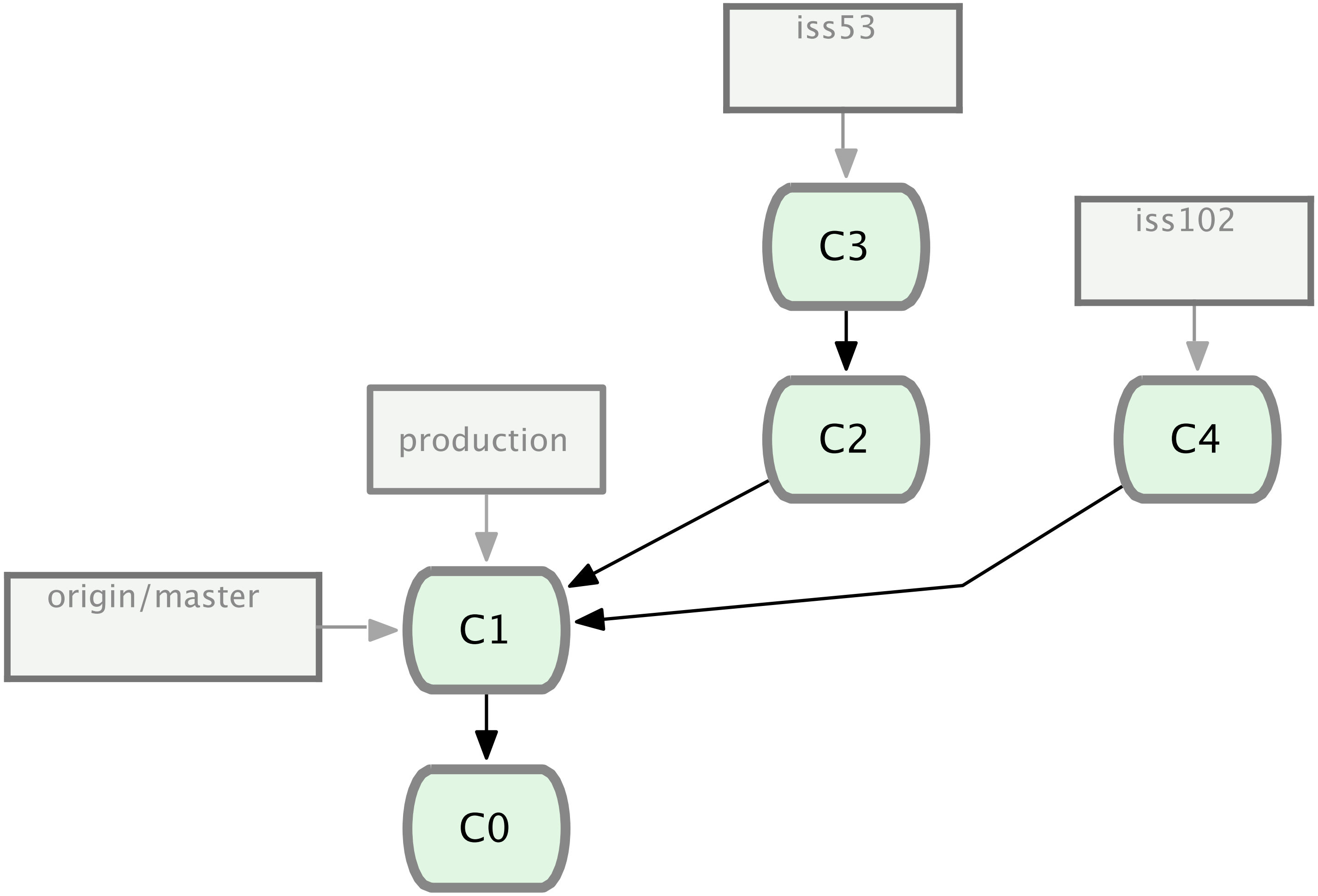
git commit



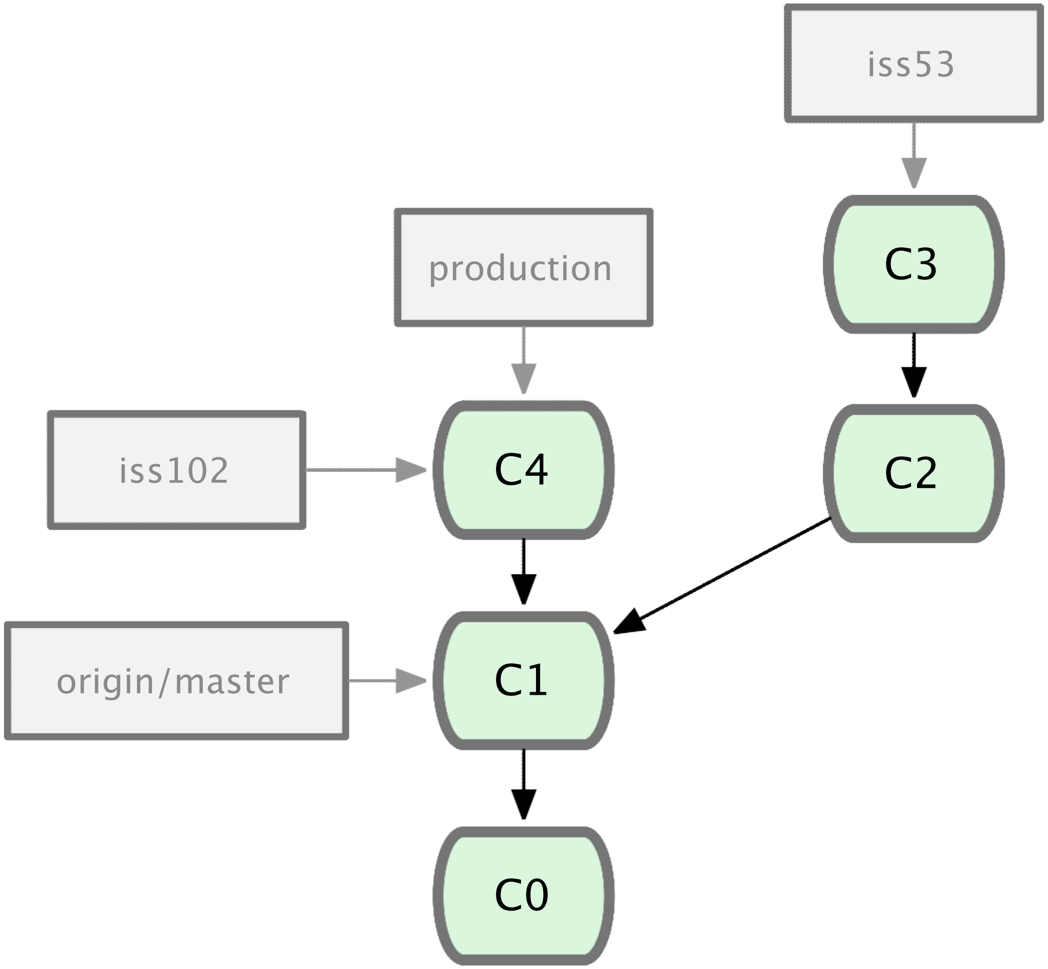
git commit



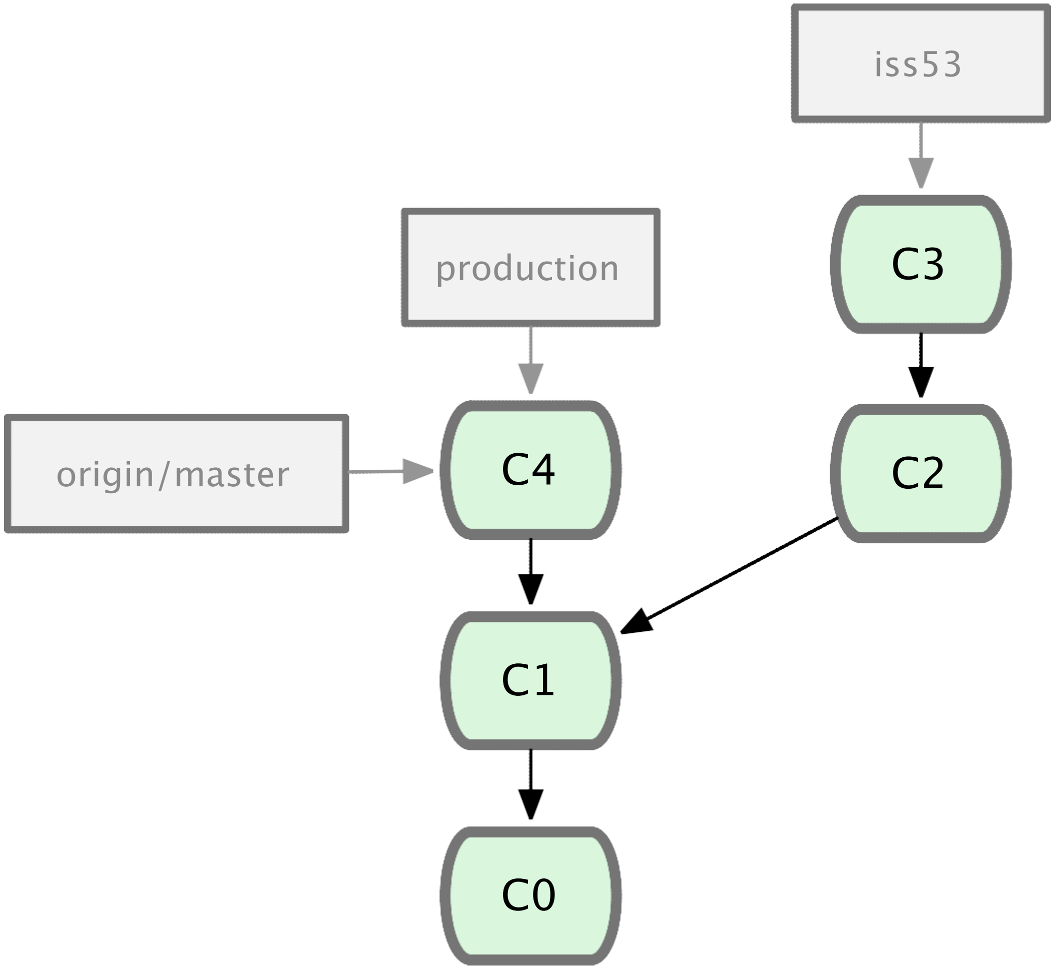
```
git checkout production
git checkout -b iss102
```



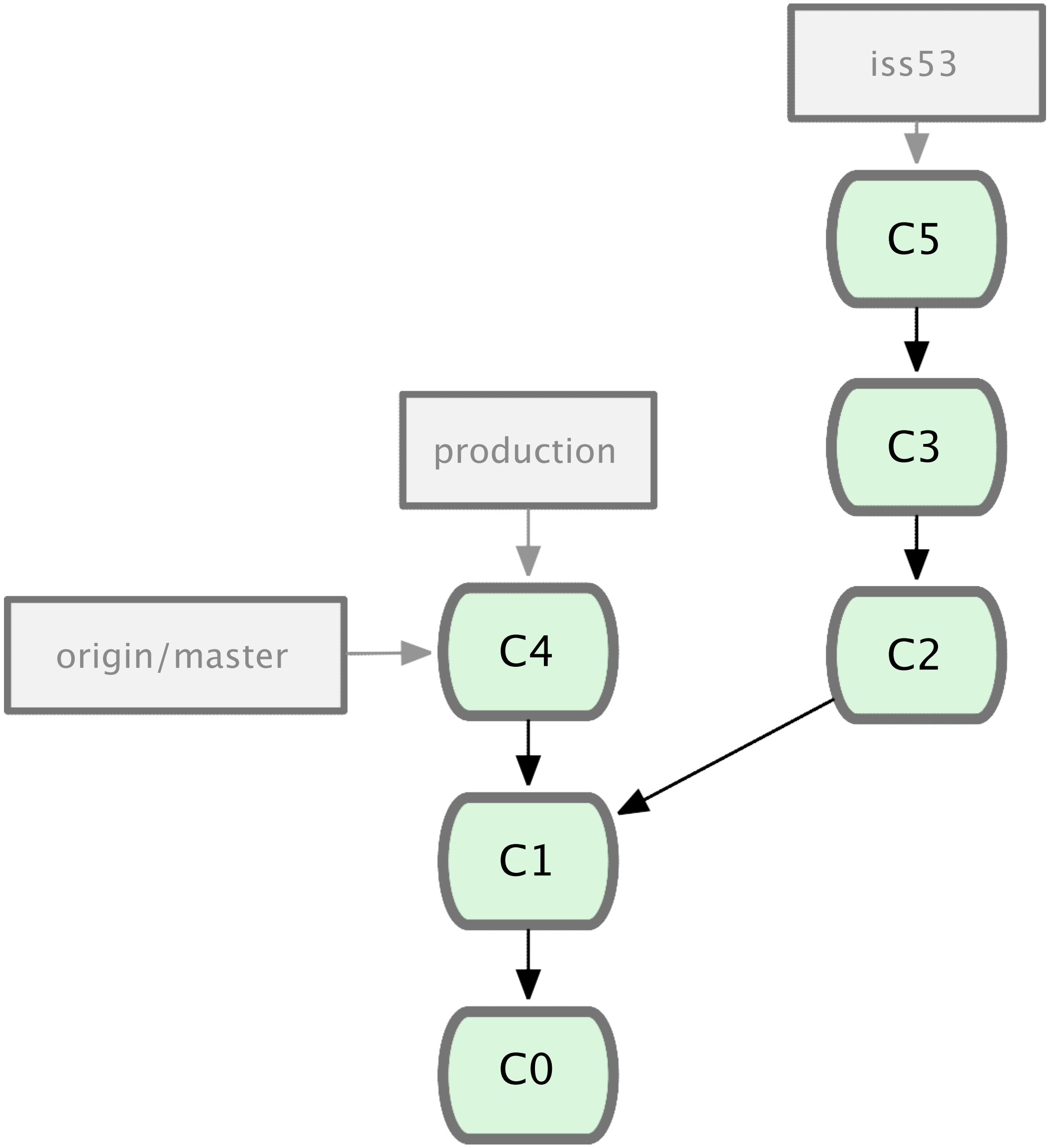
git commit



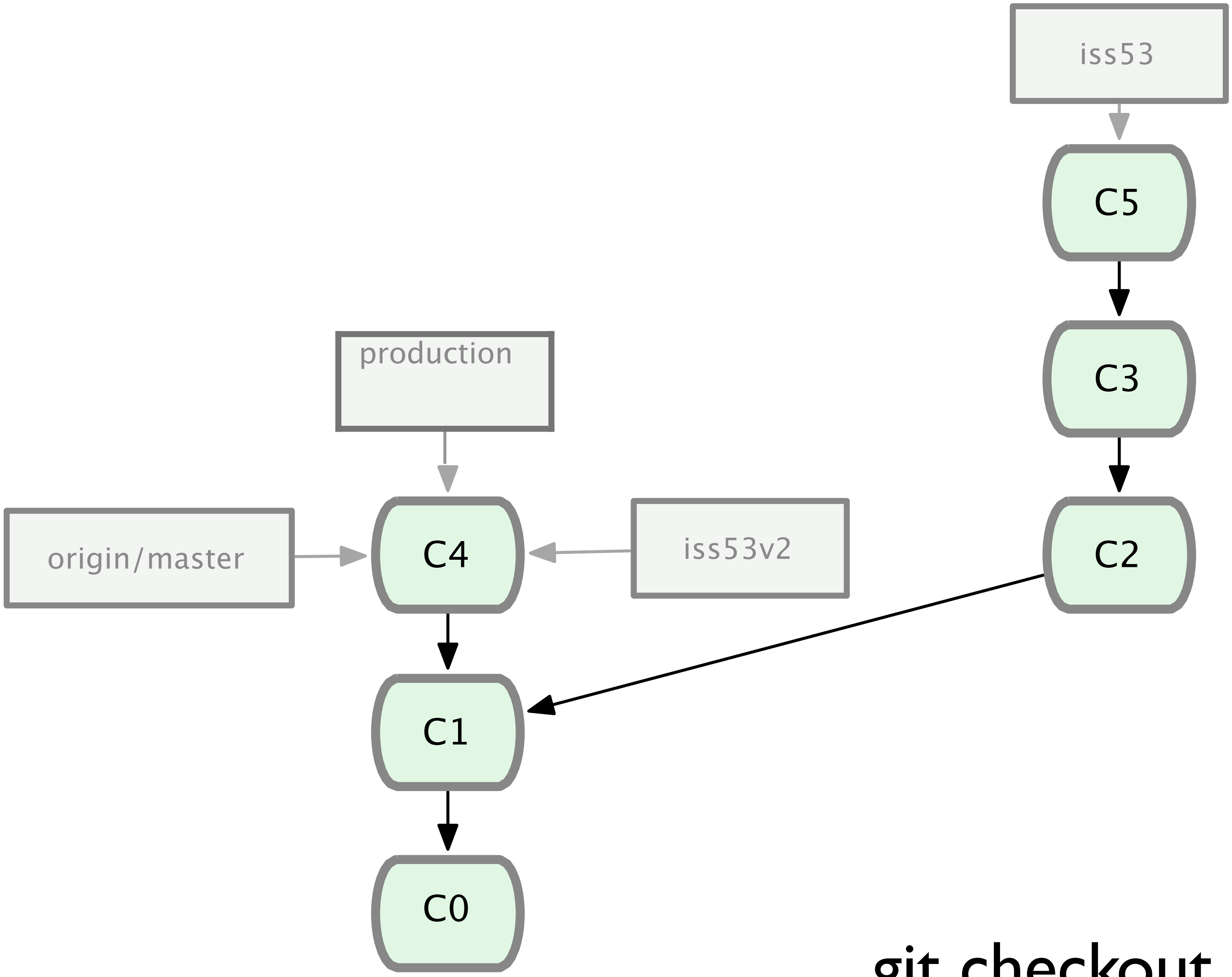
```
git checkout production
git merge iss102
```



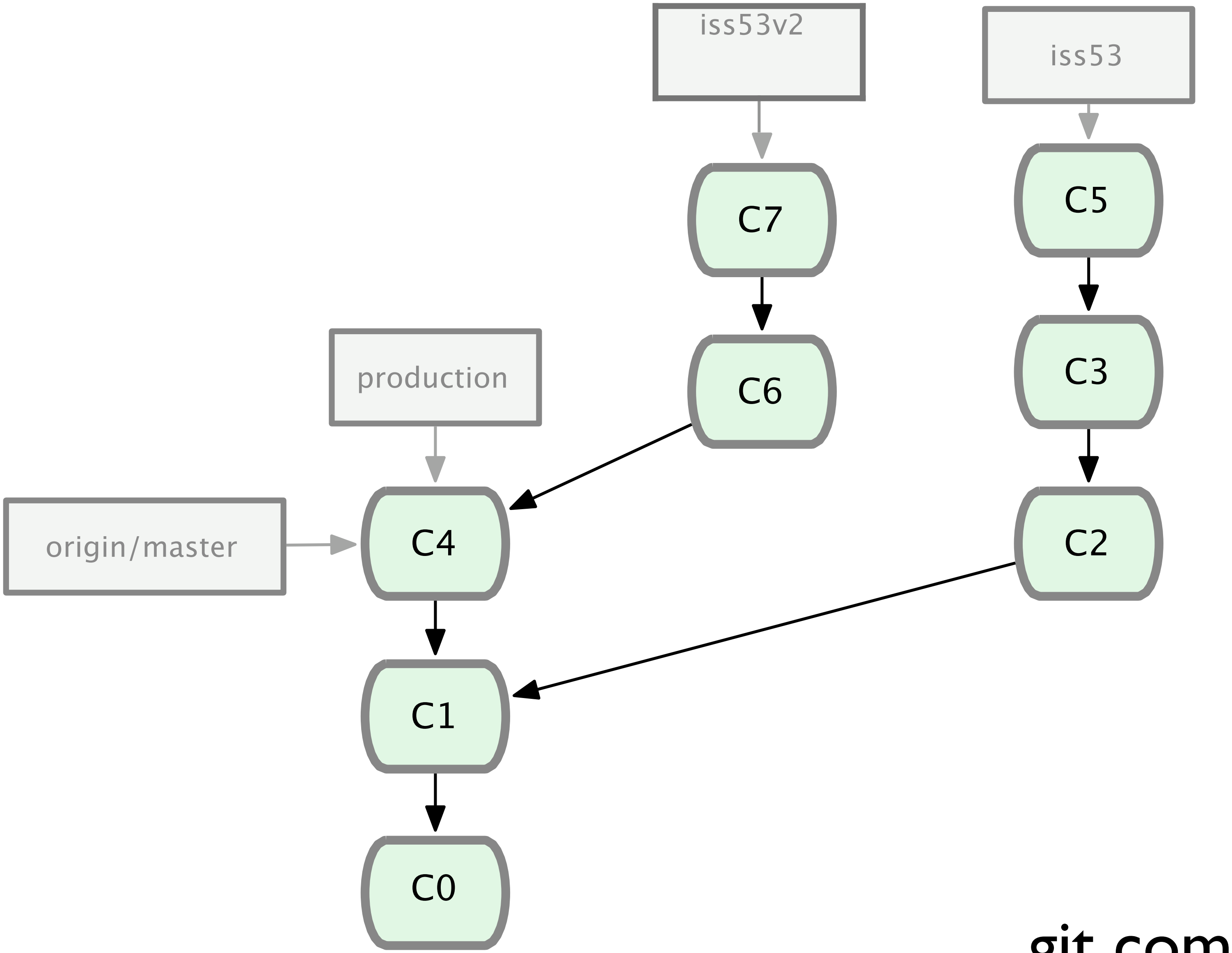
git push



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

Hydro research group remote options

1. Lab network:
   - You can define a remote as a path to a git repository on the network
   - For example git clone /nfs/thermal/usr1/jhamman/RASM/VIC/model_new/VIC/.git
   - Only accessible from inside the hydro network

2. Hydra:
   - There is a git server setup on hydra.
   - For example git clone jhamman@hydra:/git/nijssen/testrepo.git
   - Requires password but can be shared outside the network

3. Github:
   - Web-based hosting service for use with the Git version control system
   - For example git clone git@github.com:UW-Hydro/VIC.git
   - Can control who has write permissions, but free version is publically readable.

notes

- this has been just an overview and you’ll undoubtedly need to look at other resources to make it work

  practice - start using git

- git is merely a tool, you have to think about a workflow around this to make it truly useful. Again, the git online resources have great examples

  practice - start using git

- the overhead of learning and using any version control system is time well-spent. One aspect of science is replicability. Ask yourself whether you or someone else can reproduce your results

  practice - start using git