

Intro to Git

Scott Chacon

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-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.

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

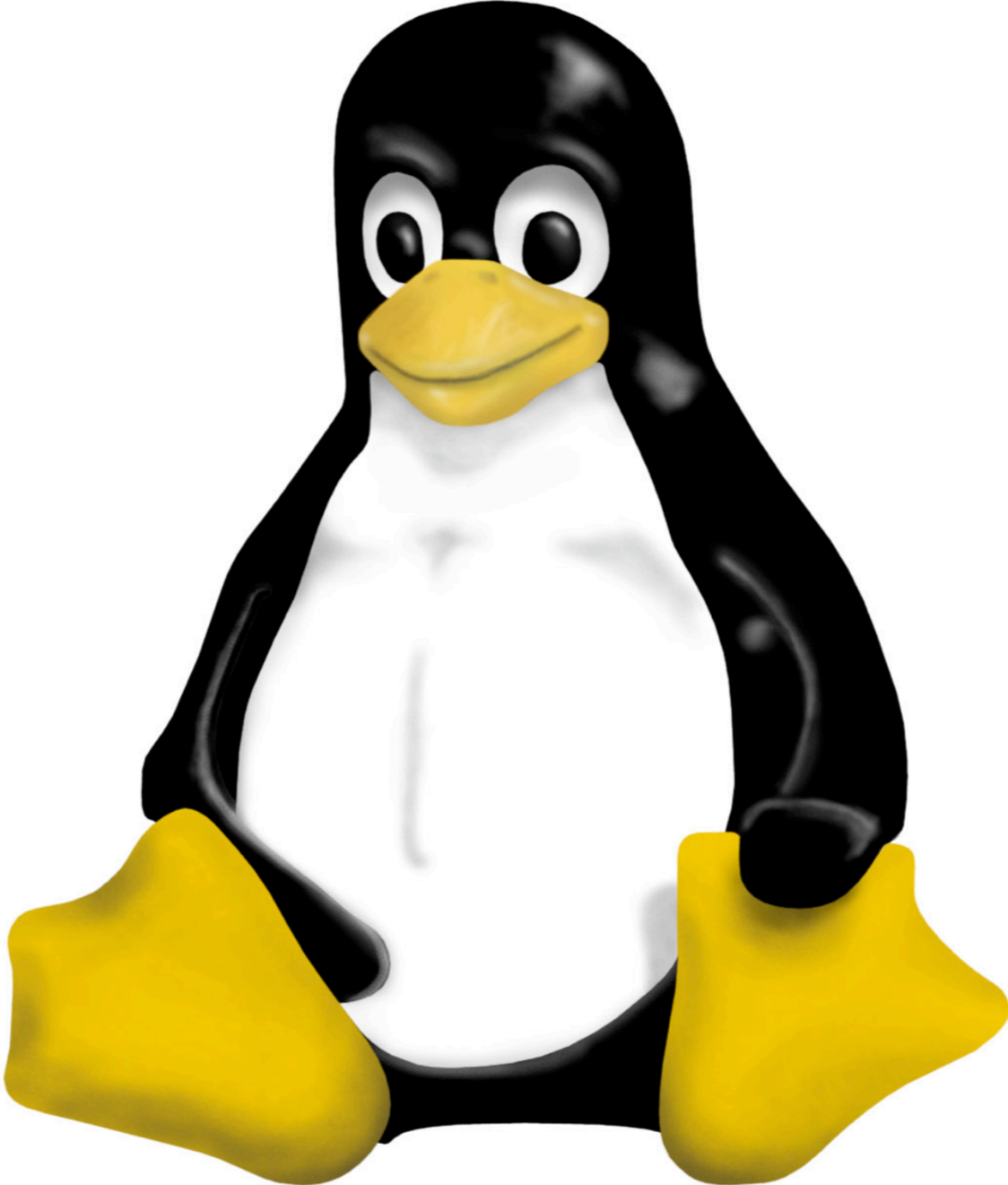
which means

everything is fast

every clone is a backup

work offline


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



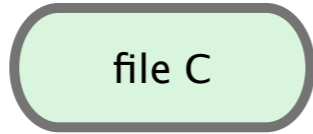
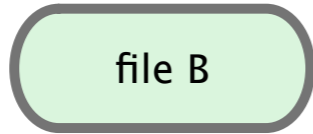
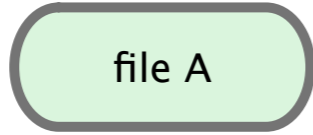
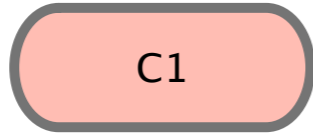
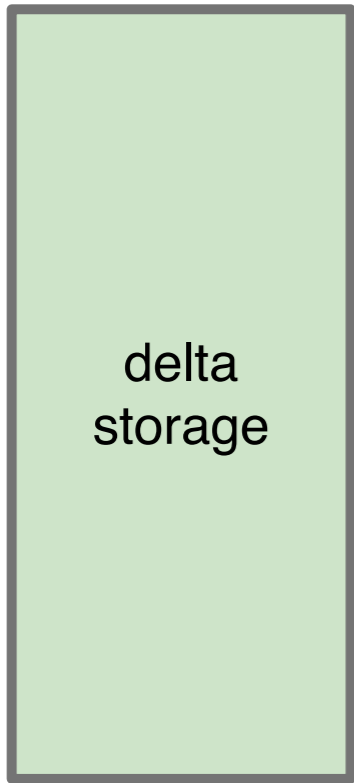
Immutable

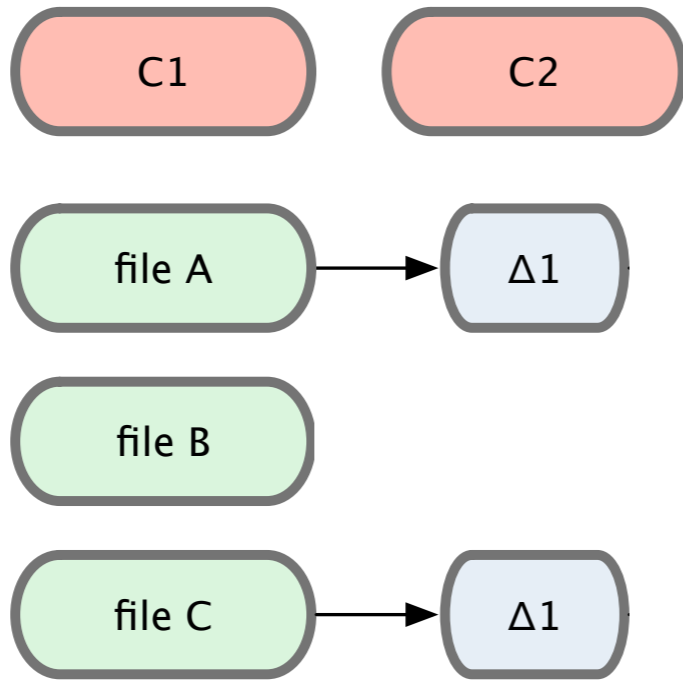
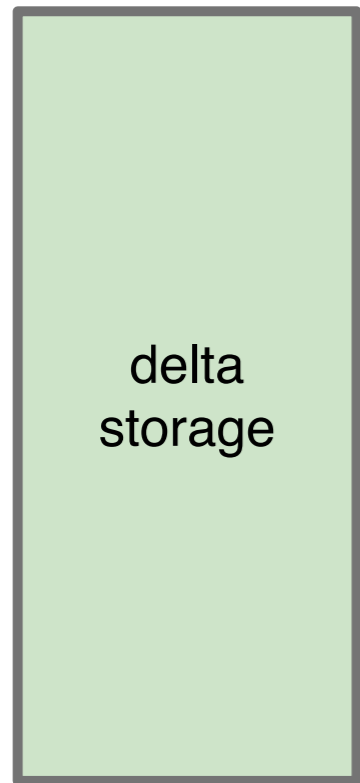
(almost) never removes data

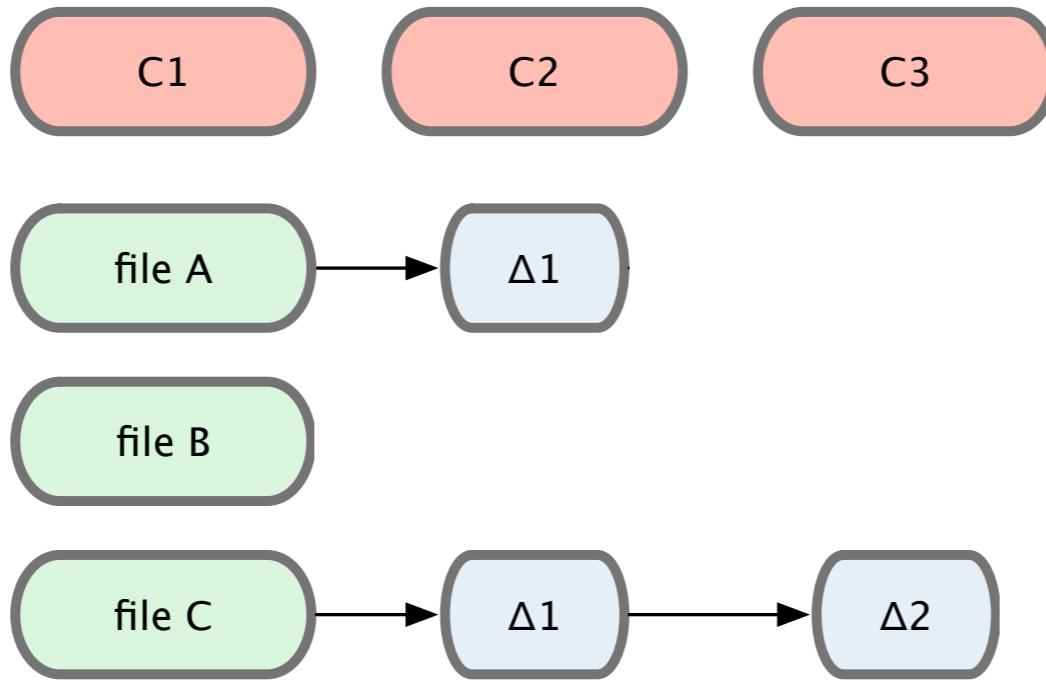
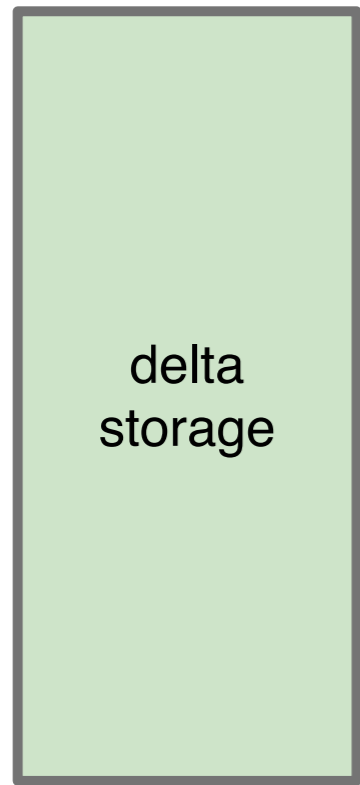
Snapshots, not Patches

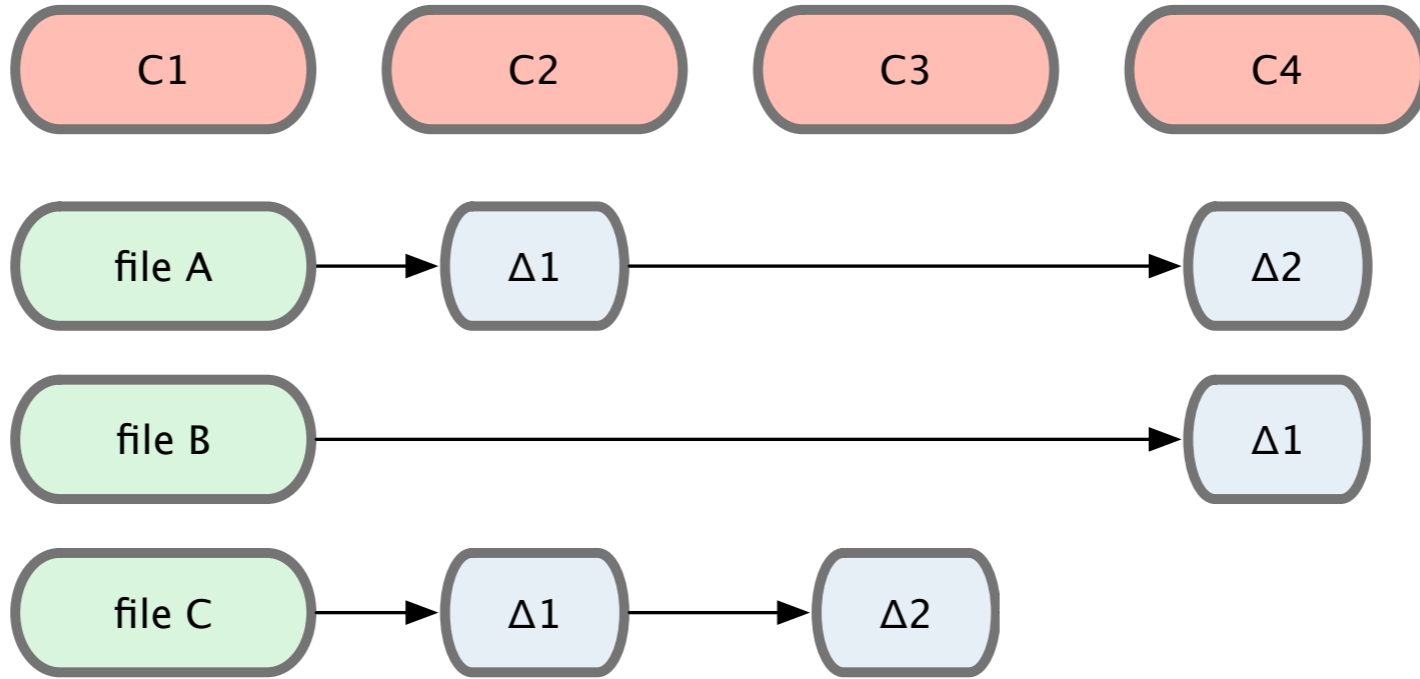
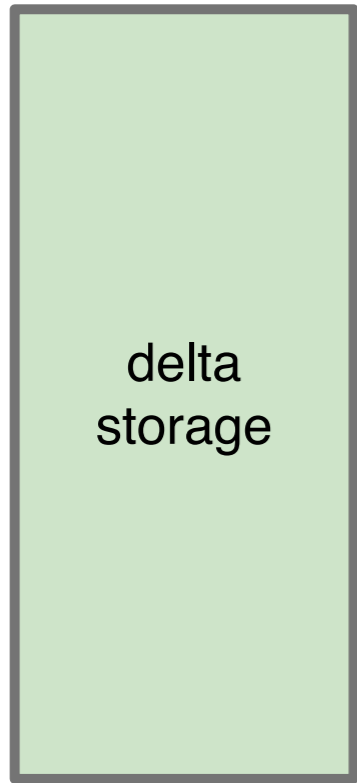


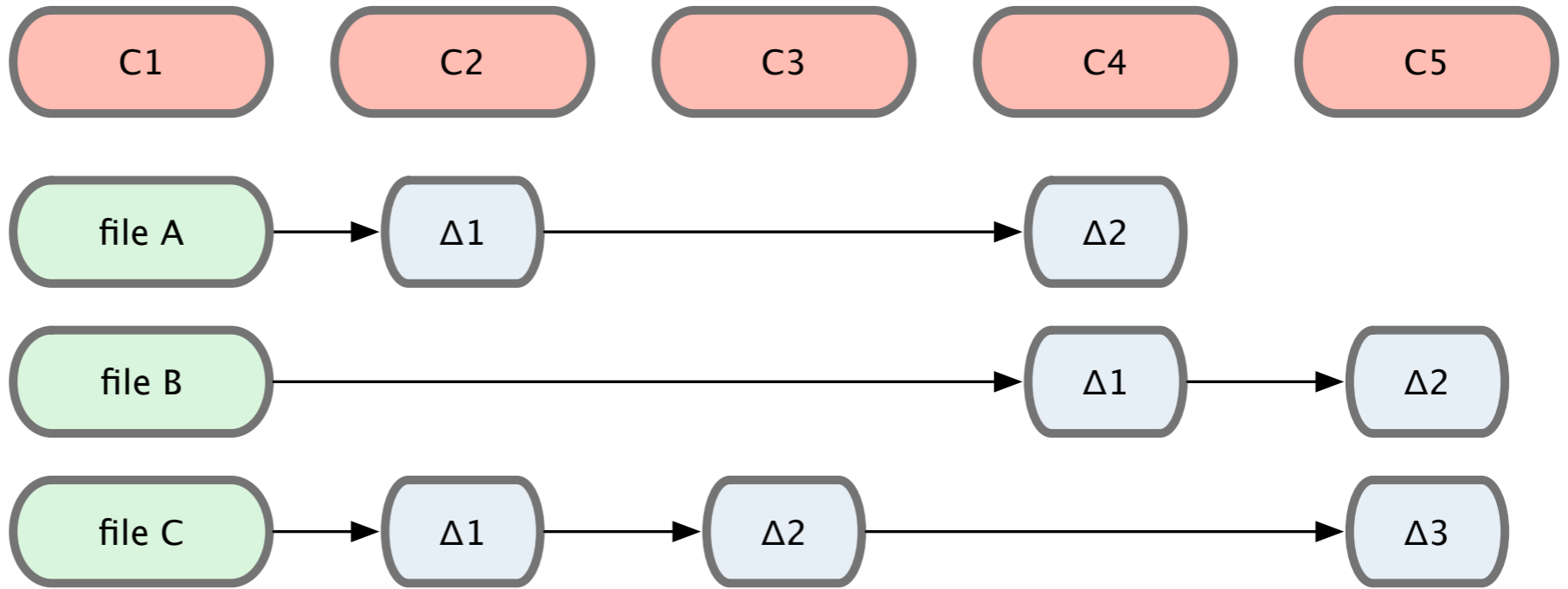
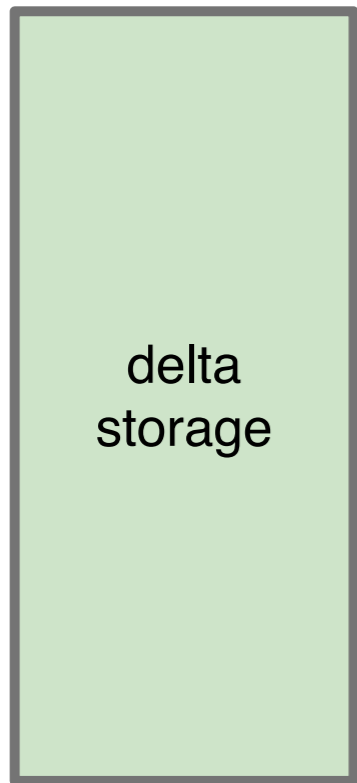
delta
storage

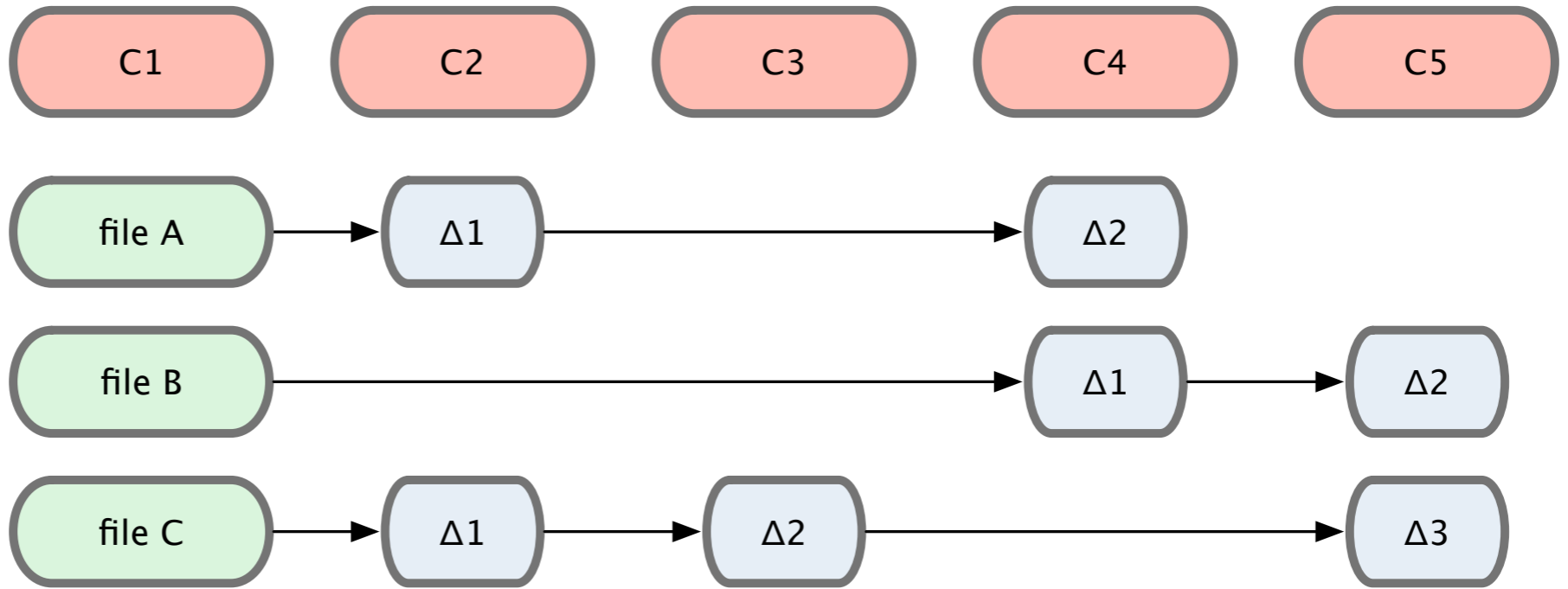
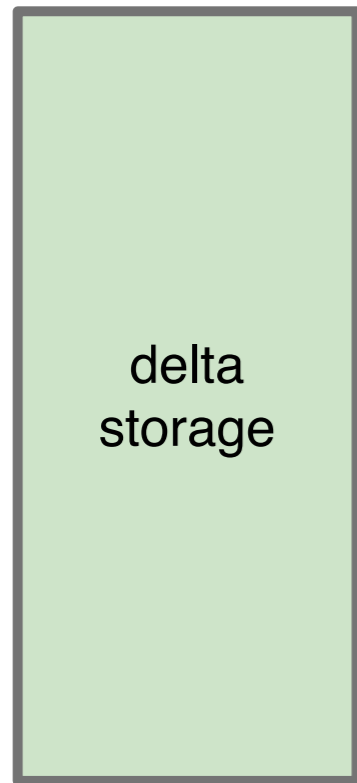


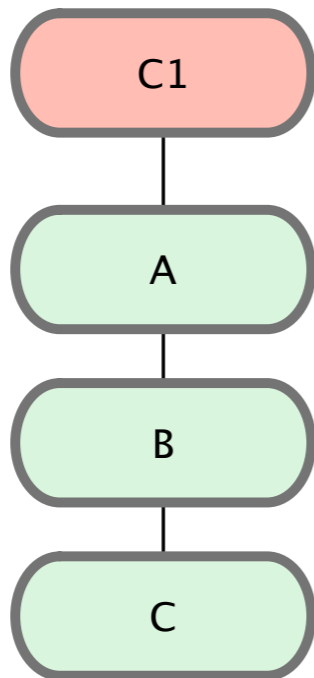
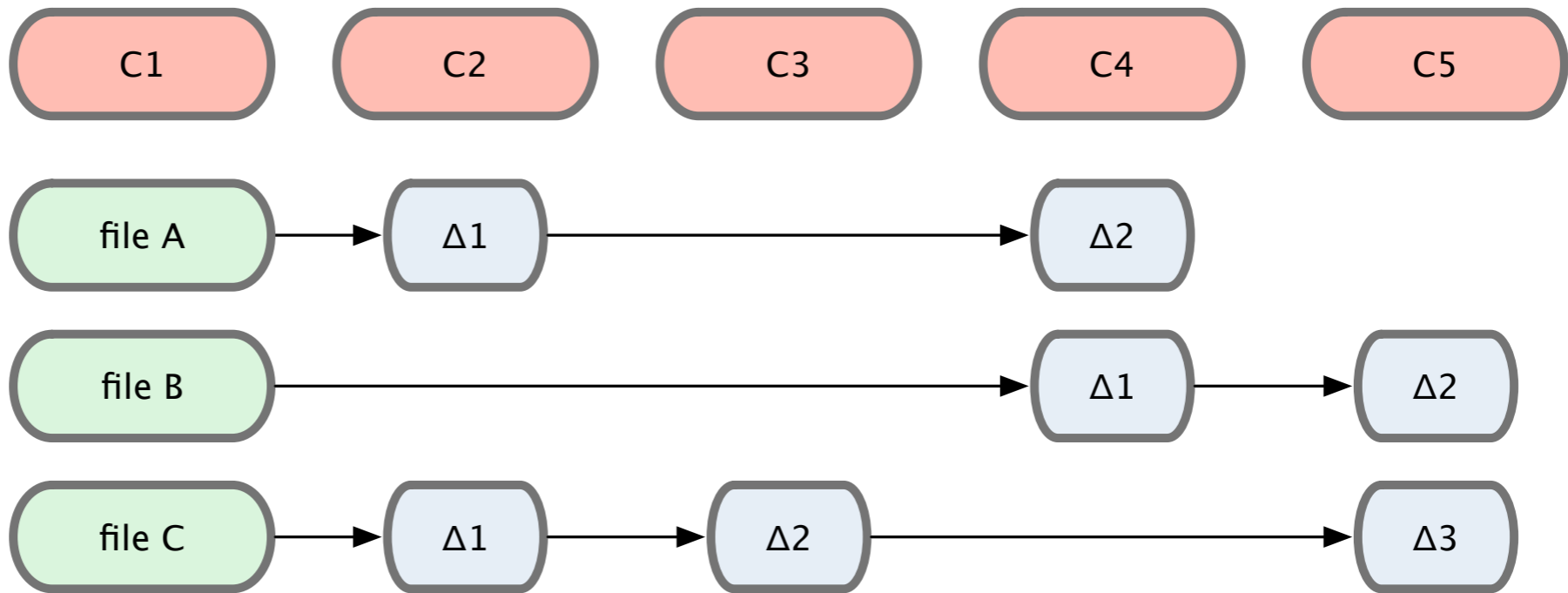
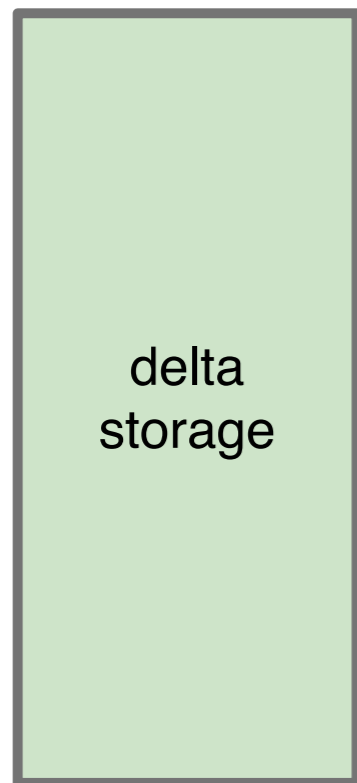


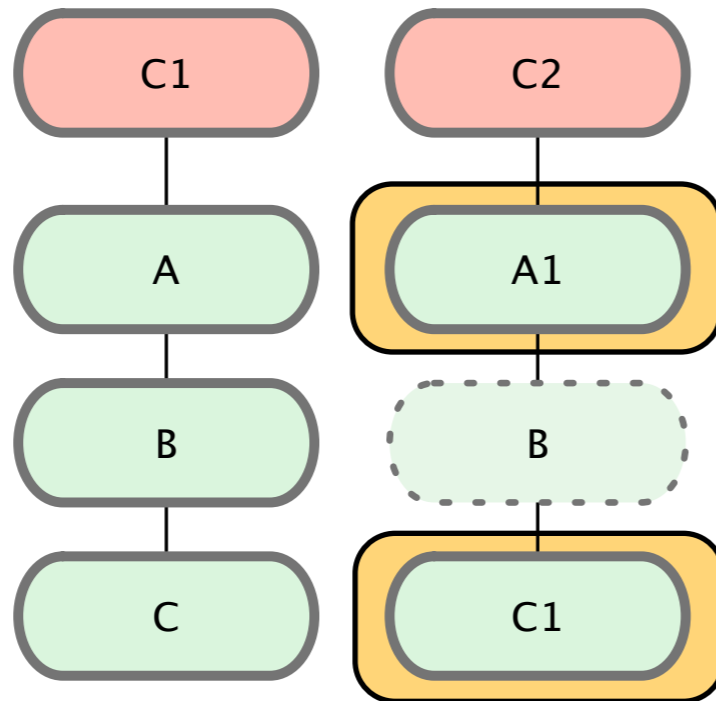
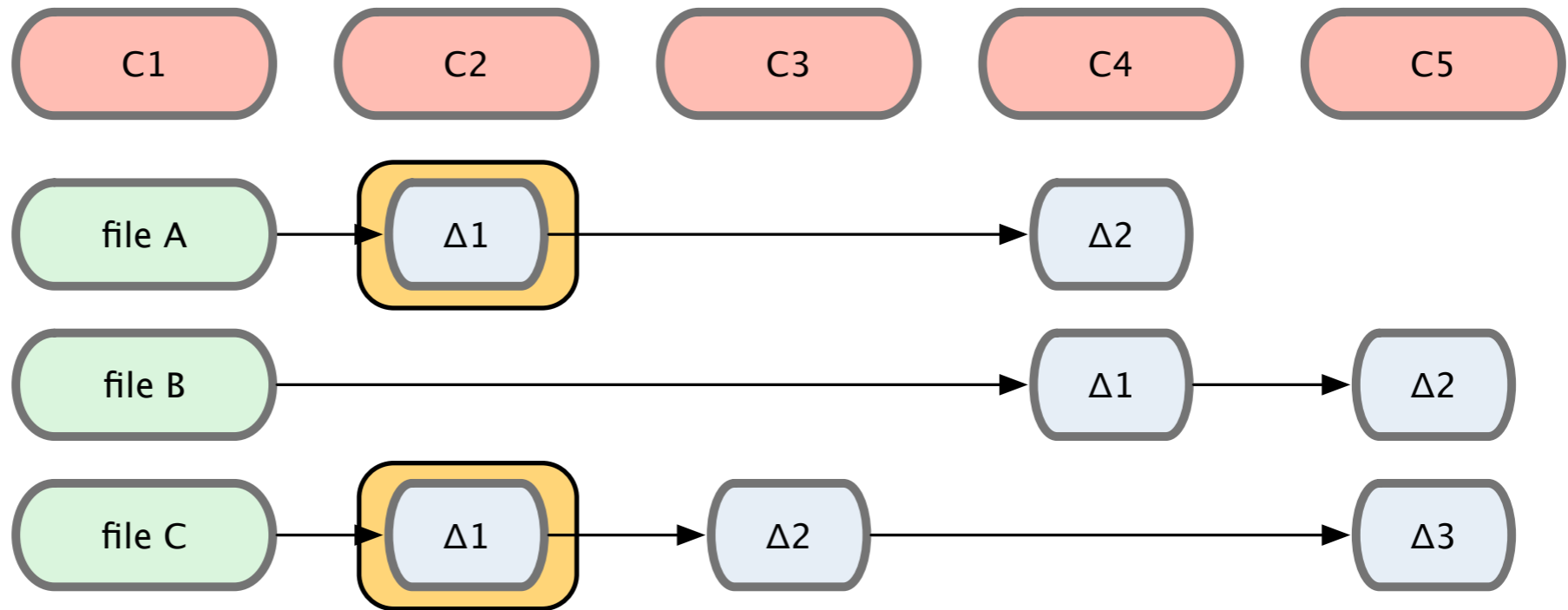
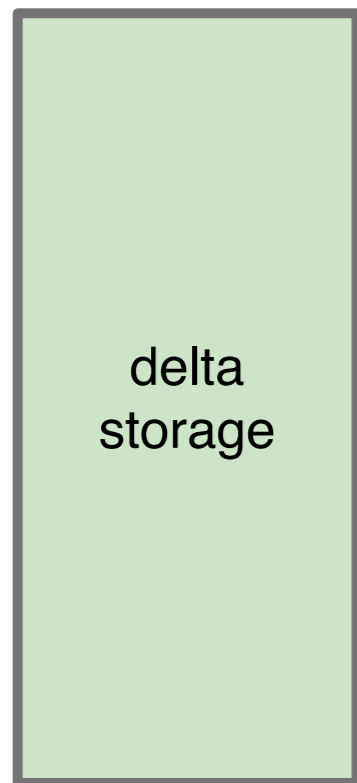


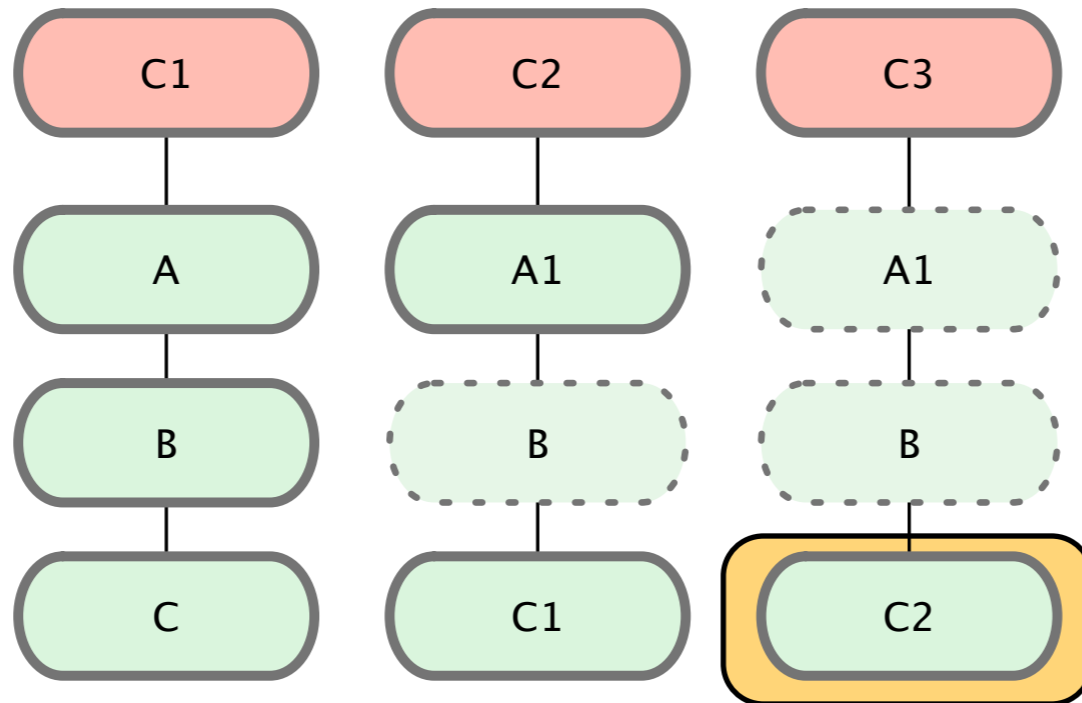
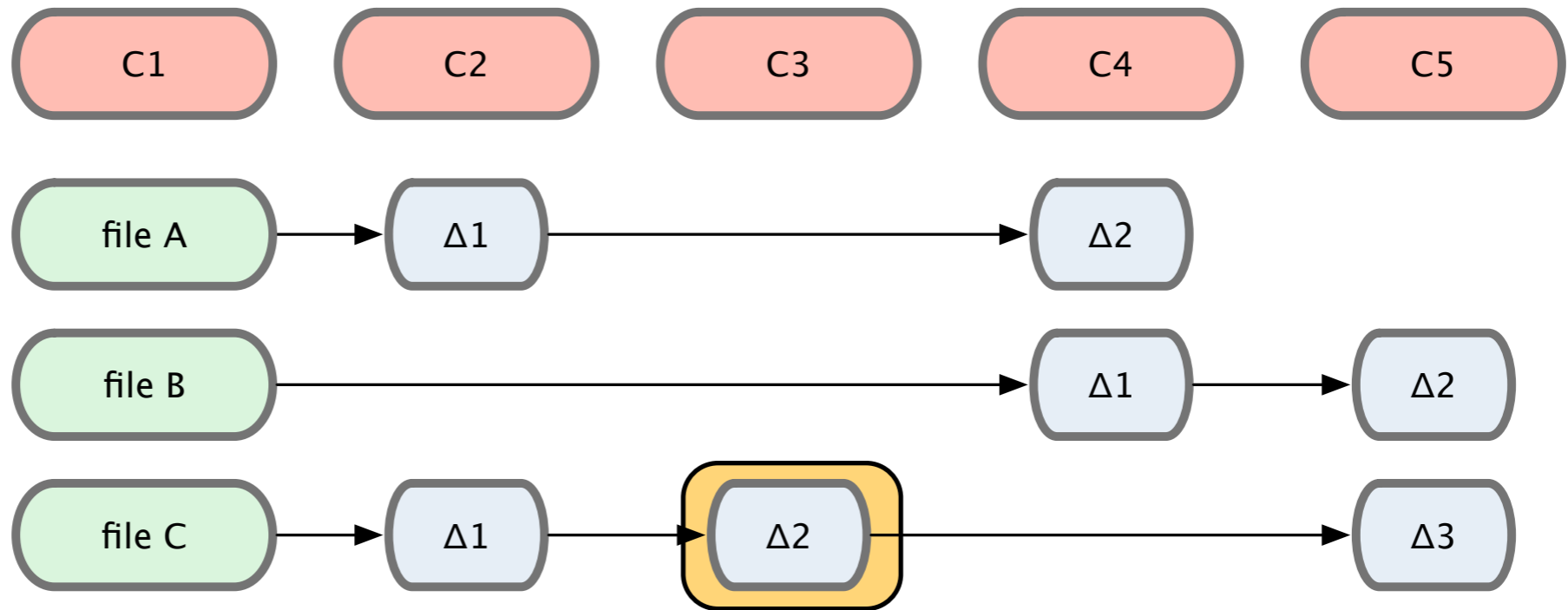
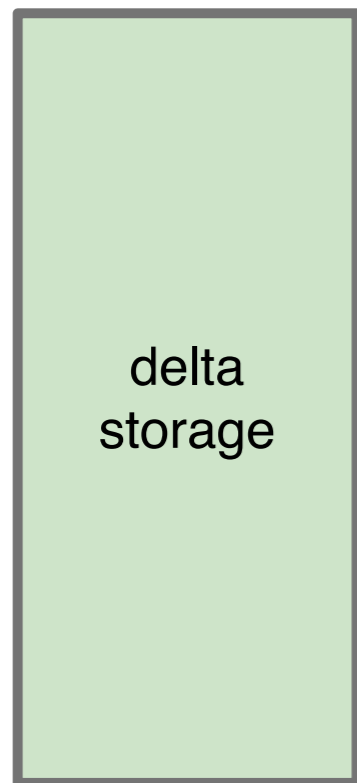


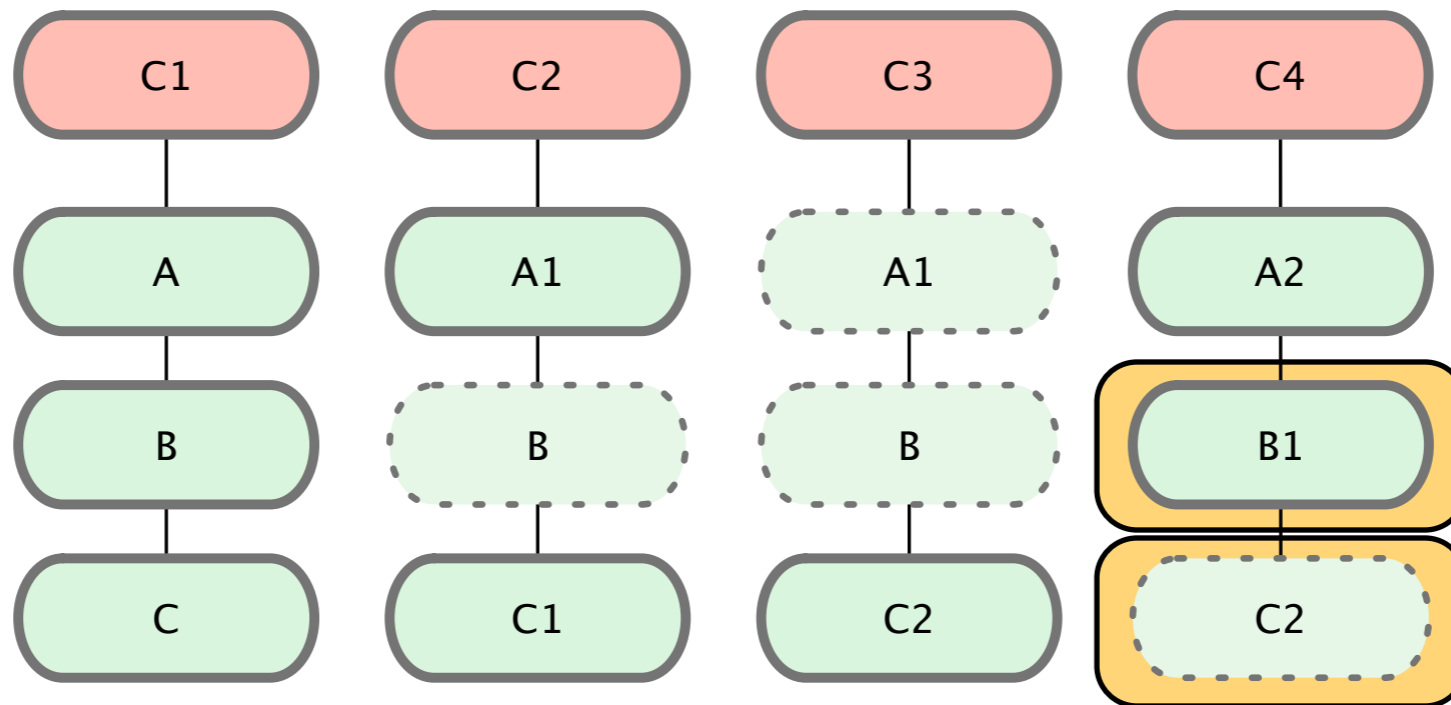
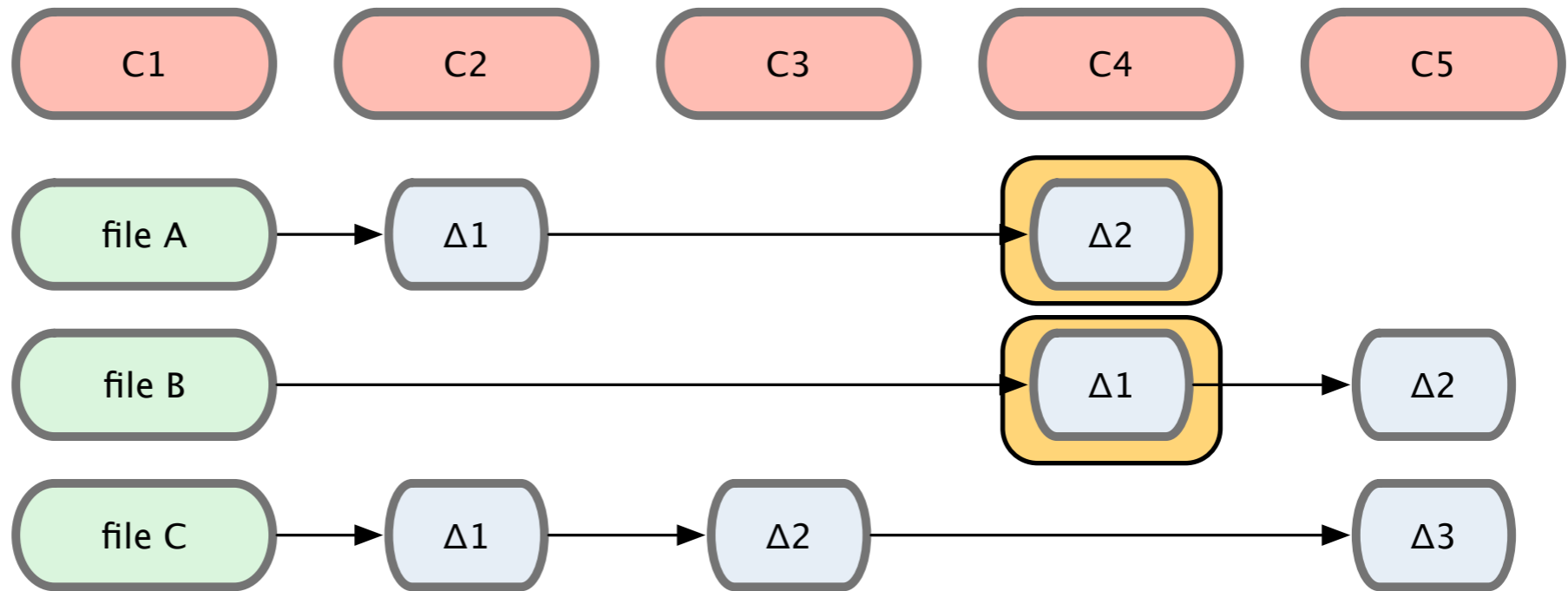
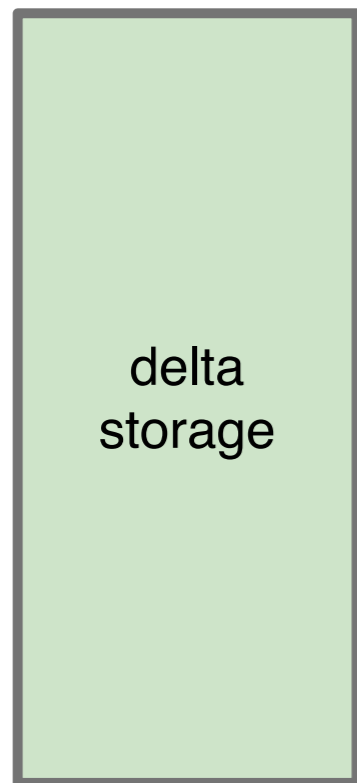


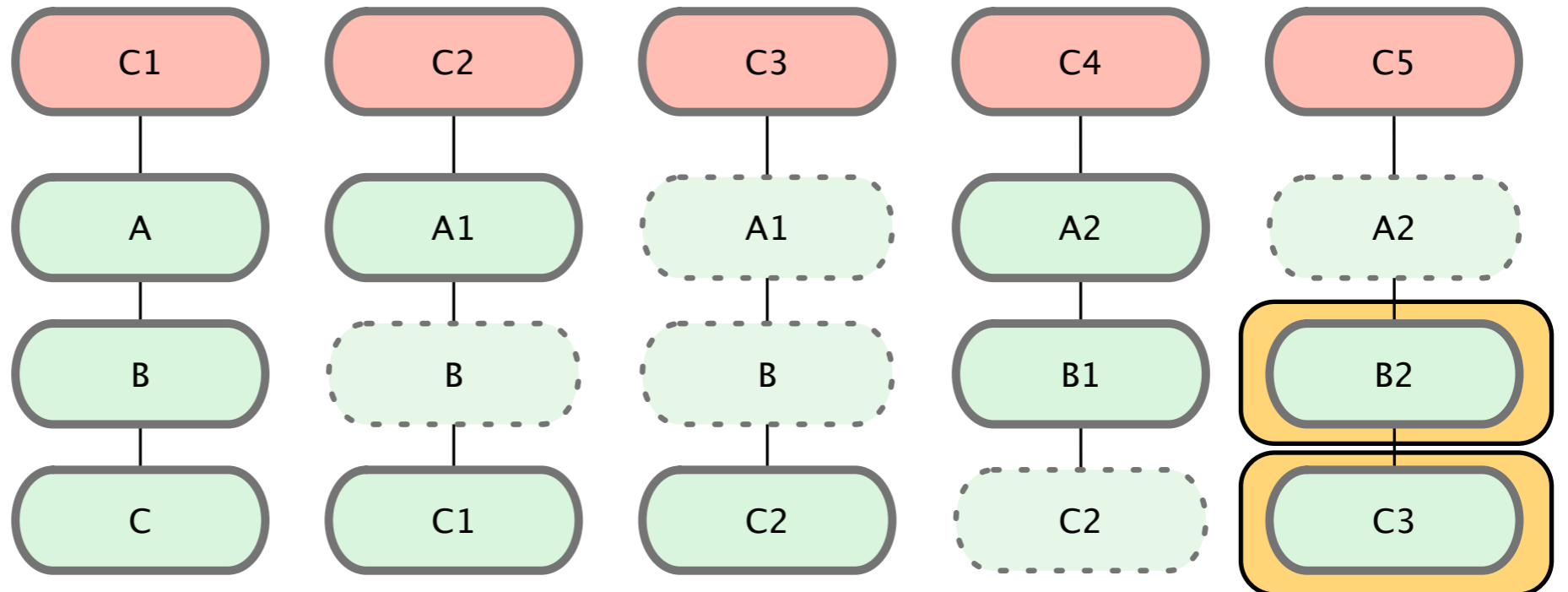
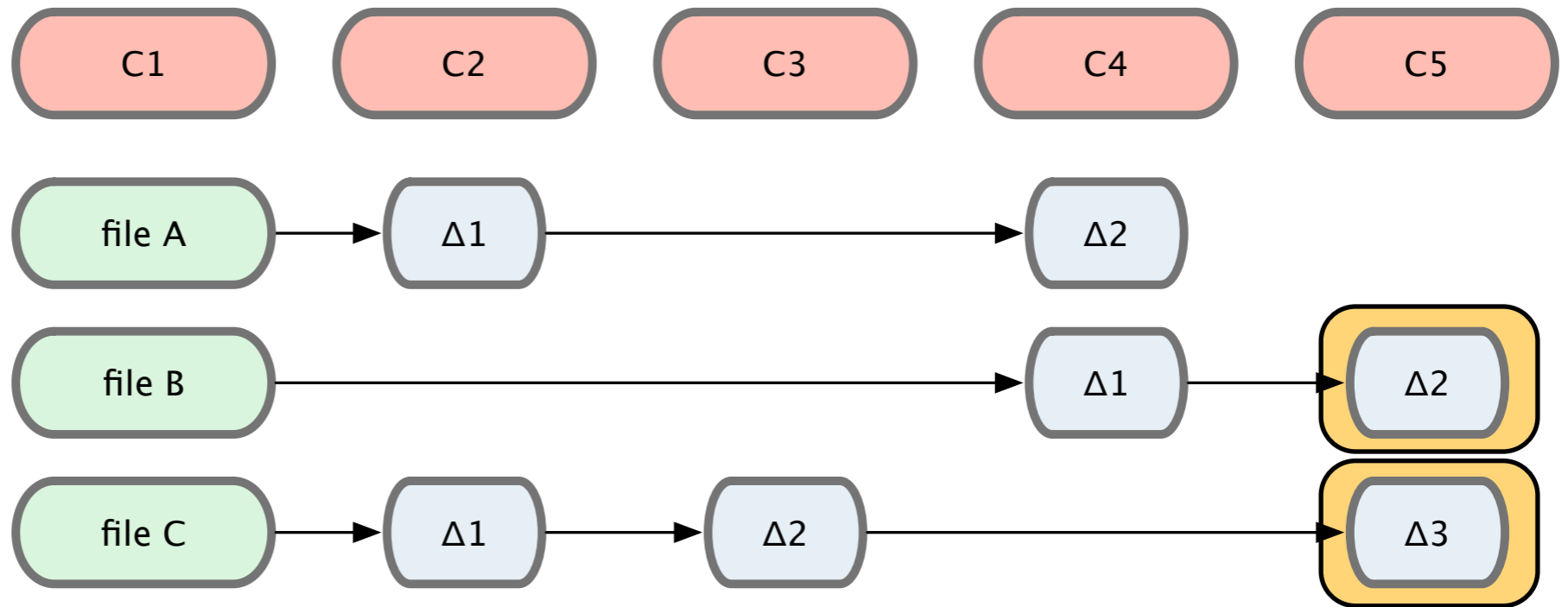
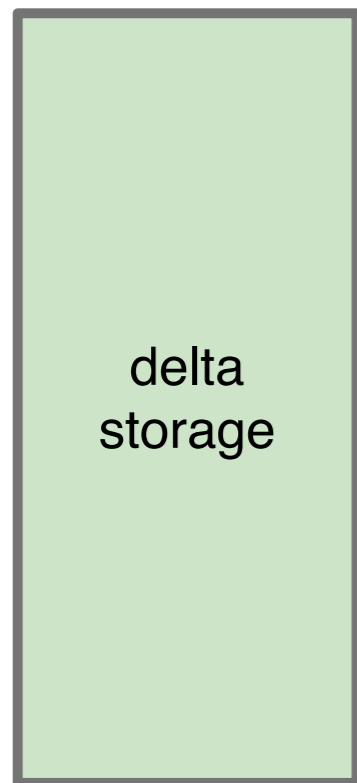


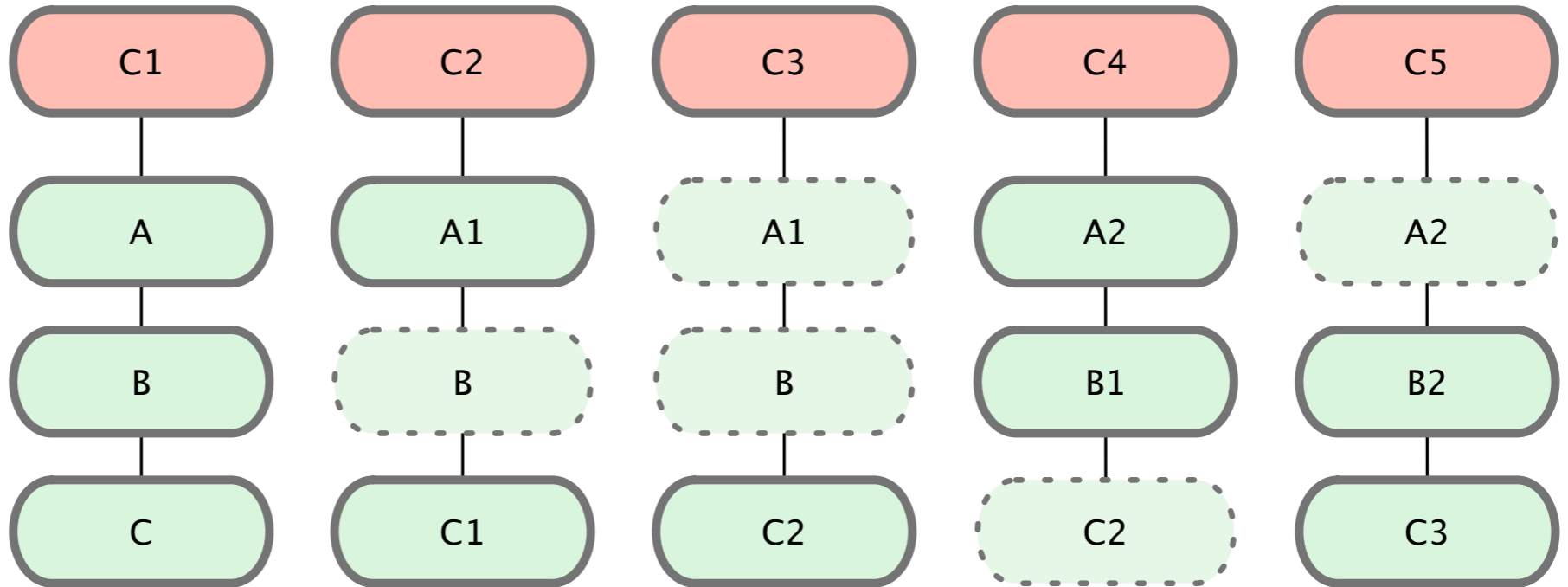
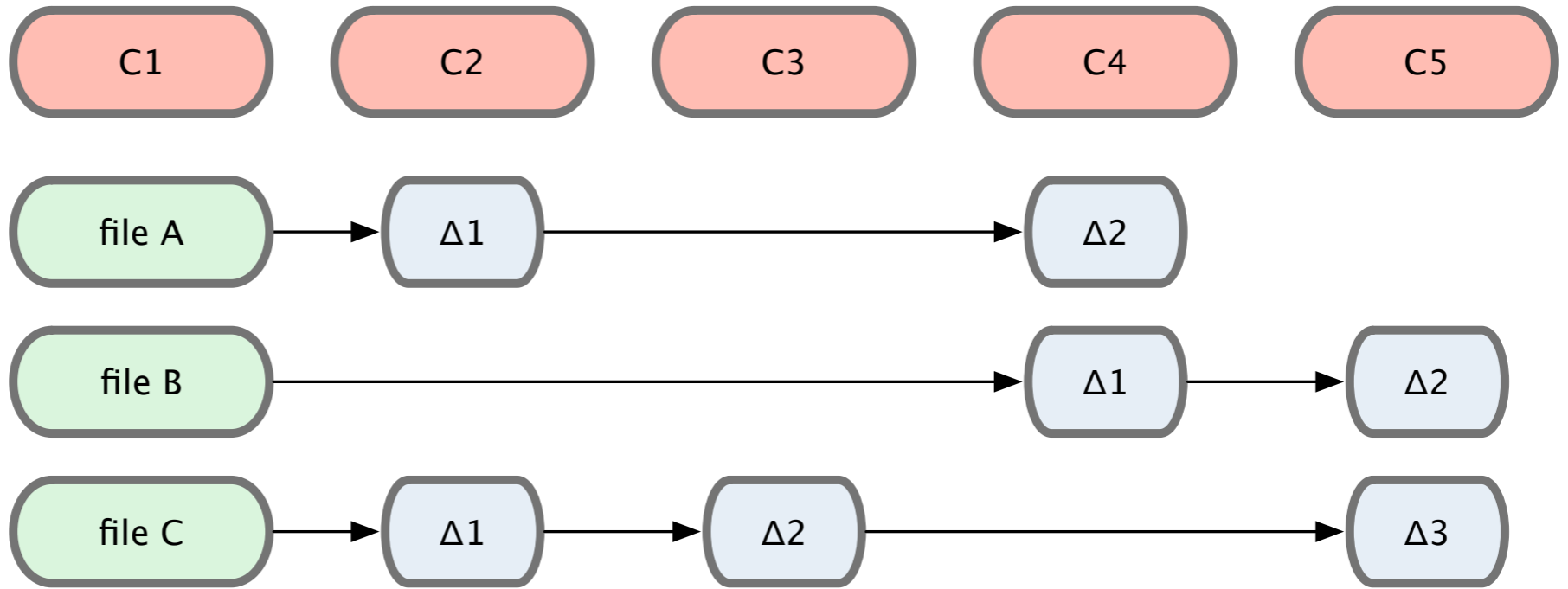
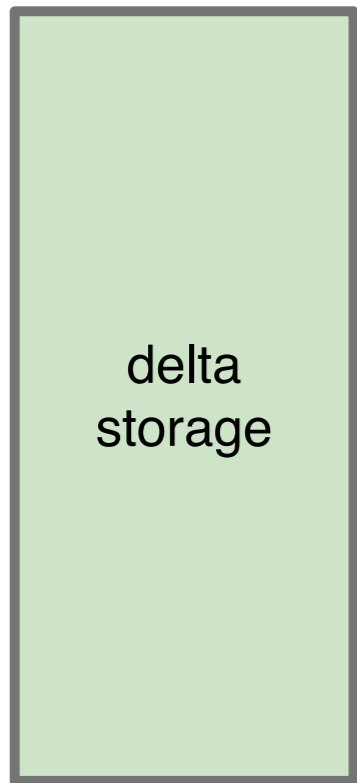


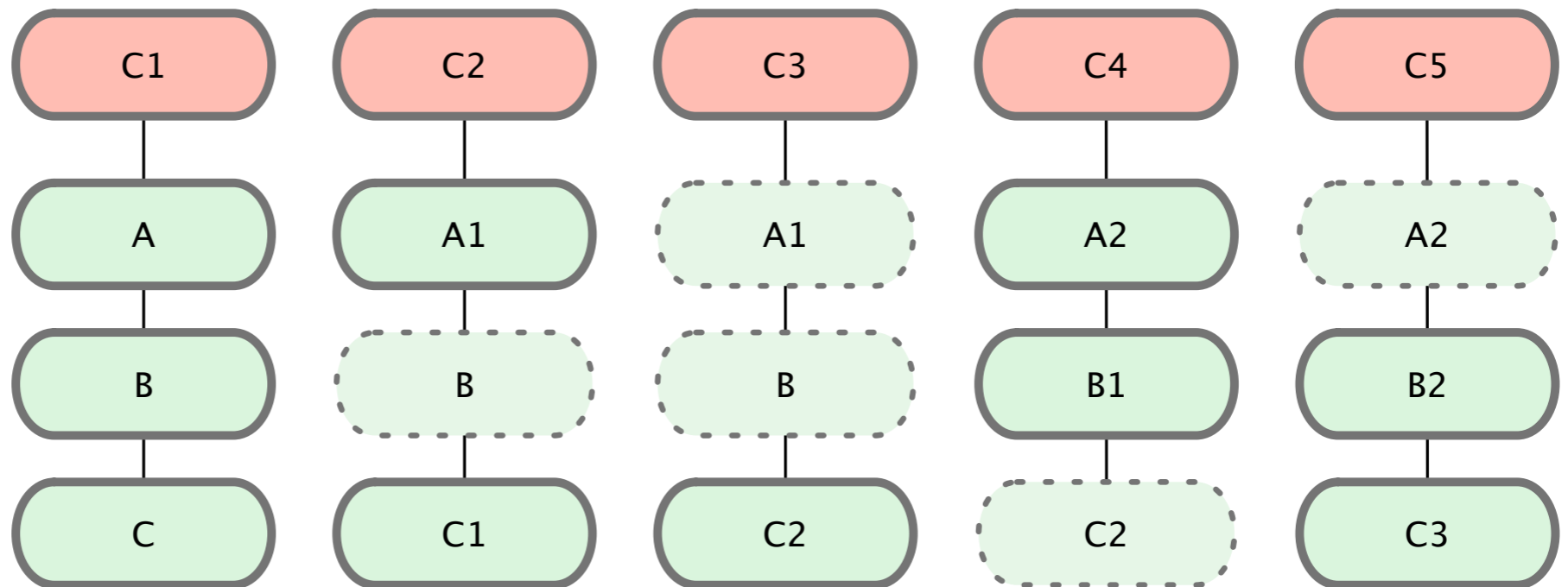
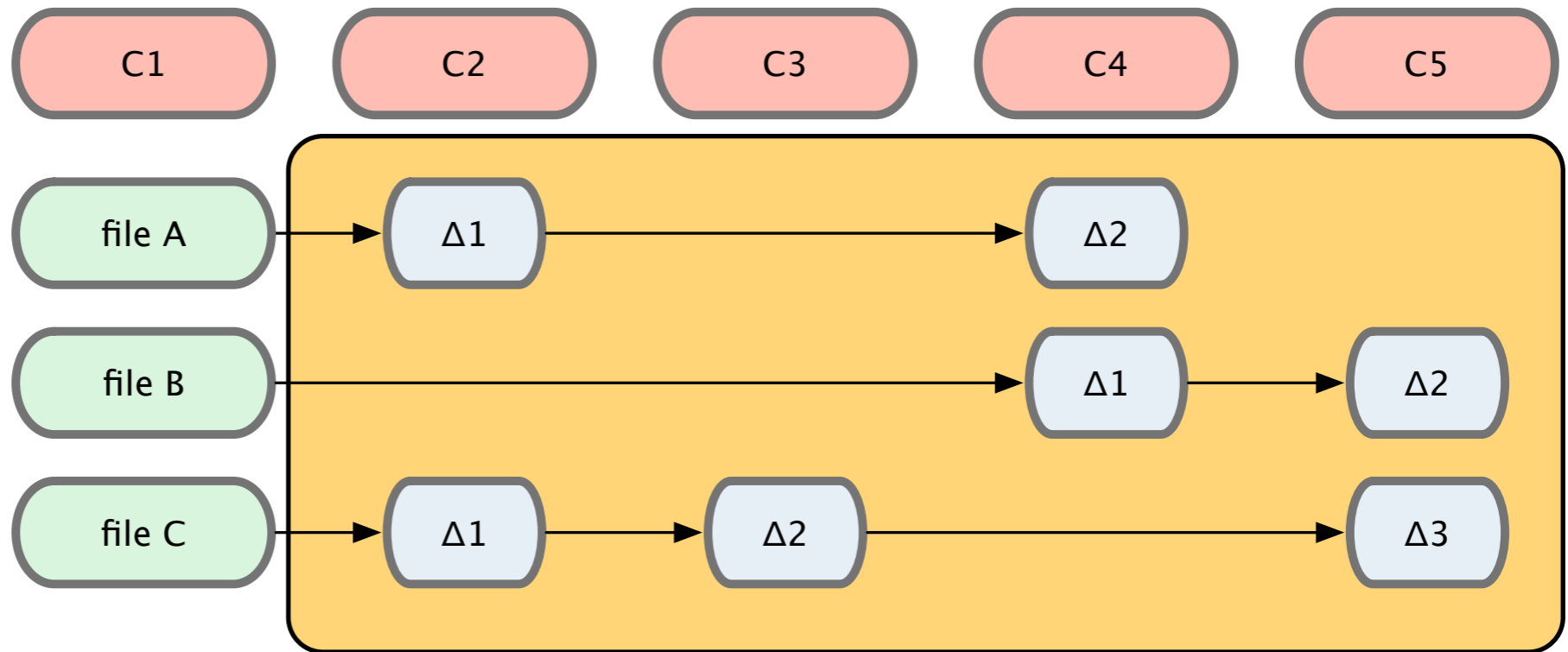
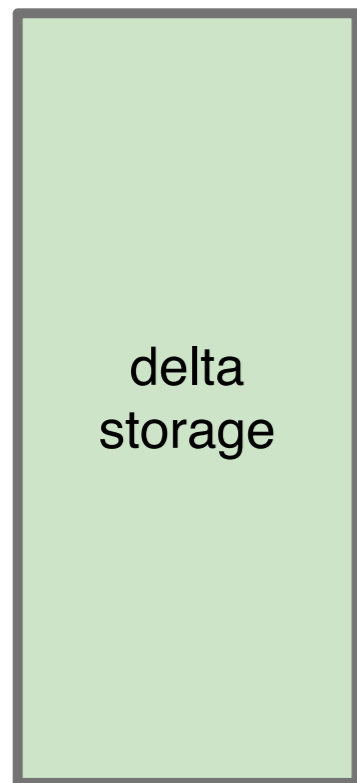


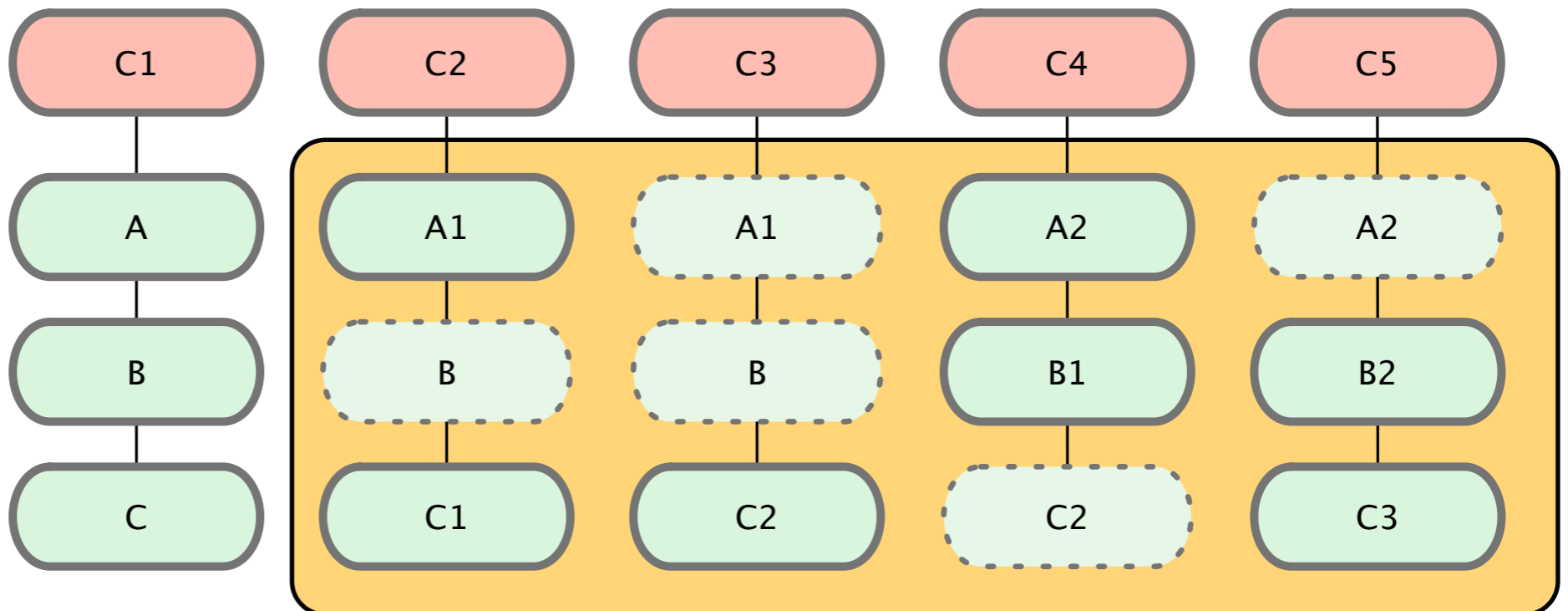
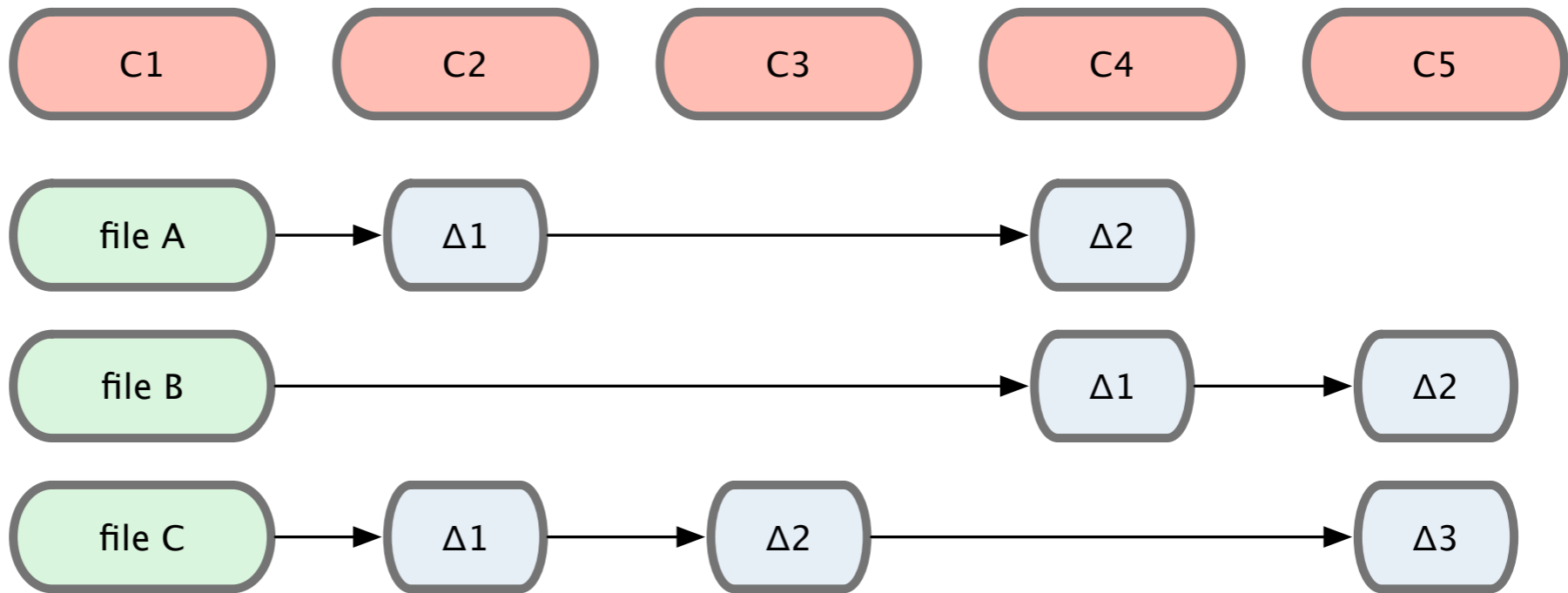
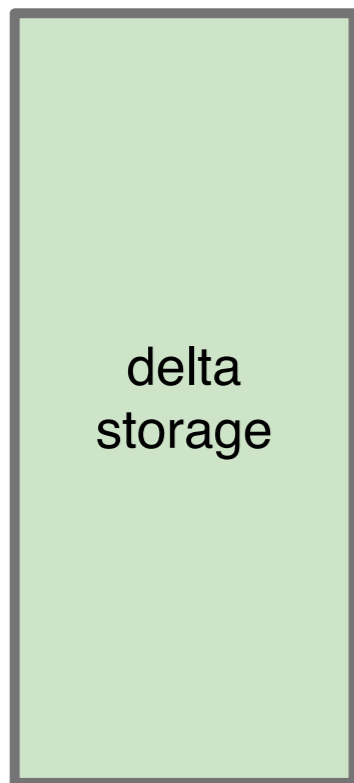












How is this Different
Than {cvs|svn|p4}?

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

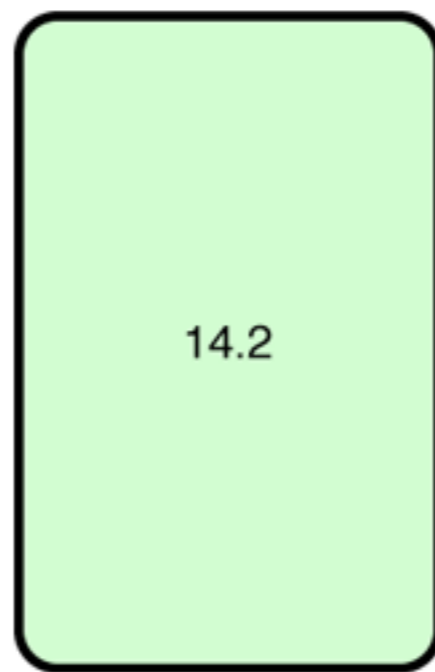
Merging branches

Obtaining any other revision of a file

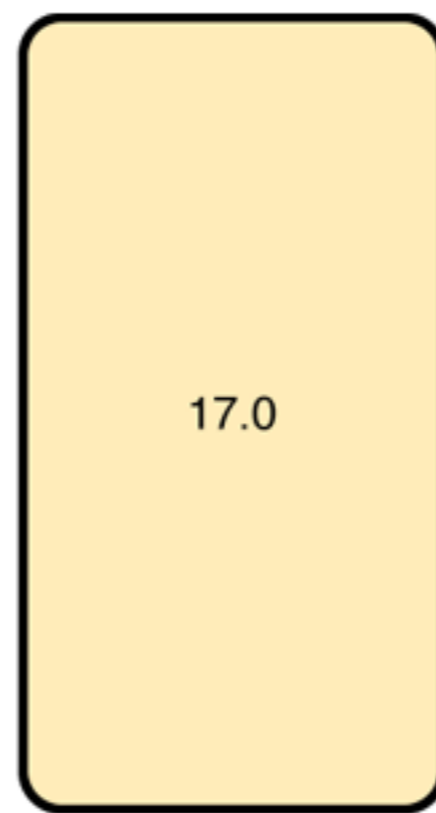
Switching branches

jQuery tests

Clone/Checkout

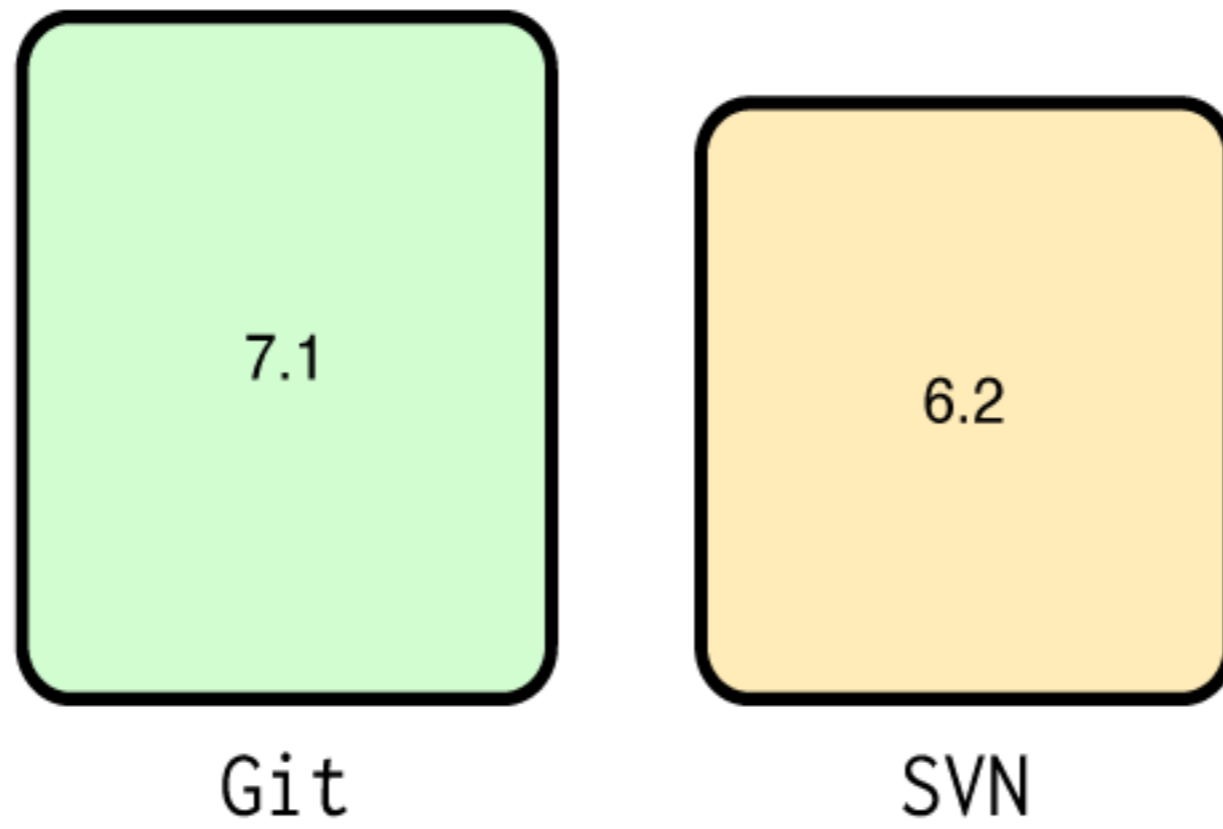


Git

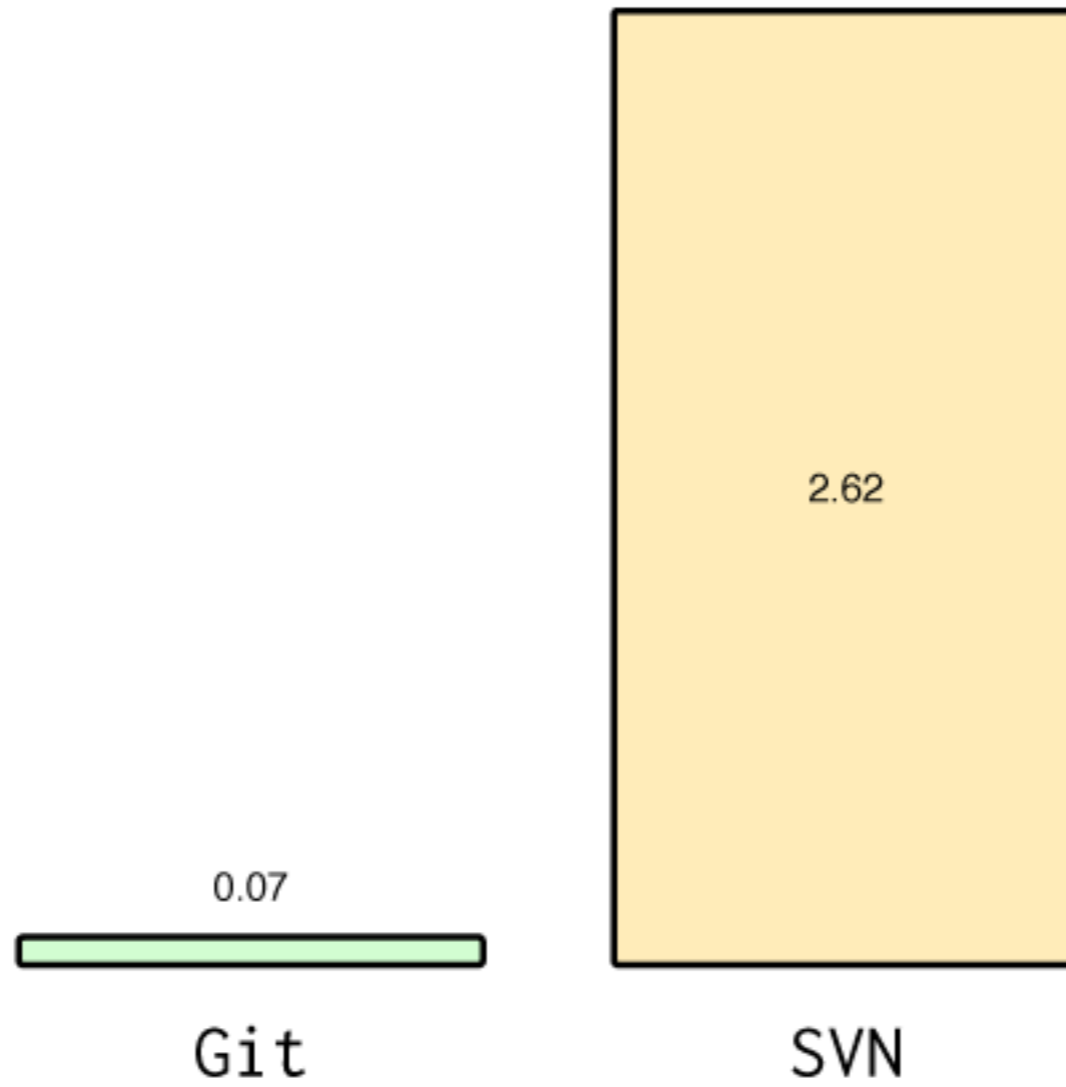


SVN

Size

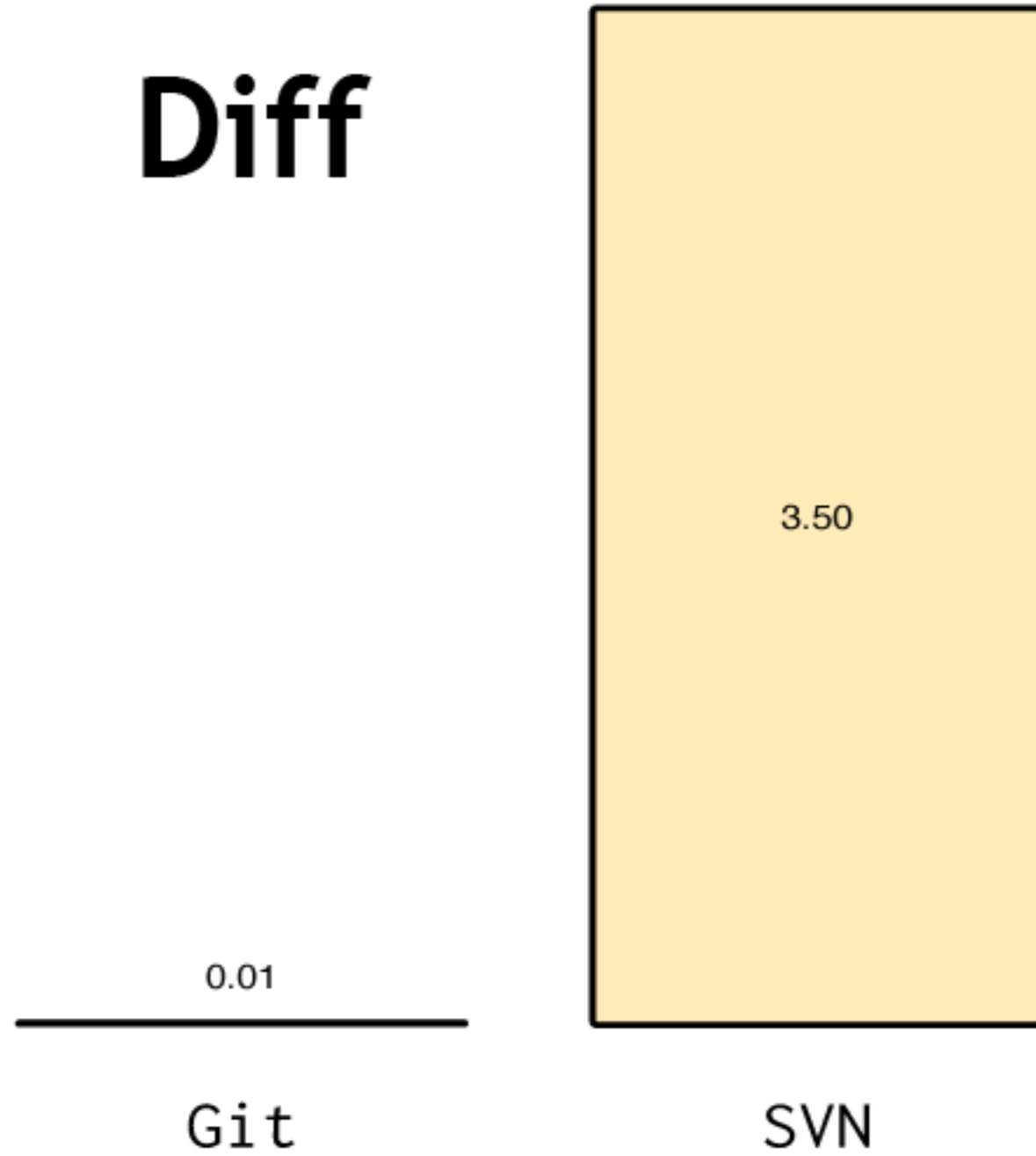


Log



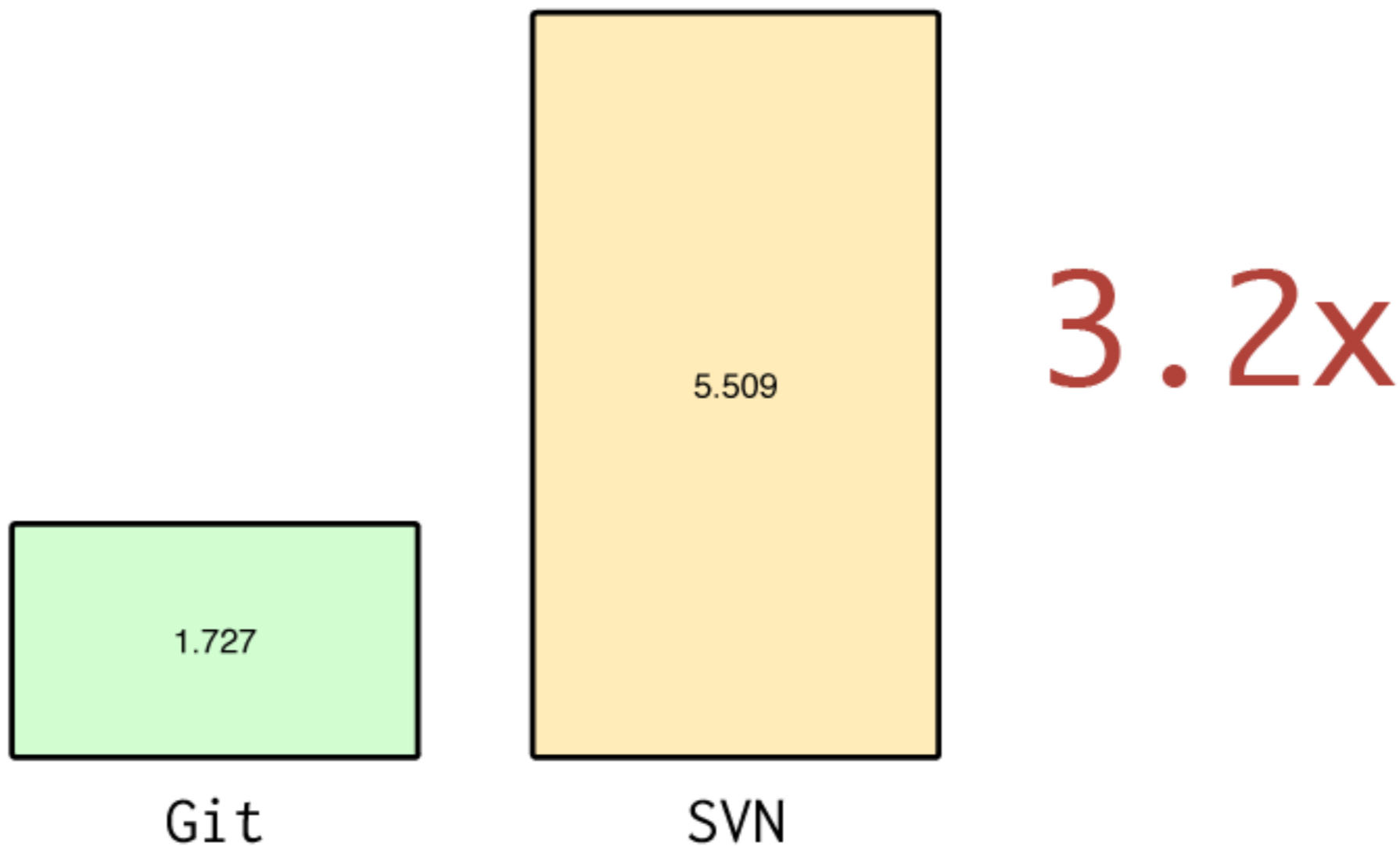
34x

Diff

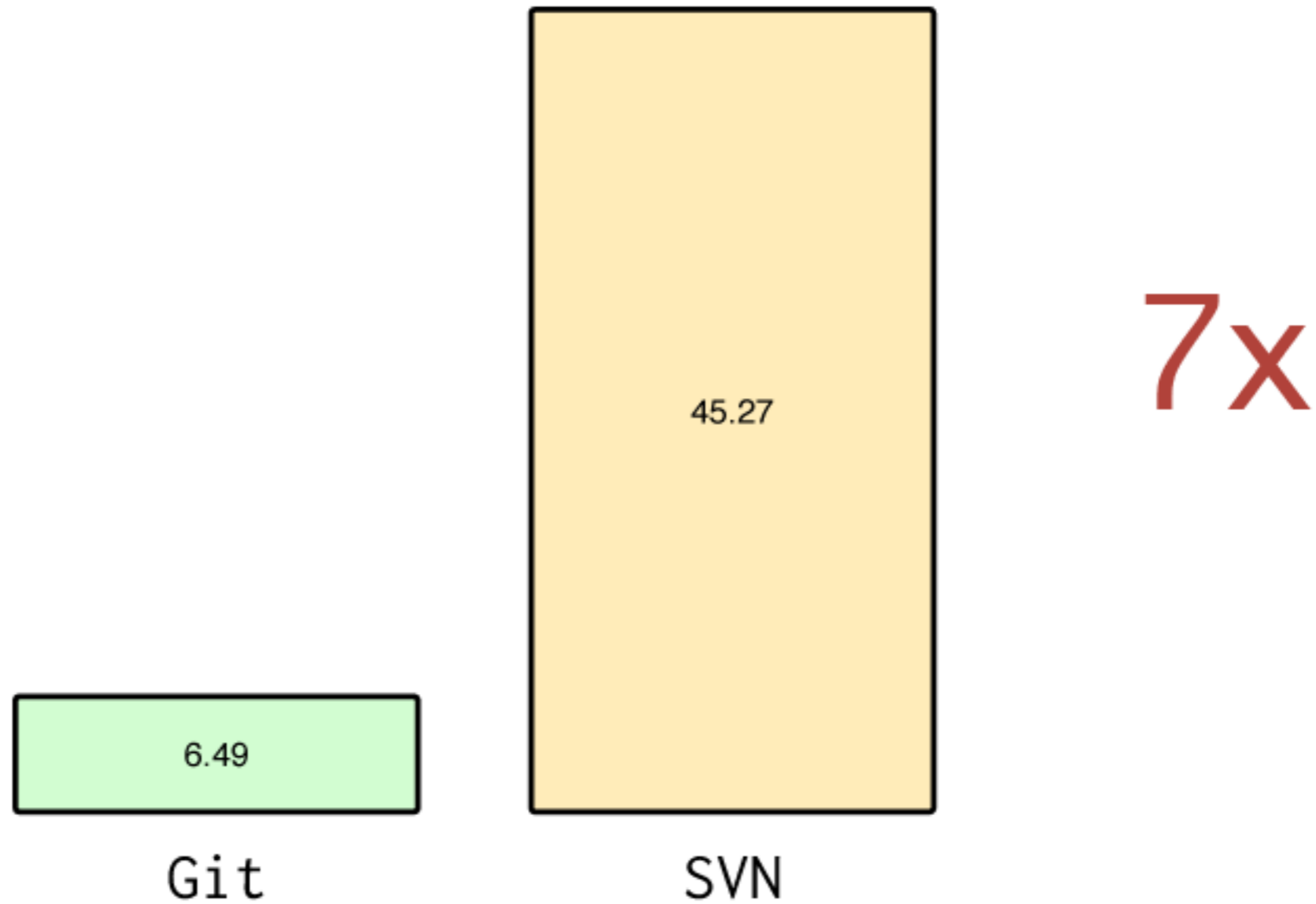


318x

Commit



Substantial Commit



Work Offline

No .svn Directories

Cheap Branching

```
$ 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
```

Easy Merging

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
$
```

The Git Directory

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
```

Areas of Interest


```
$ 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

`/etc/gitconfig`

`/etc/gitconfig`

`~/.gitconfig`

`/etc/gitconfig`

`~/.gitconfig`

`.git/config`

```
$ 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

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



```
$ touch hello_world.rb  
$ git init  
$ git add .  
$ git commit -m 'first commit'
```

```
$ tree -a
```

```
.
|-- .git
|   |-- COMMIT_EDITMSG
|   |-- HEAD
|   |-- branches
|   |-- config
|   |-- description
|   |-- hooks
|   |   |-- applypatch-msg.sample
|   |   `-- update.sample
|   |-- index
|   |-- info
|   |   `-- exclude
|   |-- logs
|   |   |-- HEAD
|   |   `-- refs
|   |       |-- heads
|   |       `-- master
|   |-- objects
|   |   |-- 32/09658ac8d80bc9726d3a33d77e3dfc5fe6035e
|   |   |-- 53/9cd7886a627841d525a78d45cbc6396be20b41
|   |   |-- e6/9de29bb2d1d6434b8b29ae775ad8c2e48c5391
|   |   |-- info
|   |   `-- pack
|   |-- refs
|   |   |-- heads
|   |   |   `-- master
|   |   `-- tags
|   `-- remotes
`-- hello_world.rb
```

```
17 directories, 33 files
```

```
$ tree -a
```

```
.  
|-- .git  
|   |-- COMMIT_EDITMSG  
|   |-- HEAD  
|   |-- branches  
|   |-- config  
|   |-- description  
|   |-- hooks  
|   |   |-- applypatch-msg.sample  
|   |   `-- update.sample  
|   |-- index  
|   |-- info  
|   |   `-- exclude  
|   |-- logs  
|   |   |-- HEAD  
|   |   `-- refs  
|   |       |-- heads  
|   |       `-- master  
|   |-- objects  
|   |   |-- 32/09658ac8d80bc9726d3a33d77e3dfc5fe6035e  
|   |   |-- 53/9cd7886a627841d525a78d45cbc6396be20b41  
|   |   |-- e6/9de29bb2d1d6434b8b29ae775ad8c2e48c5391  
|   |   |-- info  
|   |   `-- pack  
|   |-- refs  
|   |   |-- heads  
|   |   |   `-- master  
|   |   `-- tags  
|   `-- remotes  
`-- hello_world.rb
```

```
17 directories, 33 files
```

```
$ tree -a
```

```
.  
|-- .git  
|   |-- COMMIT_EDITMSG  
|   |-- HEAD  
|   |-- branches  
|   |-- config  
|   |-- description  
|   |-- hooks  
|   |   |-- applypatch-msg.sample  
|   |   `-- update.sample  
|   |-- index  
|   |-- info  
|   |   `-- exclude  
|   |-- logs  
|   |   |-- HEAD  
|   |   `-- refs  
|   |       |-- heads  
|   |       `-- master  
|   |-- objects  
|   |   |-- 32/09658ac8d80bc9726d3a33d77e3dfc5fe6035e  
|   |   |-- 53/9cd7886a627841d525a78d45cbc6396be20b41  
|   |   |-- e6/9de29bb2d1d6434b8b29ae775ad8c2e48c5391  
|   |   |-- info  
|   |   `-- pack  
|   |-- refs  
|   |   |-- heads  
|   |   |   |-- master  
|   |   |   `-- tags  
|   |   `-- remotes  
|   `-- hello_world.rb
```

```
17 directories, 33 files
```

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

repository

“staging”

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 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
```

```
$ 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

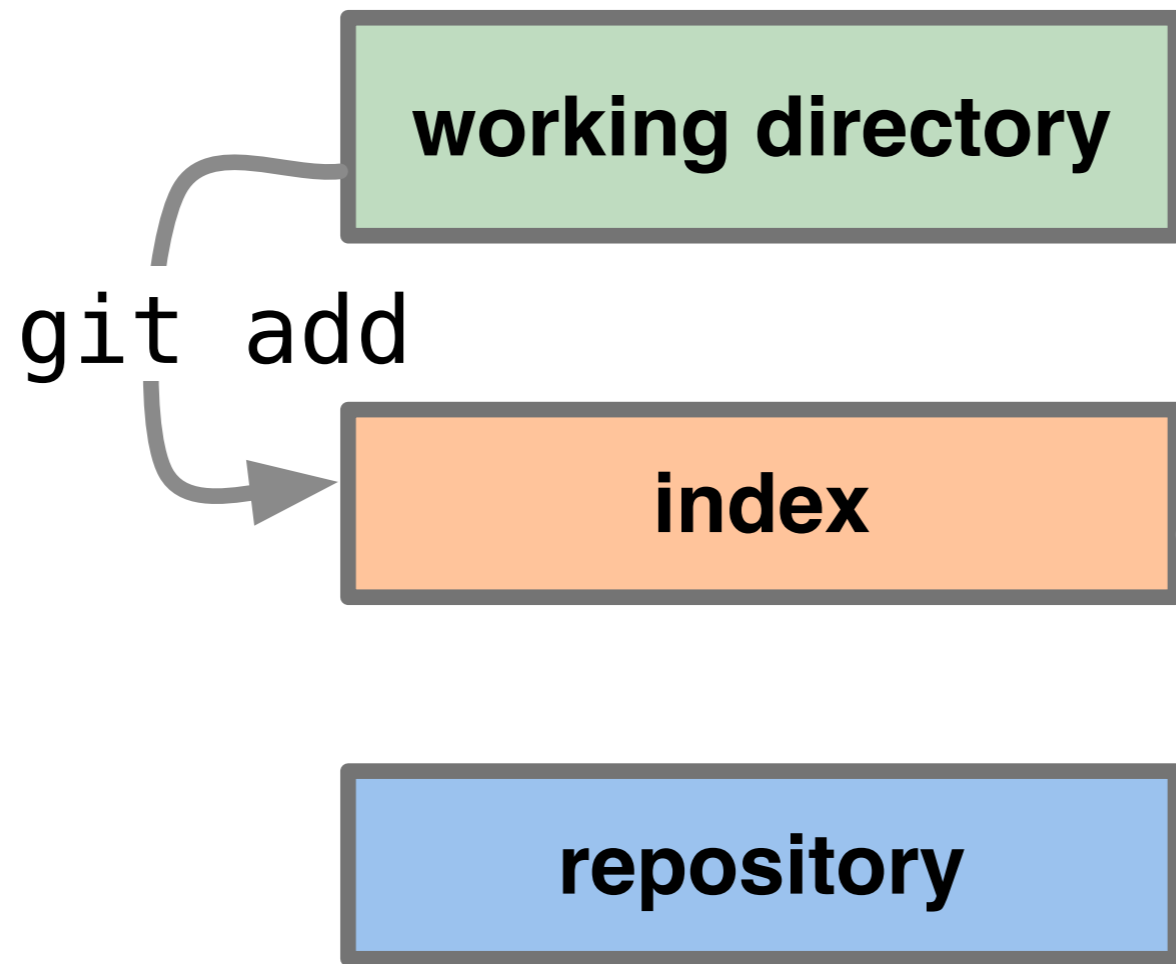
Commit the changes

`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 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('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
```

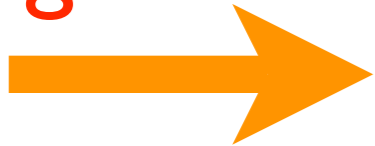
```
#
```

```
# 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(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()
```

**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
```

```
#
```

A Basic Workflow

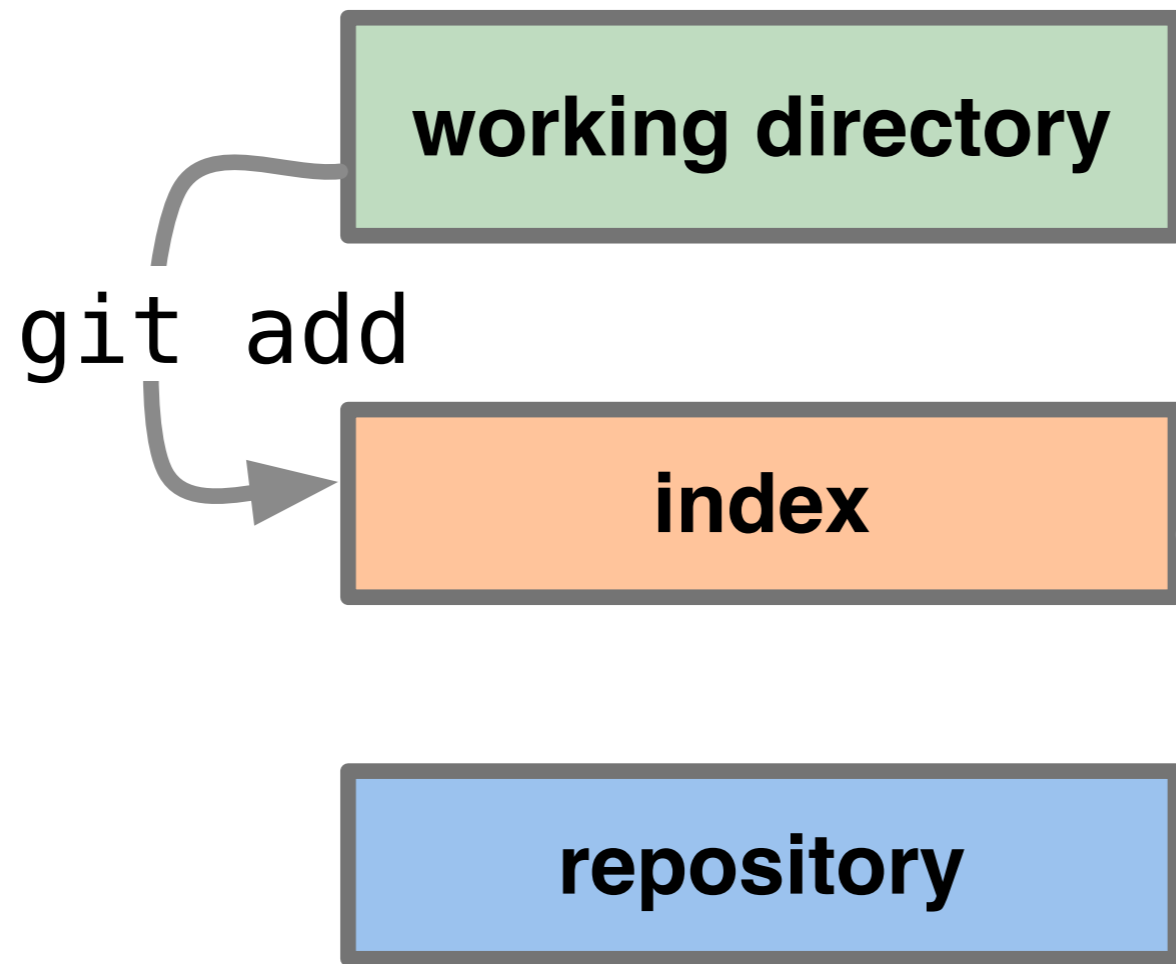
Edit files

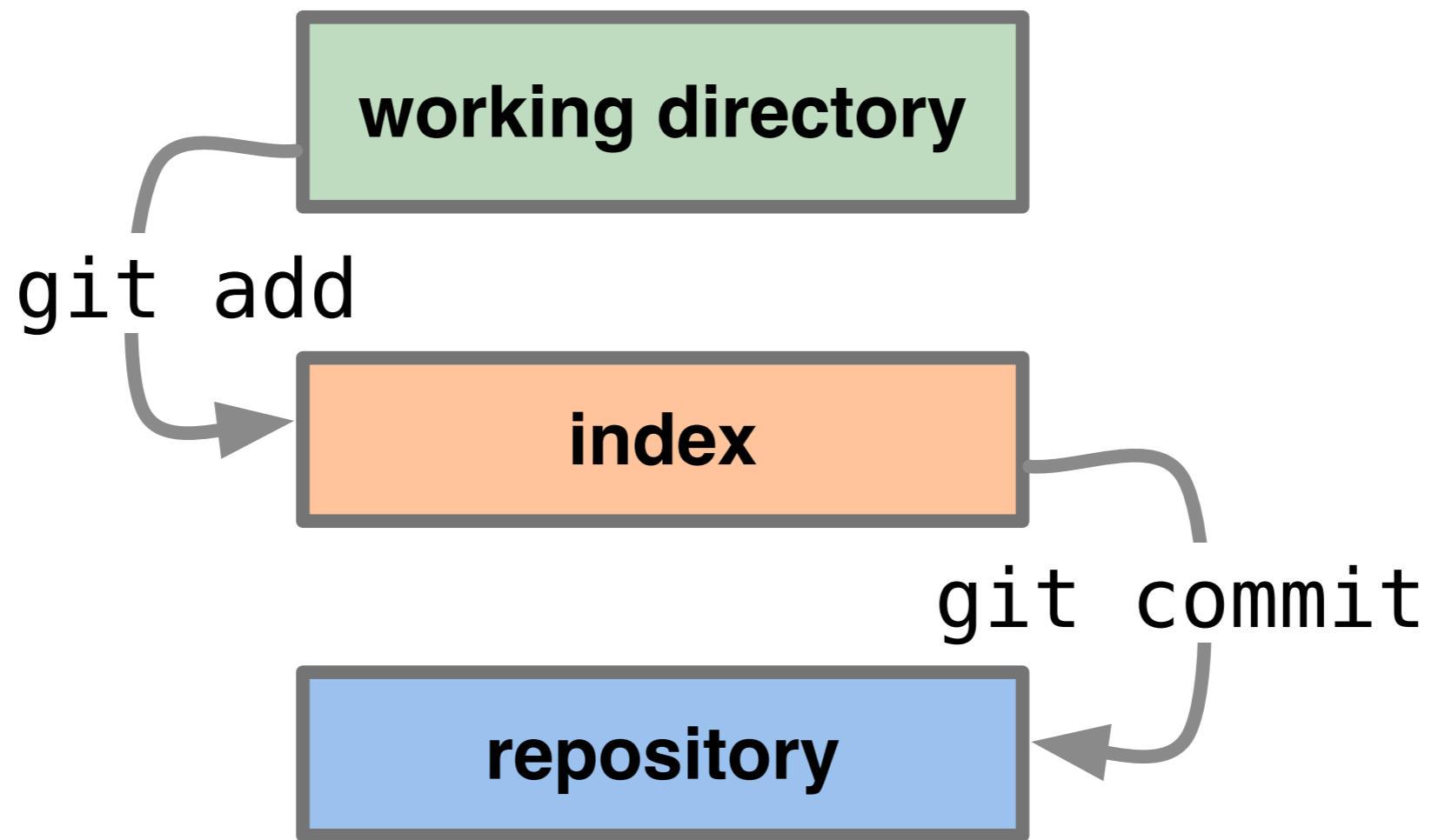
Stage the changes

Review your changes

Commit the changes

`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

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 Basicer Workflow

A Basicer Workflow

Edit files

vim / emacs / etc

A Basicer 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

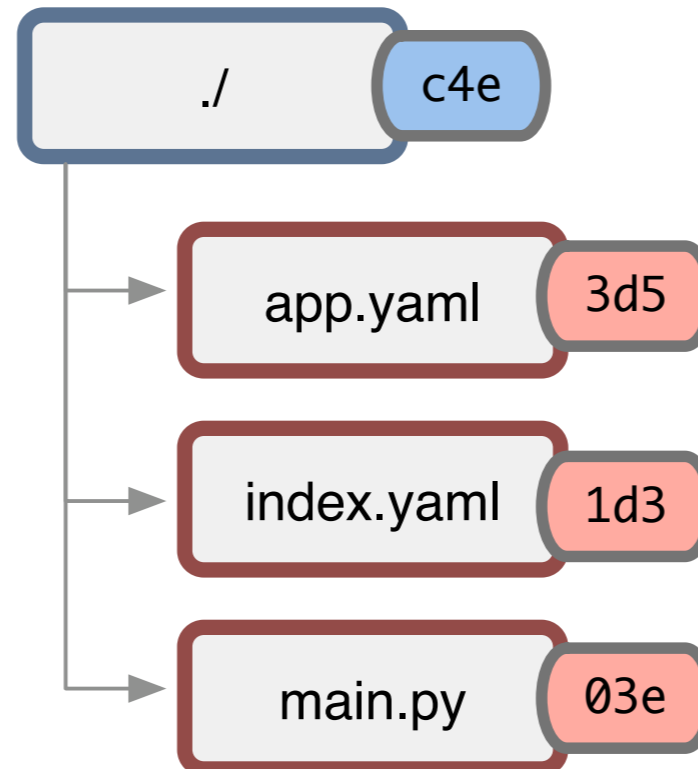
descriptive commit message
```

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

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

77d3001a1de6bf8f5e431972fe4d25b01e595c0b

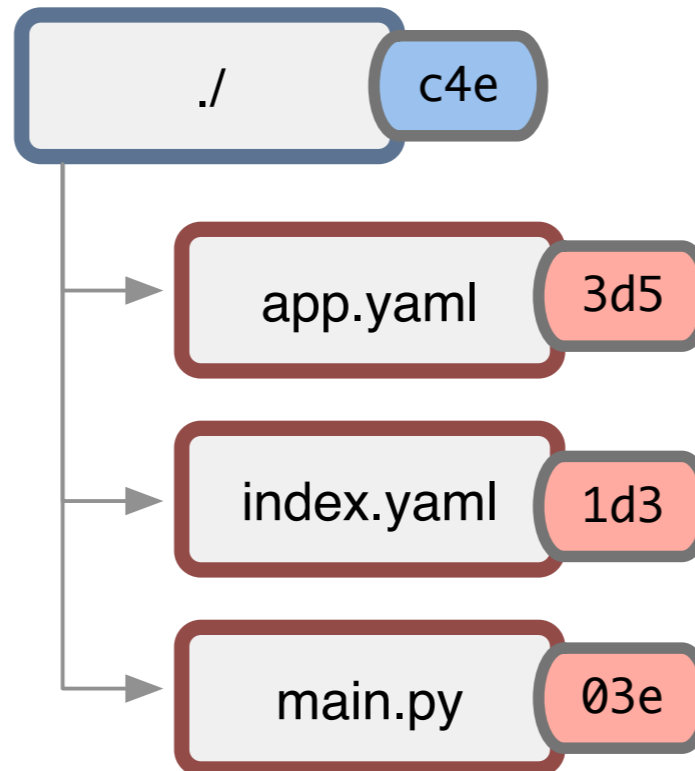
commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	



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

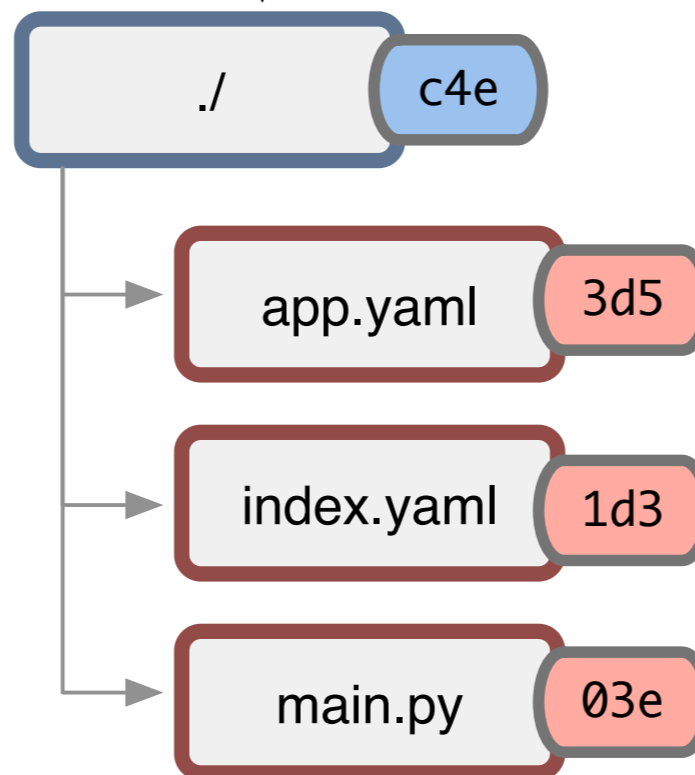
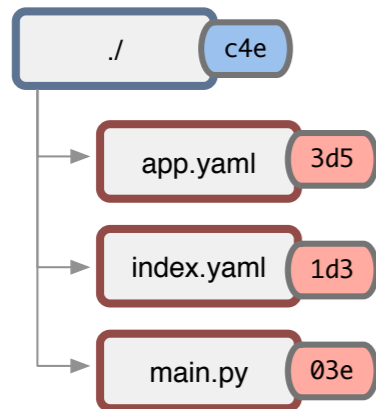


commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	



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	

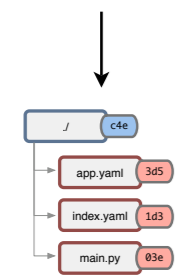
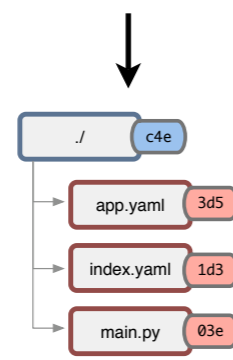
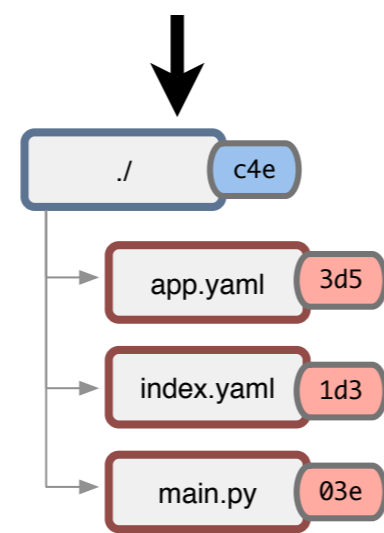
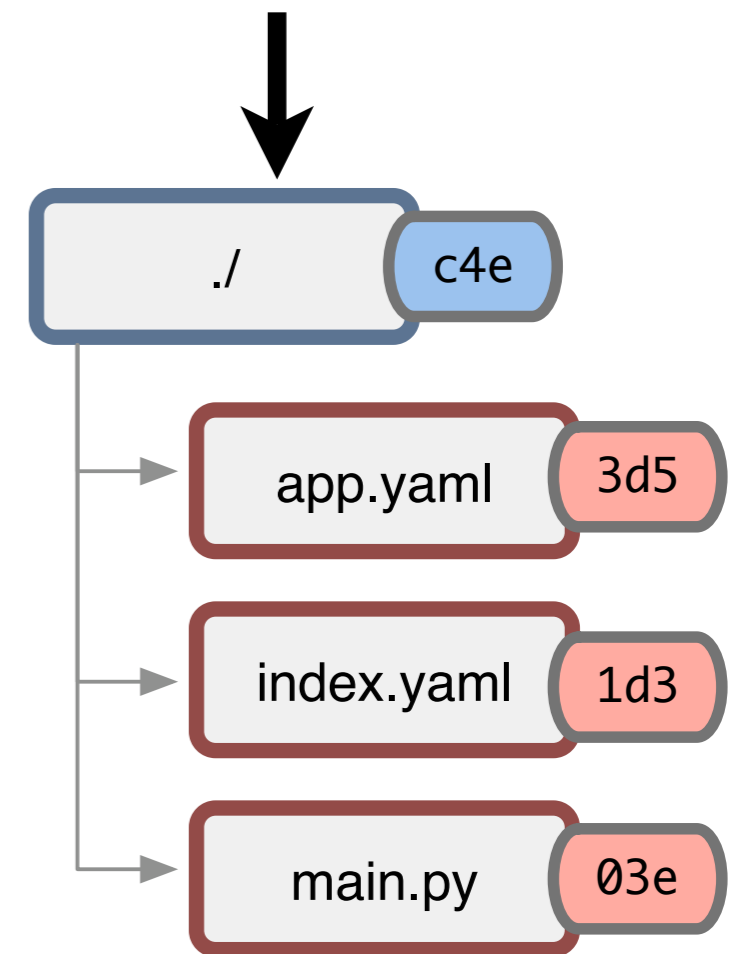


commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	

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	2fe65
parent	90ecd
author	Scott
committer	Scott
this is the commit before that and I'm not sure why	

commit	size
tree	2fe65
parent	90ecd
author	Scott
committer	Scott
this is the commit before that and I'm not sure why	



commit	size
tree	c4ec5
parent	77d 49e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	



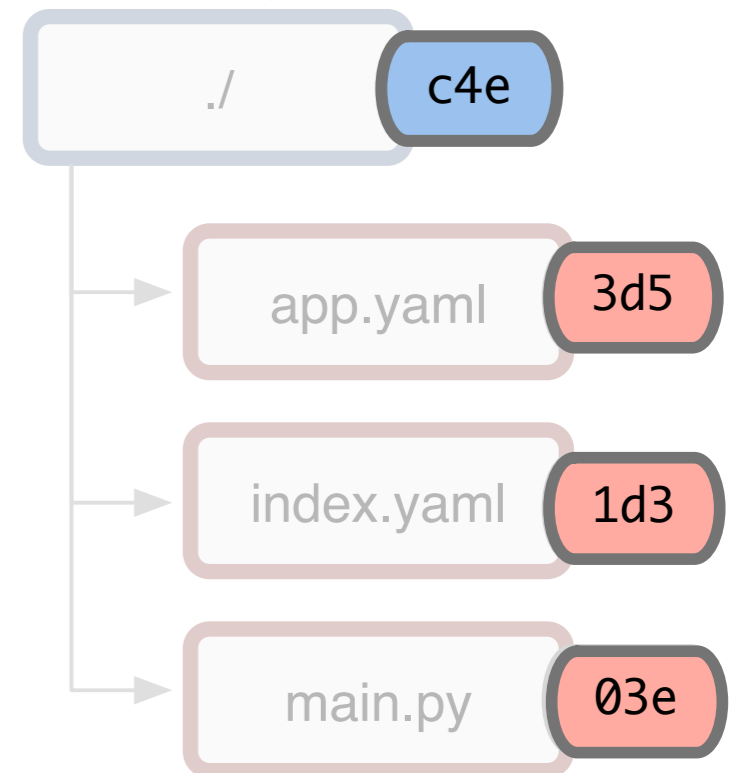
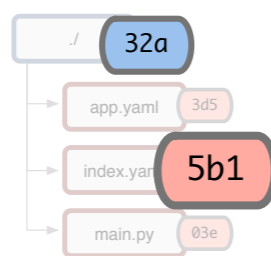
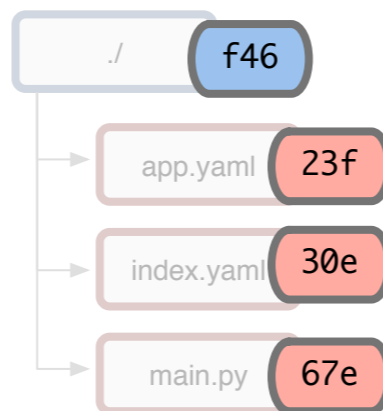
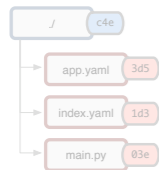
commit	size
tree	2de54
parent	a14 ef
author	Scott
committer	Scott
this is the previous commit and I am very proud of it	

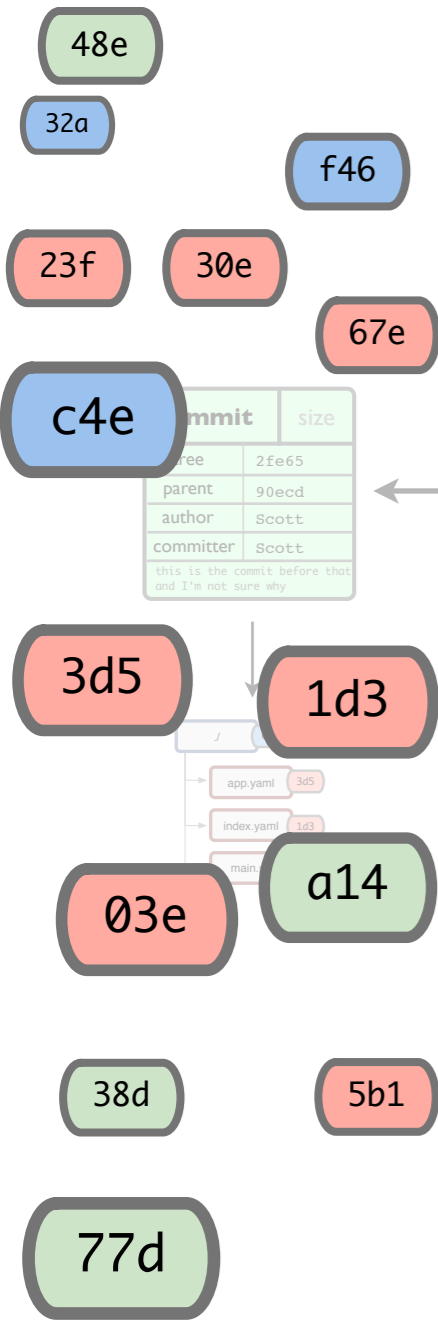


commit	size
tree	2fe65
parent	38d d
author	Scott
committer	Scott
this is the commit before that and I'm not sure why	

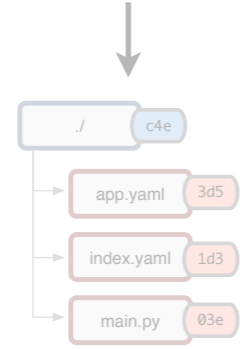


commit	size
tree	48e
parent	
author	
committer	Scott
this is the commit before that and I'm not sure why	

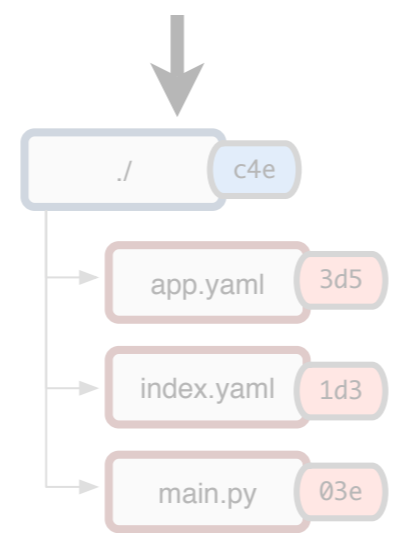




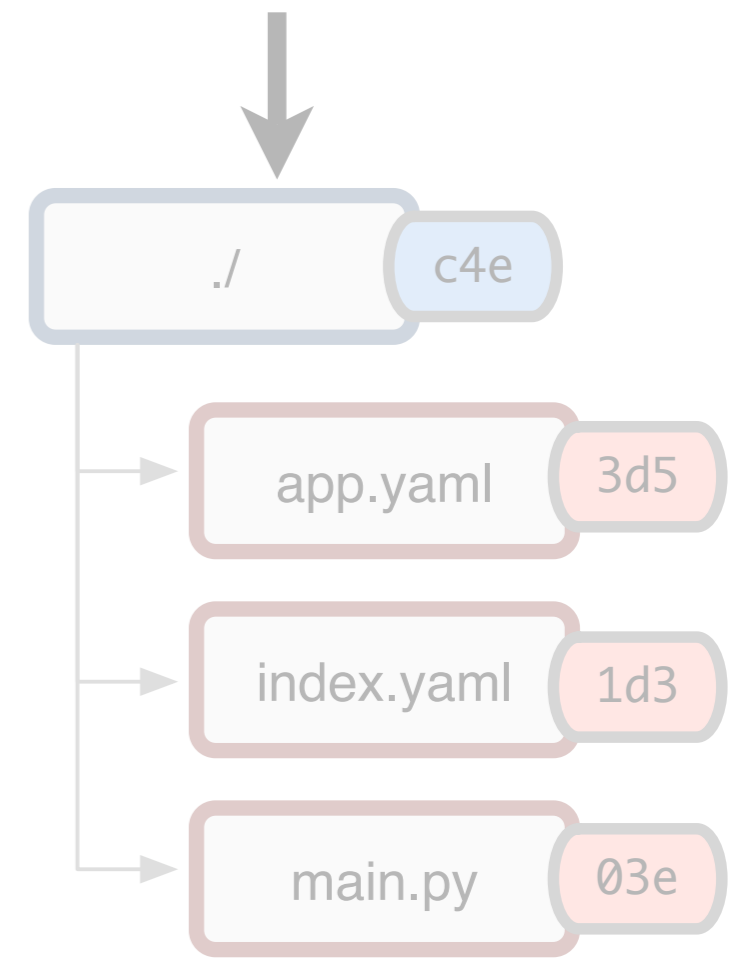
commit	size
tree	2fe65
parent	90ecd
author	Scott
committer	Scott
this is the commit before that and I'm not sure why	



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

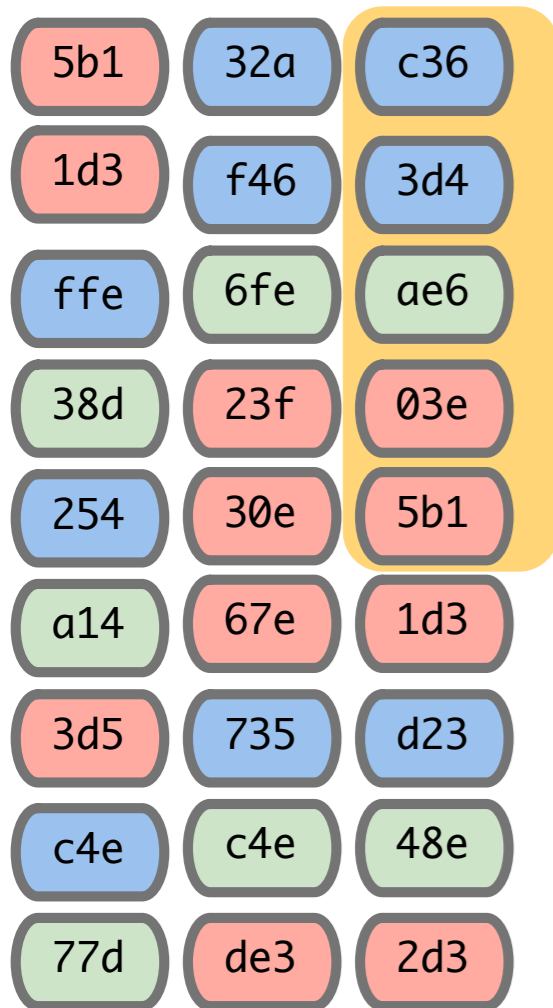
5b1	32a	c36
1d3	f46	3d4
ffe	6fe	ae6
38d	23f	03e
254	30e	5b1
a14	67e	1d3
3d5	735	d23
c4e	c4e	48e
77d	de3	2d3

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
```

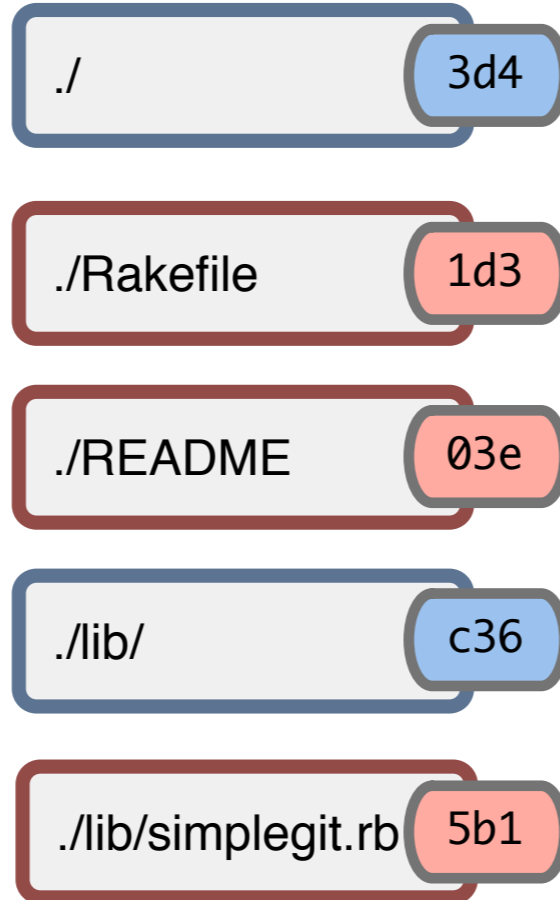
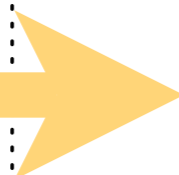
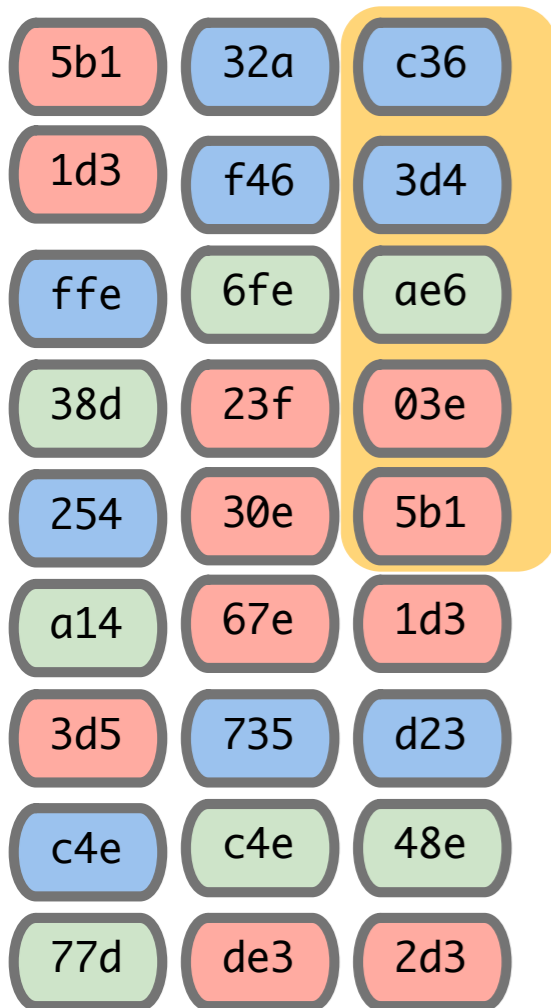
Repository



```
git checkout ae635
```

Repository

Index



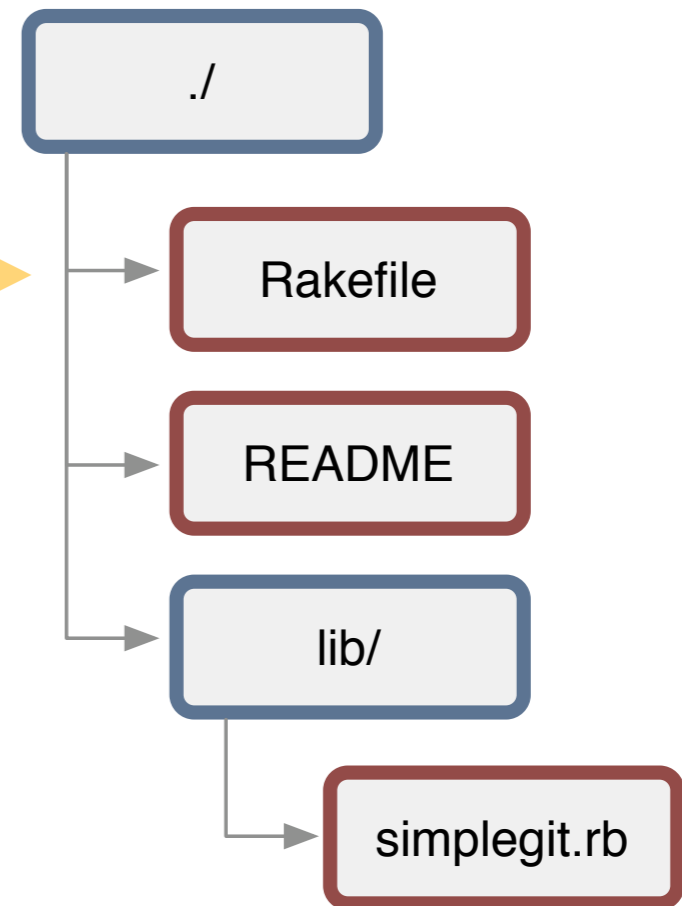
Repository

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

Index

./	3d4
./Rakefile	1d3
./README	03e
./lib/	c36
./lib/simplegit.rb	5b1

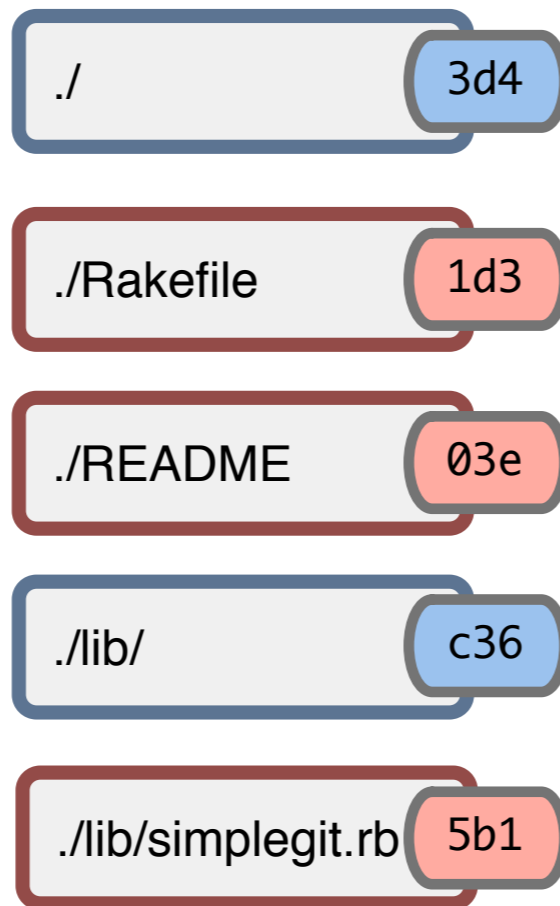
Working Directory



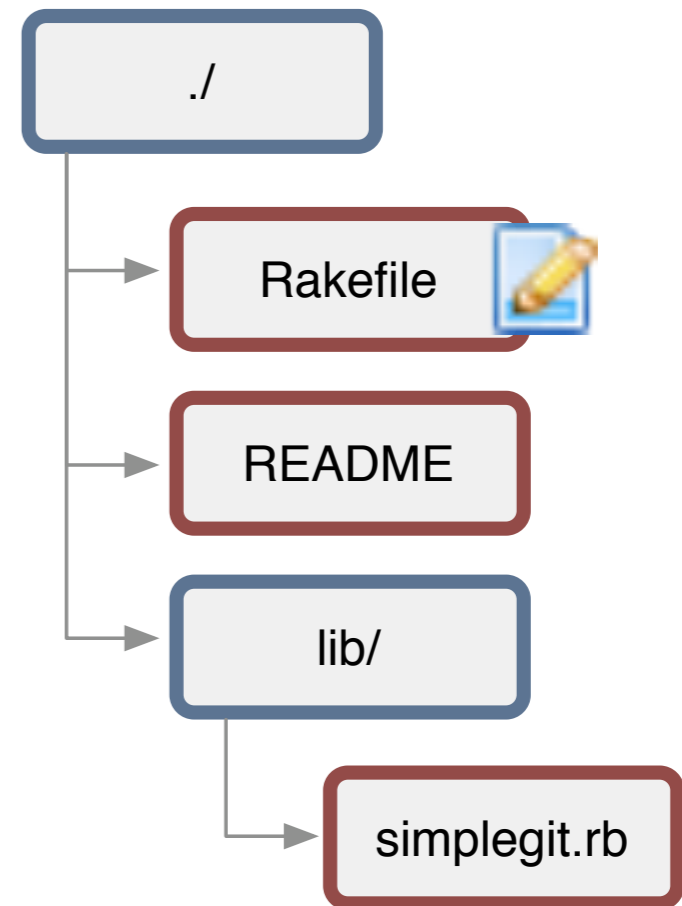
Repository



Index



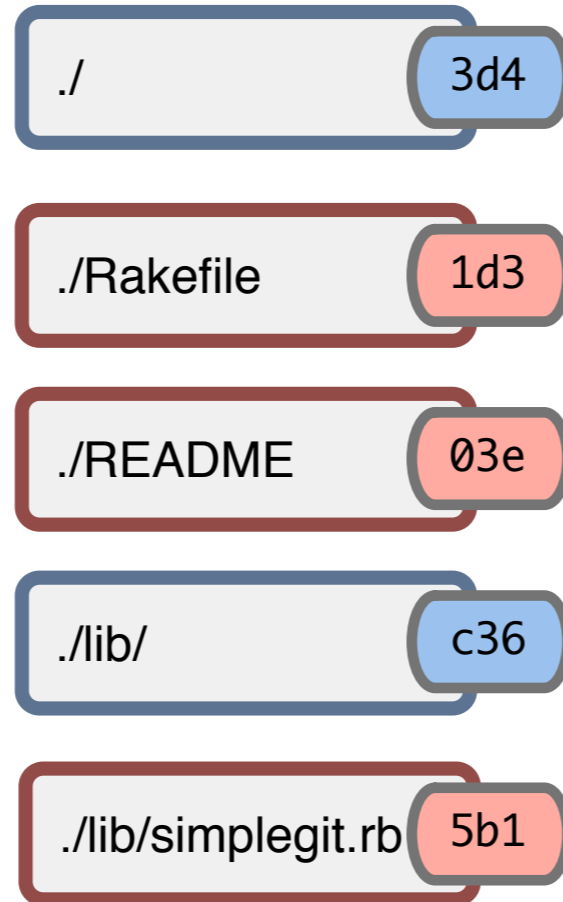
Working Directory



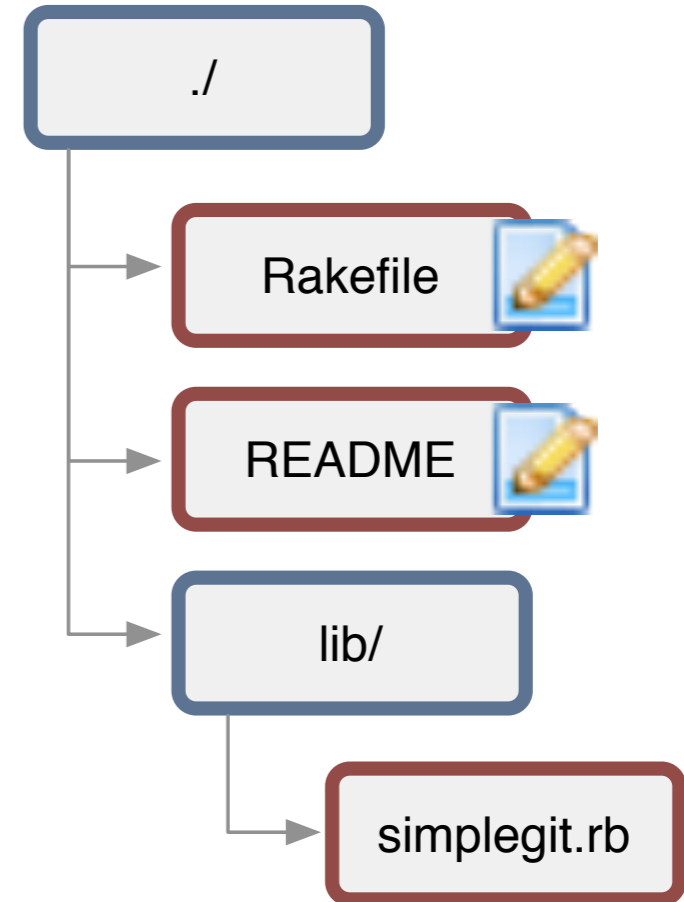
Repository



Index



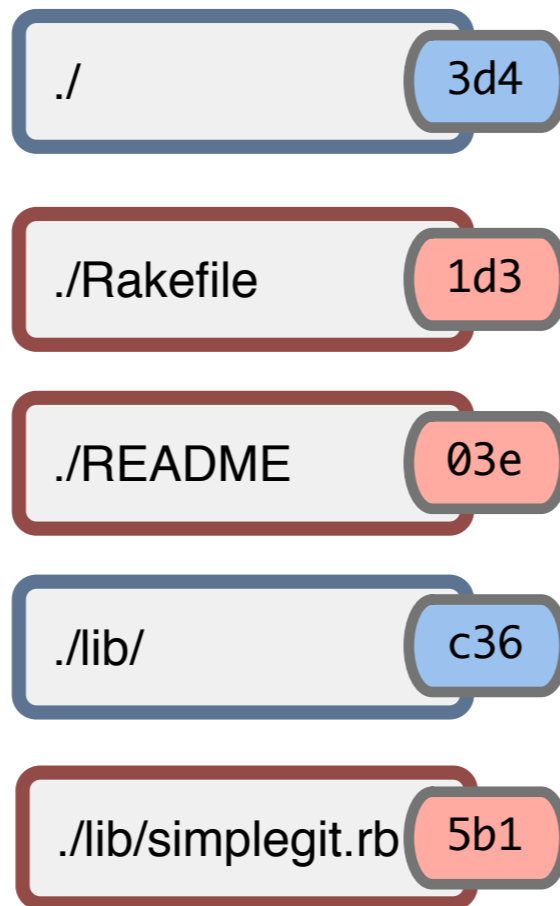
Working Directory



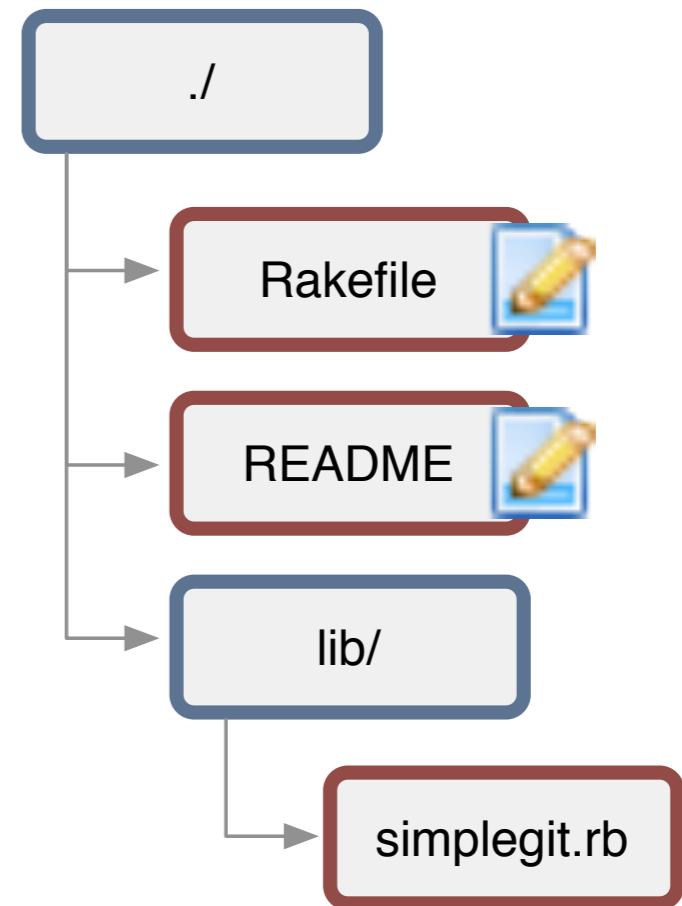
Repository



Index



Working Directory

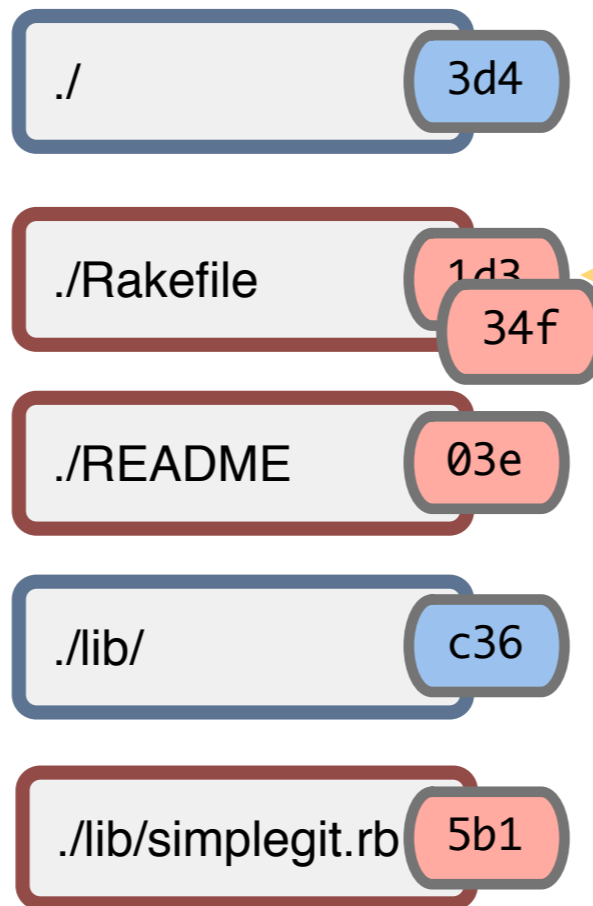


git add

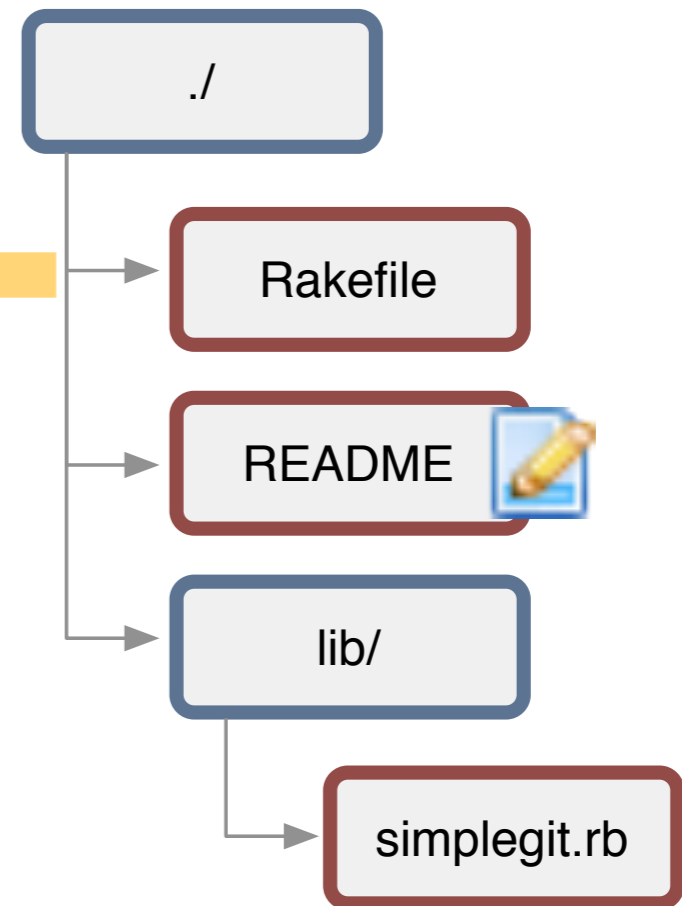
Repository



Index



Working Directory

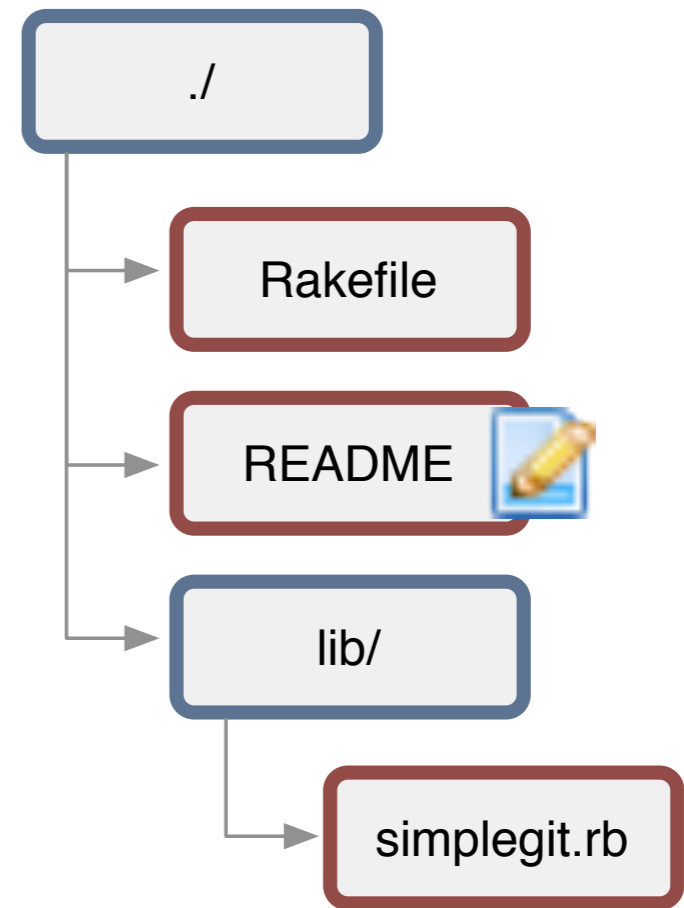
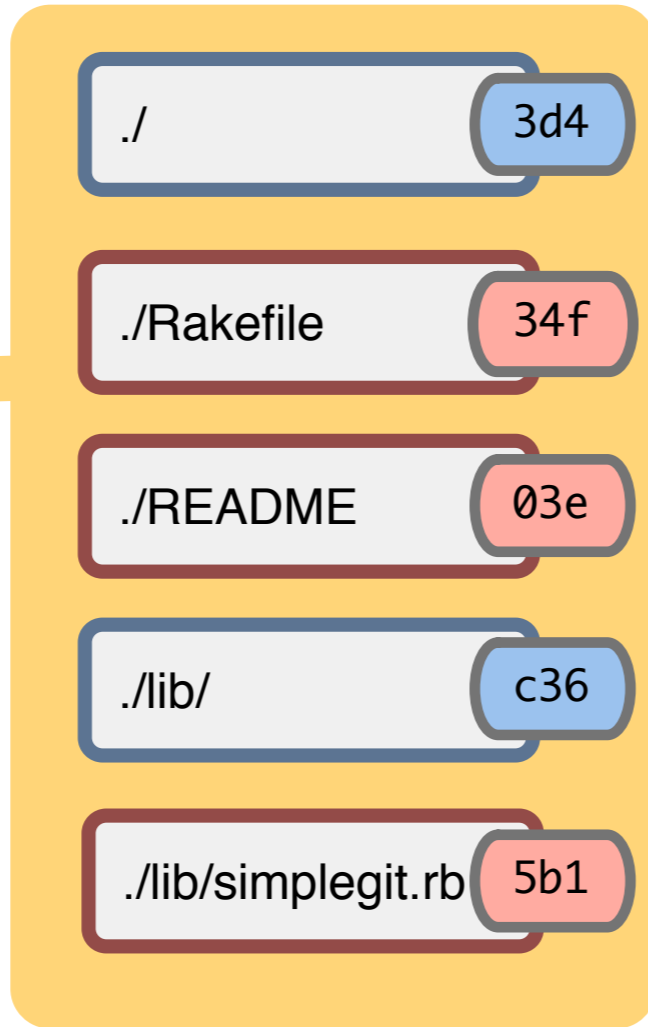
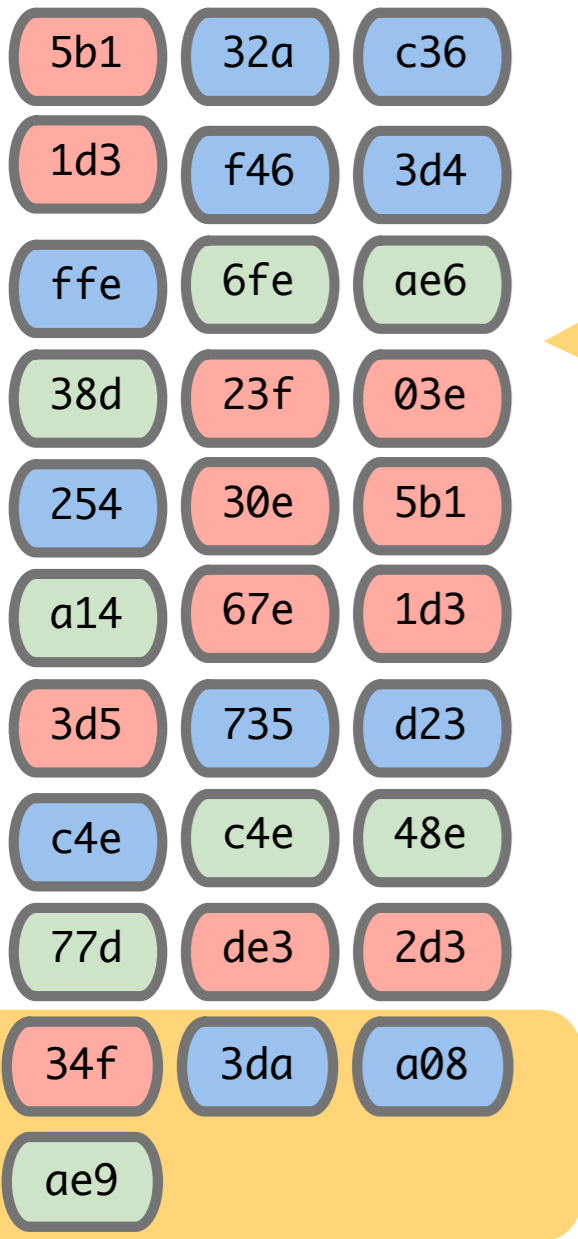


git add

Repository

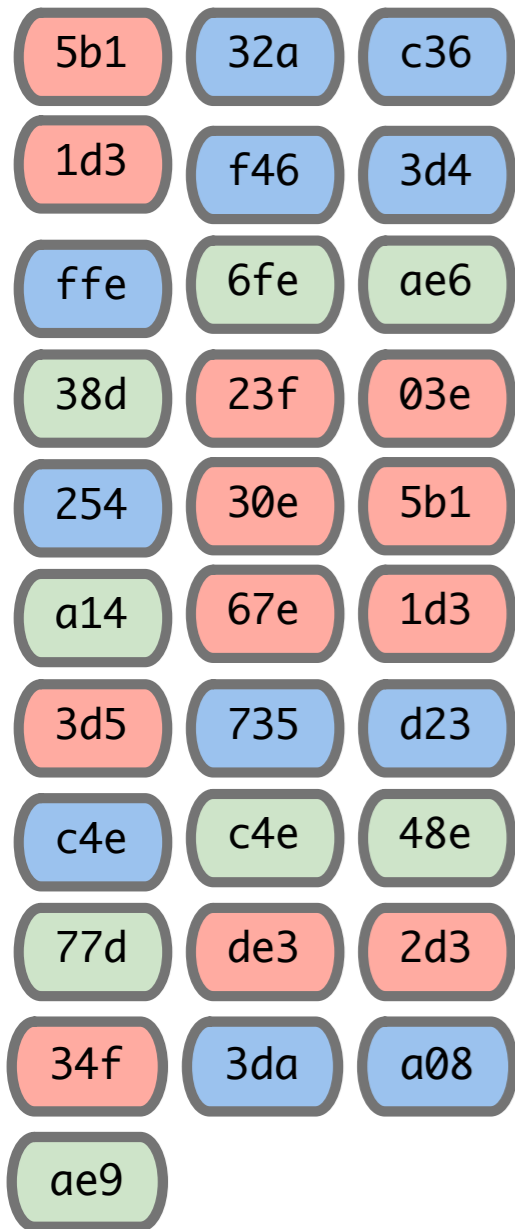
Index

Working Directory

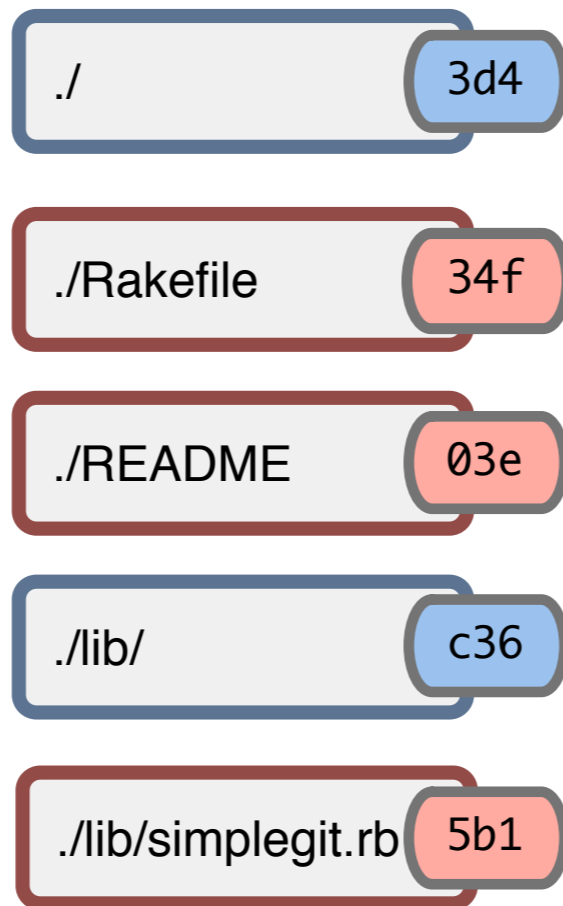


git commit

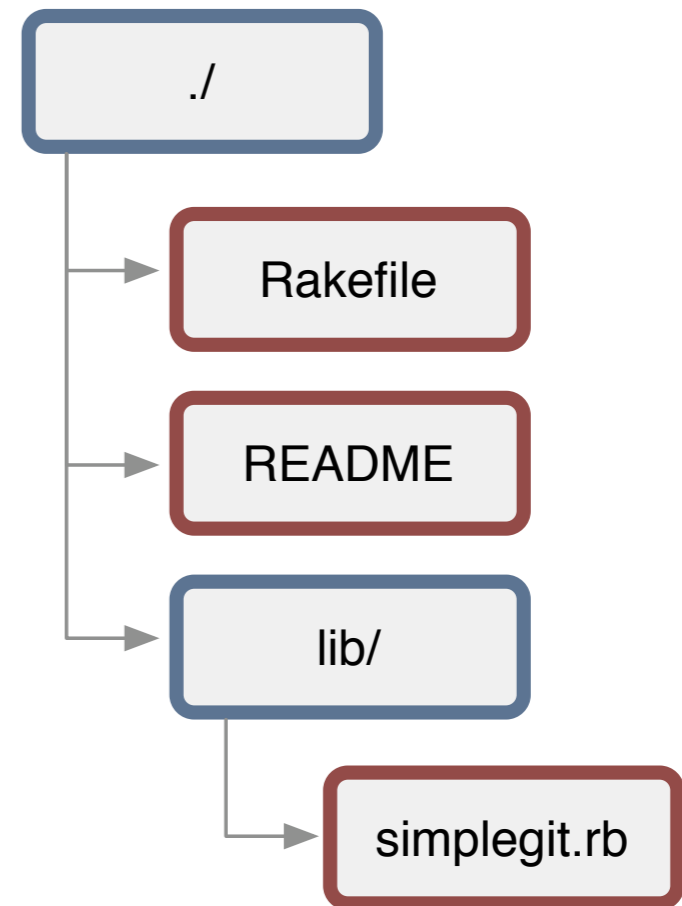
Repository



Index



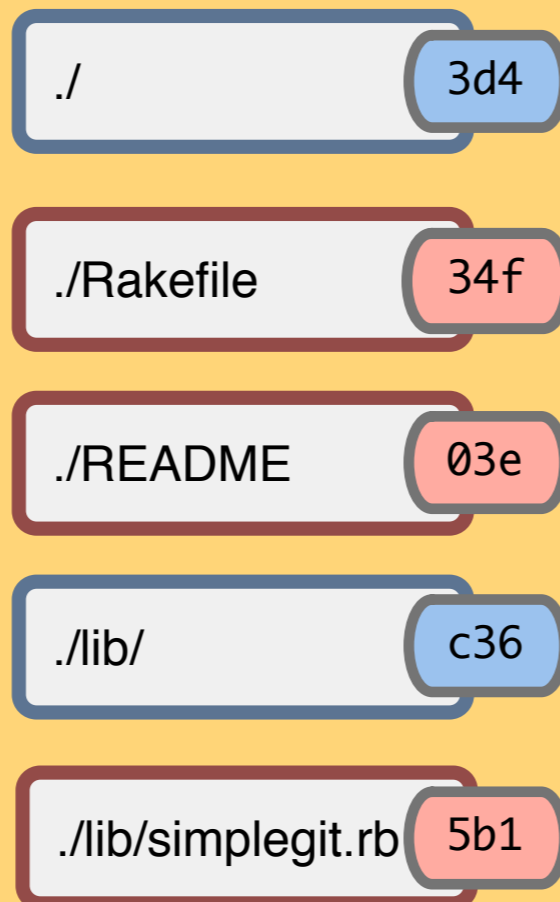
Working Directory



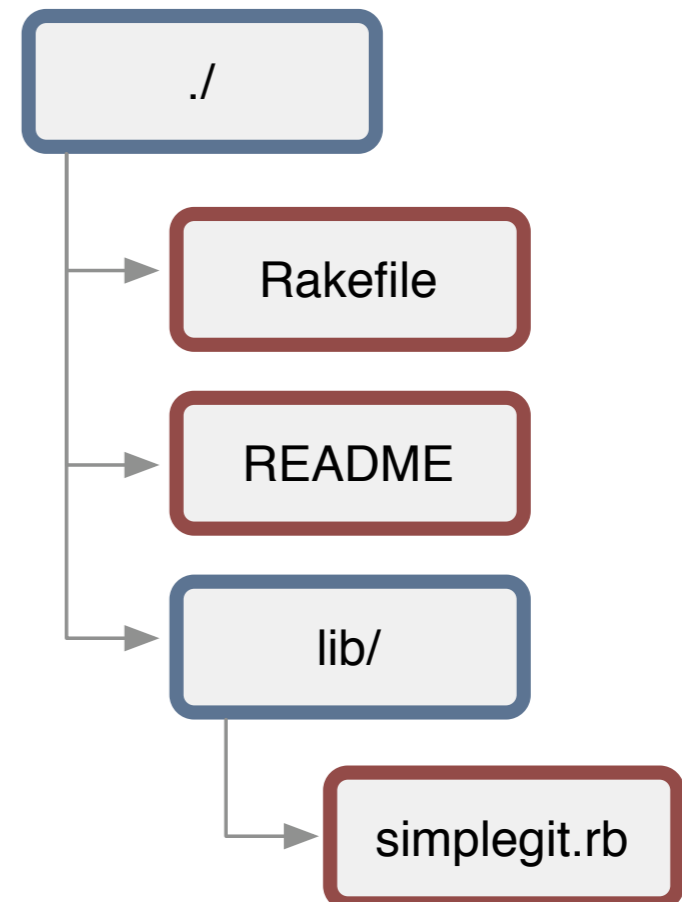
Repository



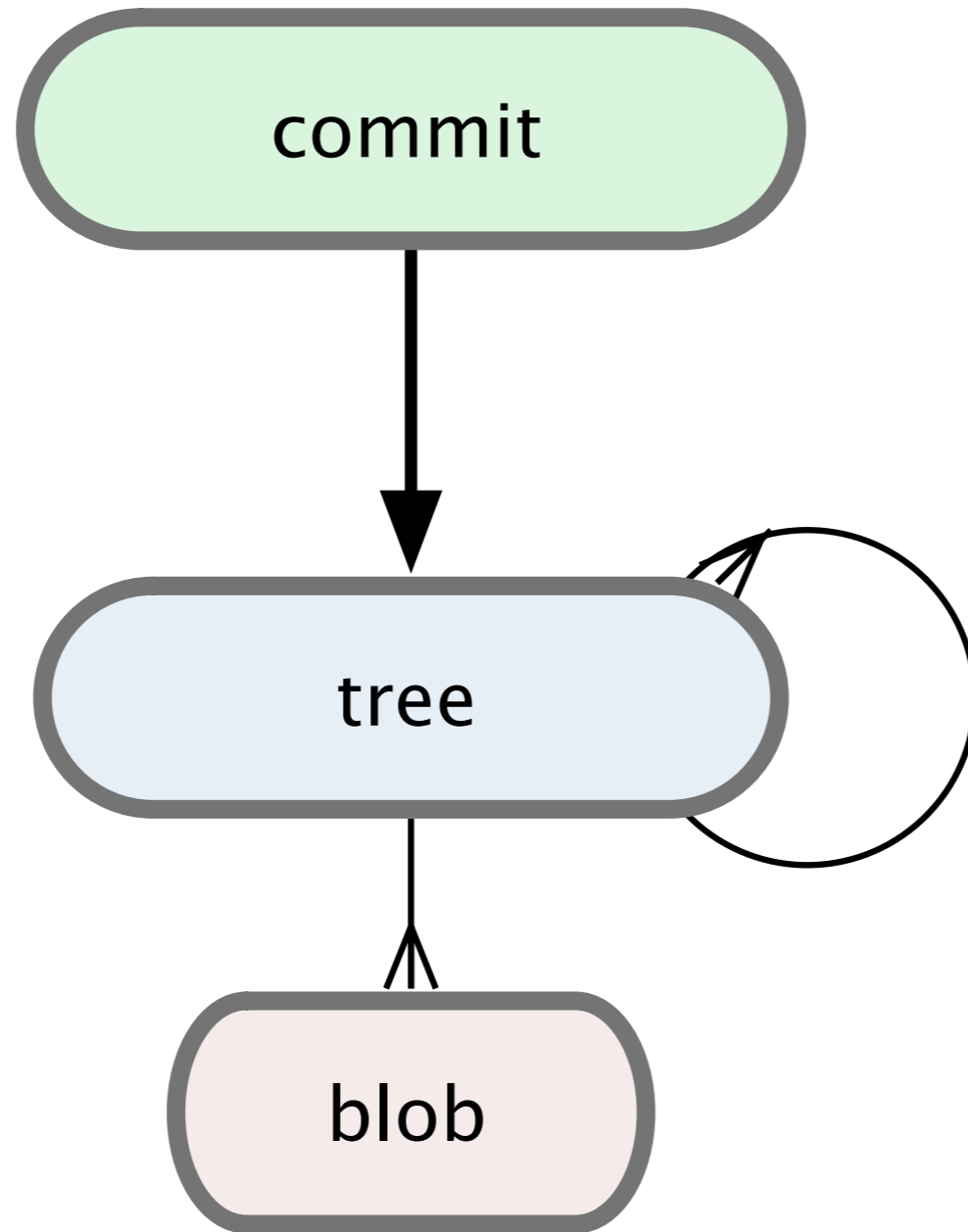
Index



Working Directory



object model

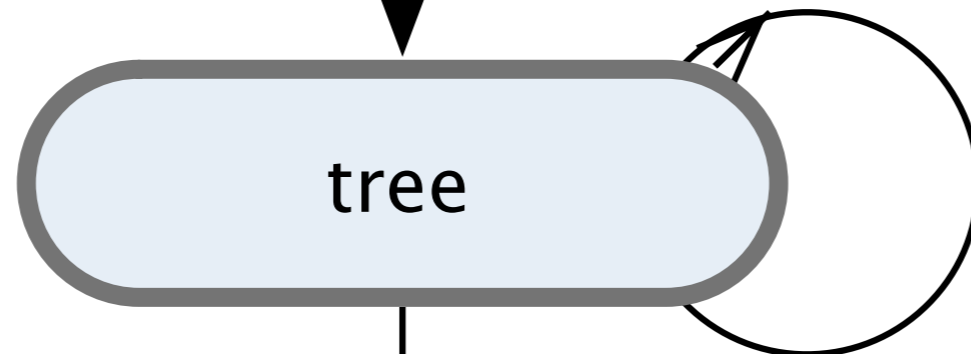


object model

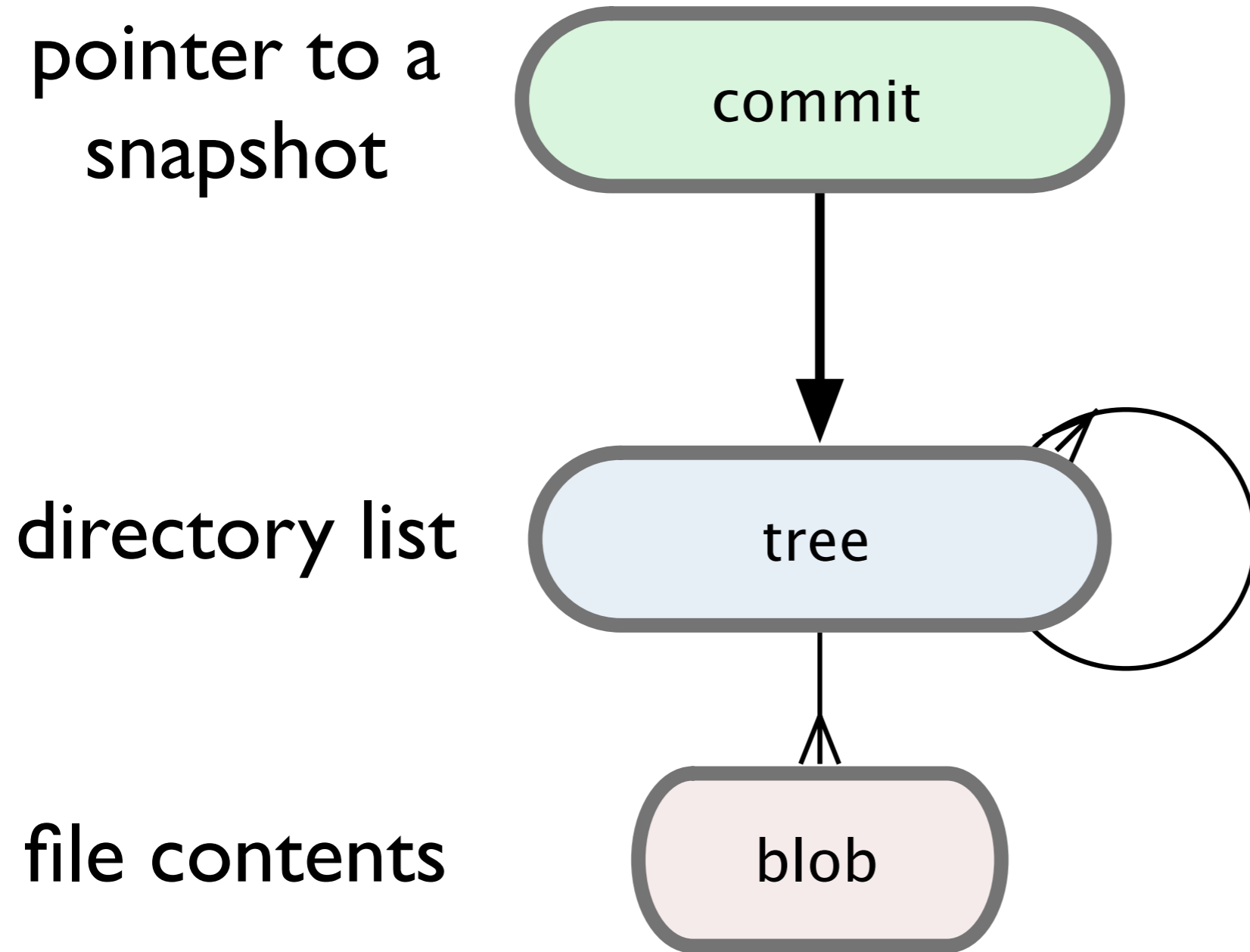
pointer to a
snapshot



directory list



file contents

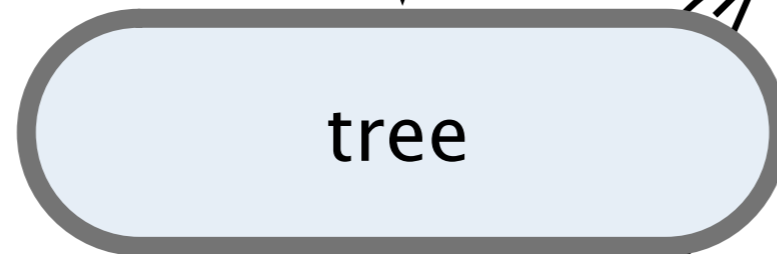


object model

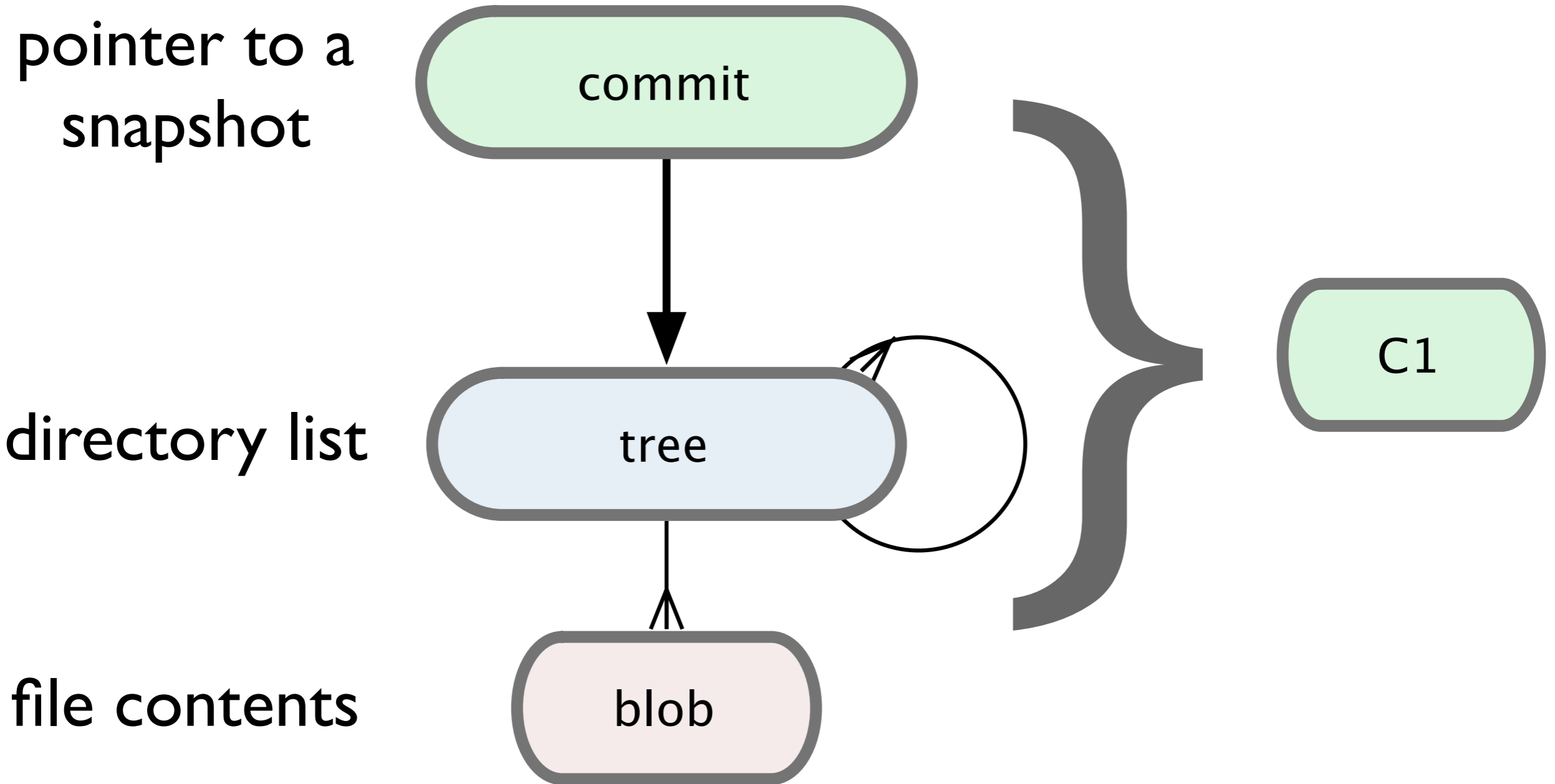
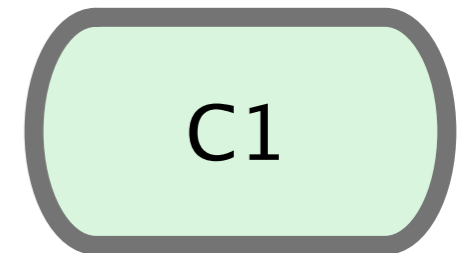
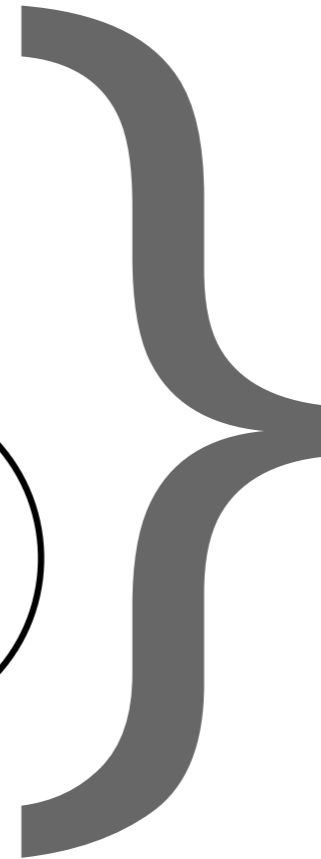
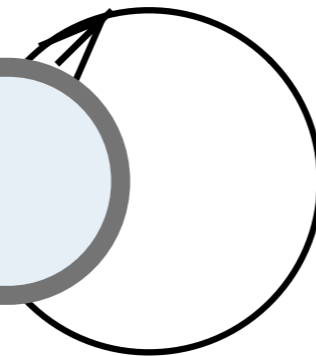
pointer to a
snapshot



directory list



file contents

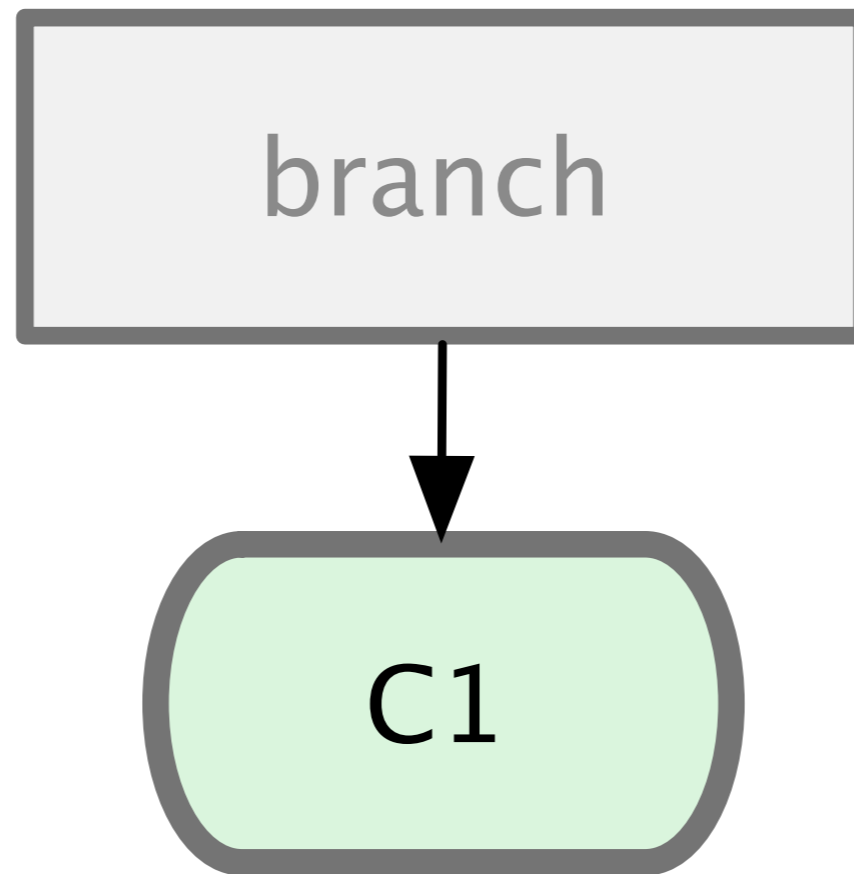


Branching and Merging

branches

branches

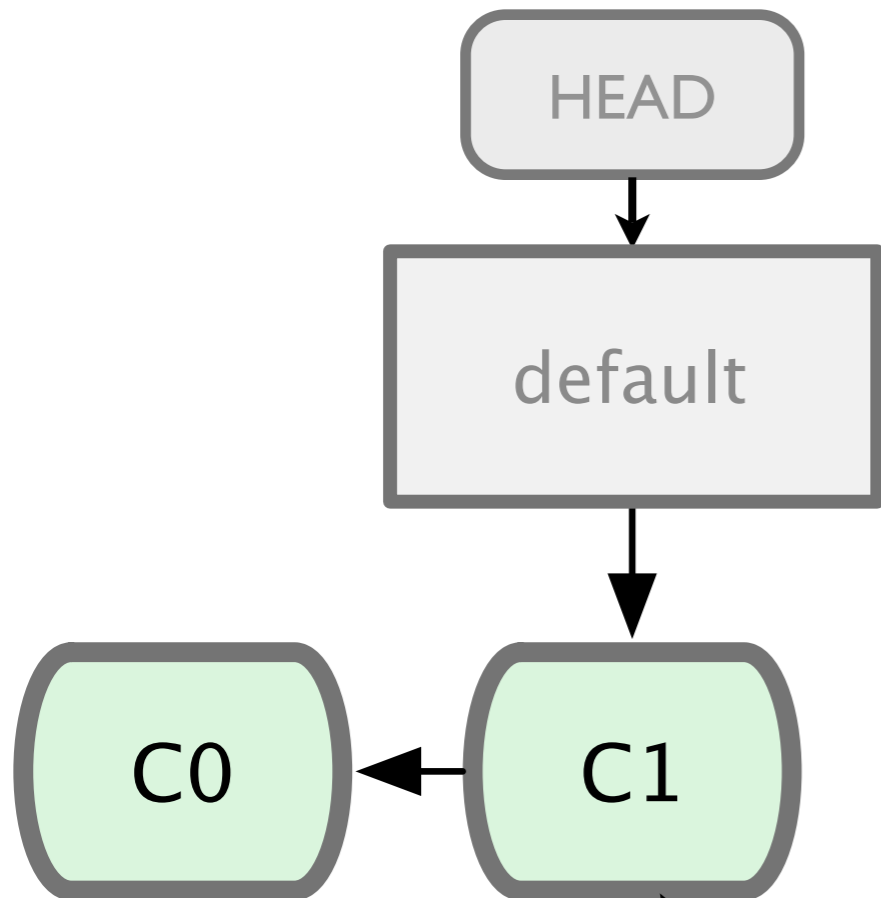
lightweight, movable
pointers to a commit

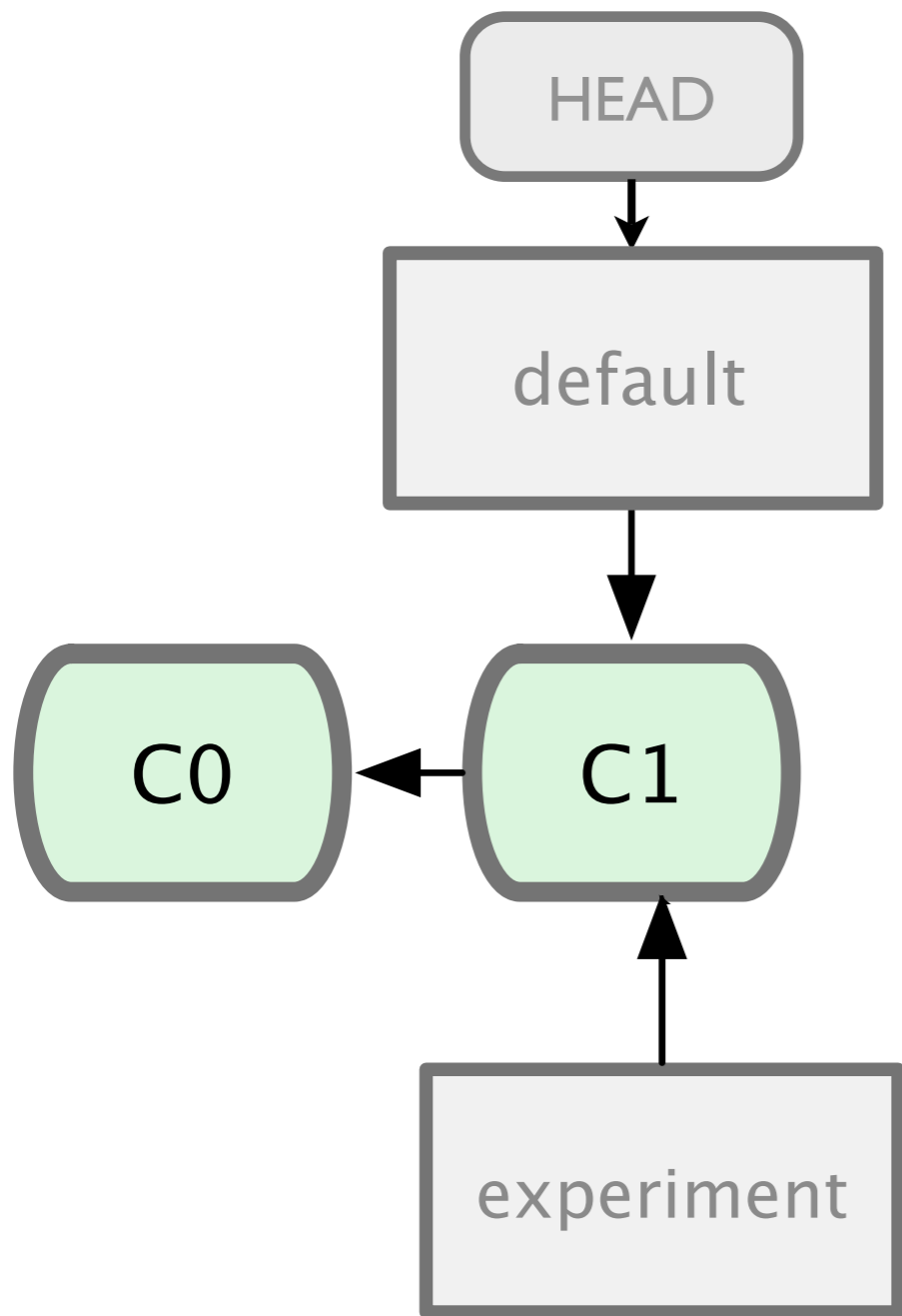


branching

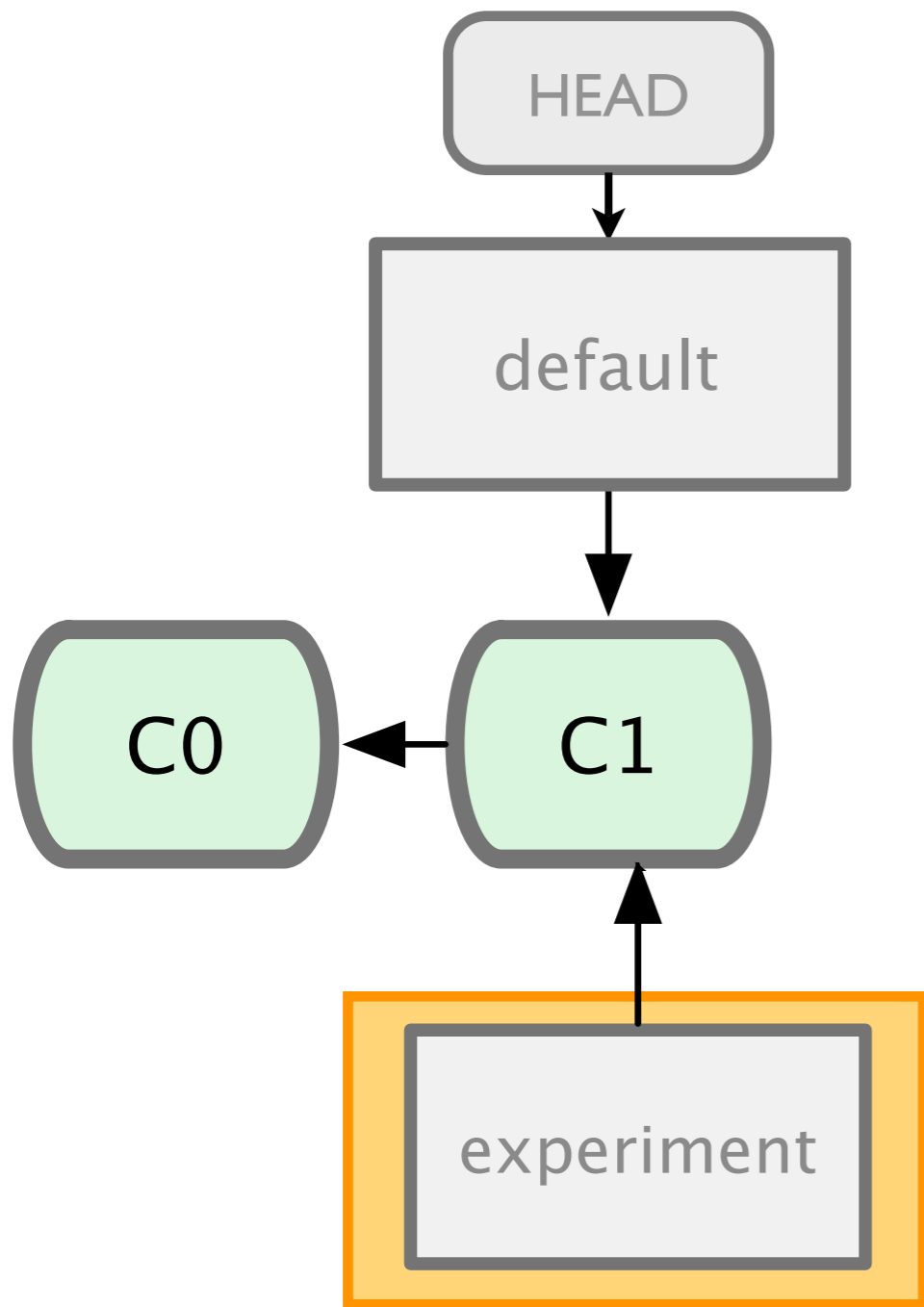
git branch

`git checkout`

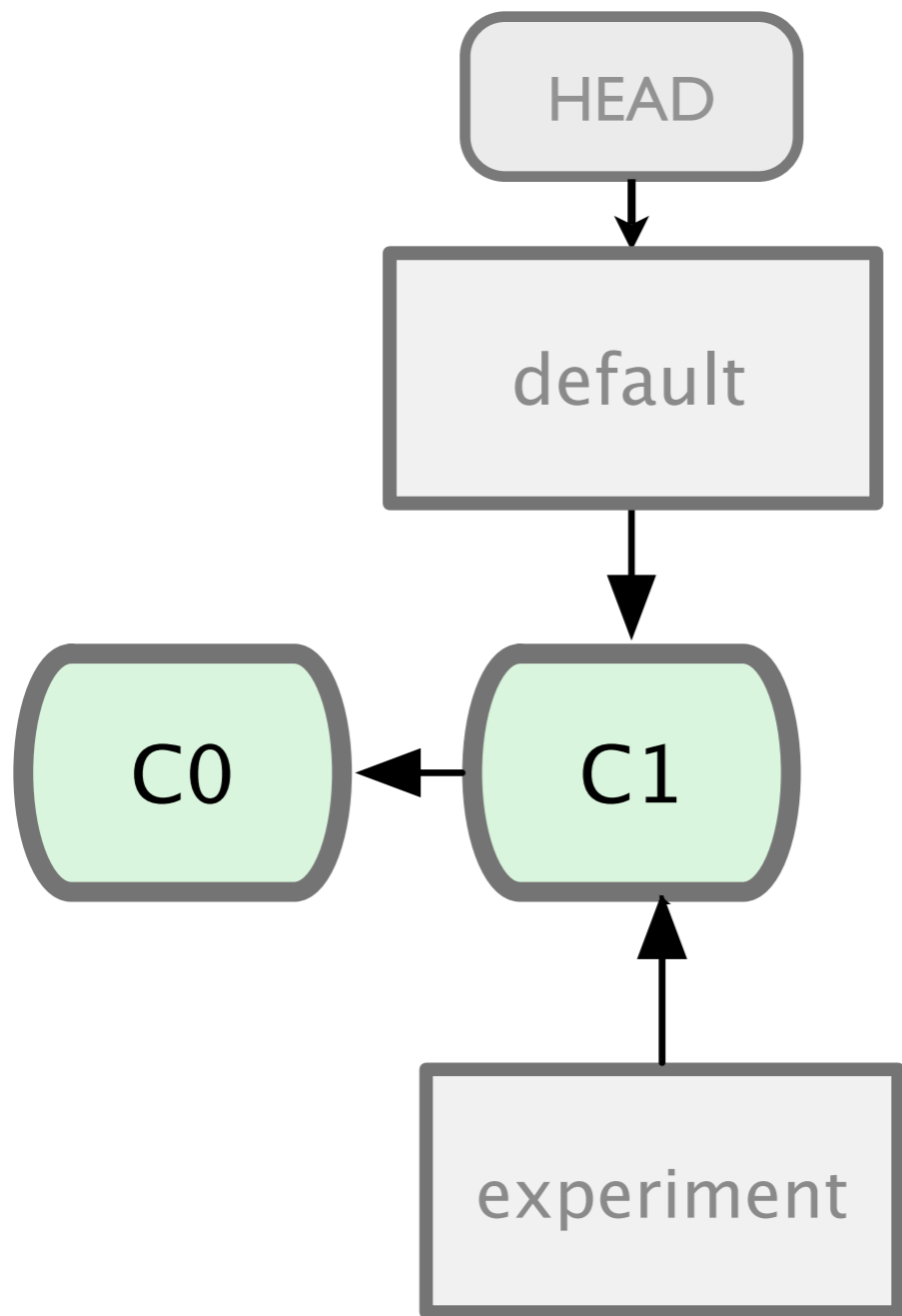




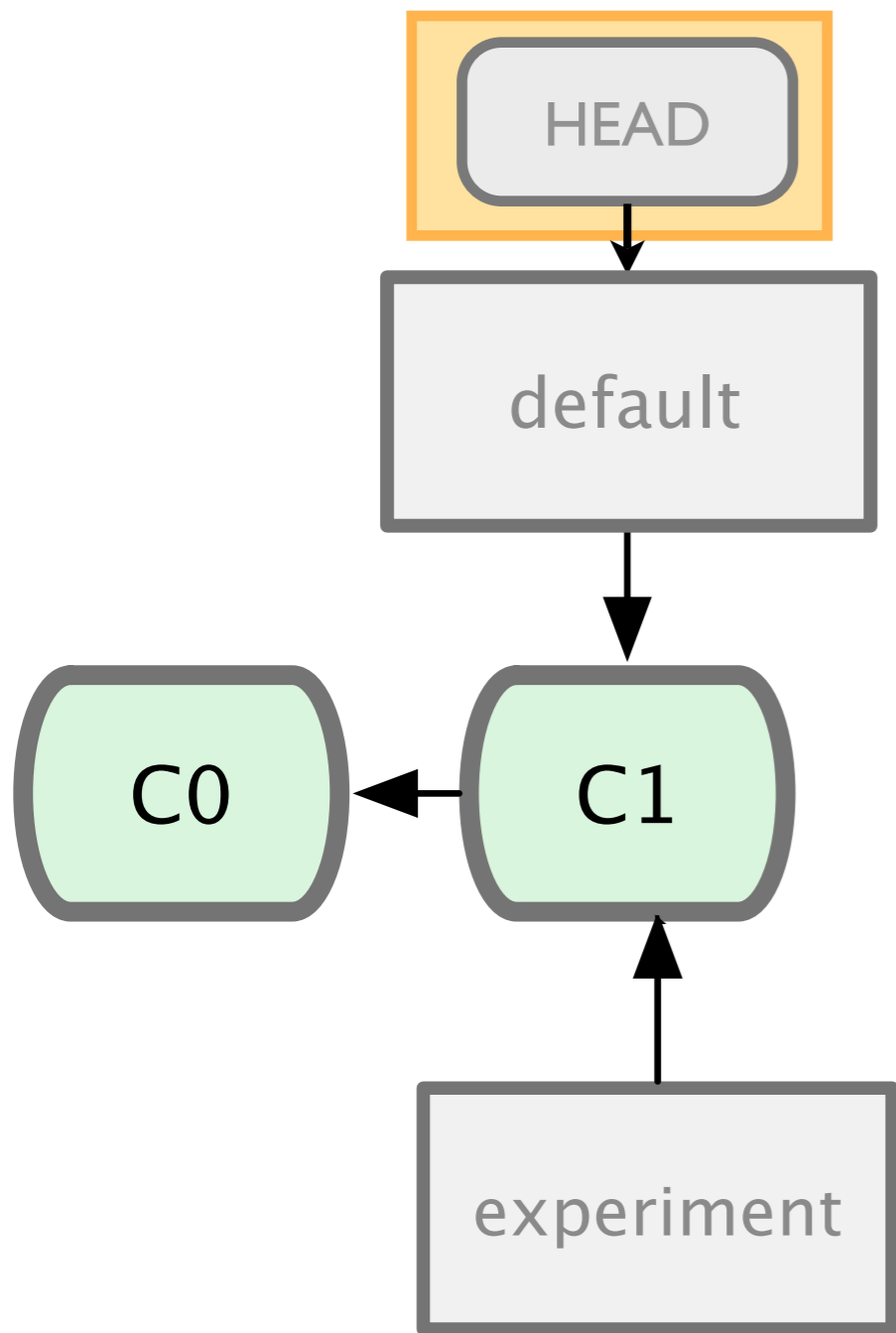
git branch experiment



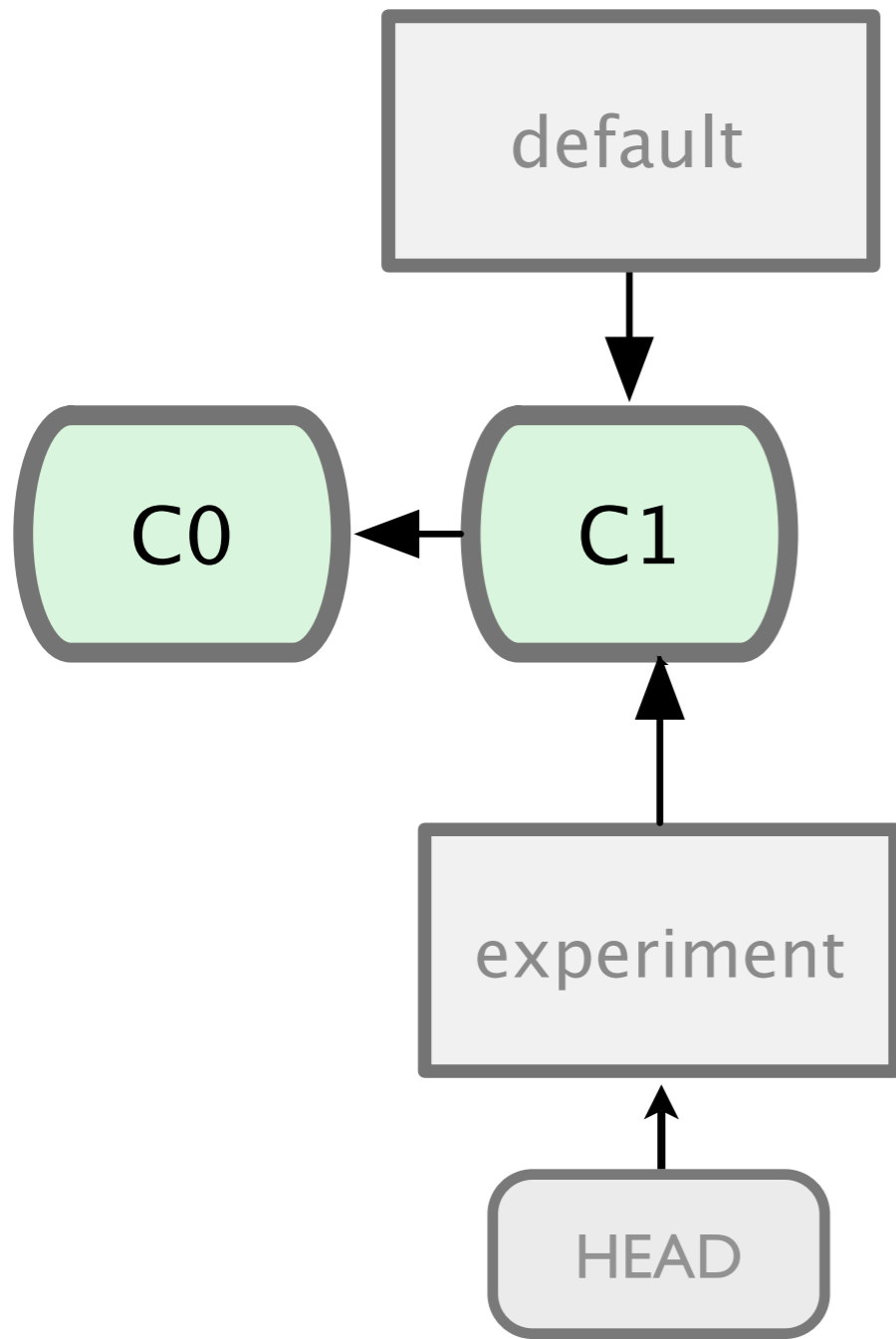
git branch experiment



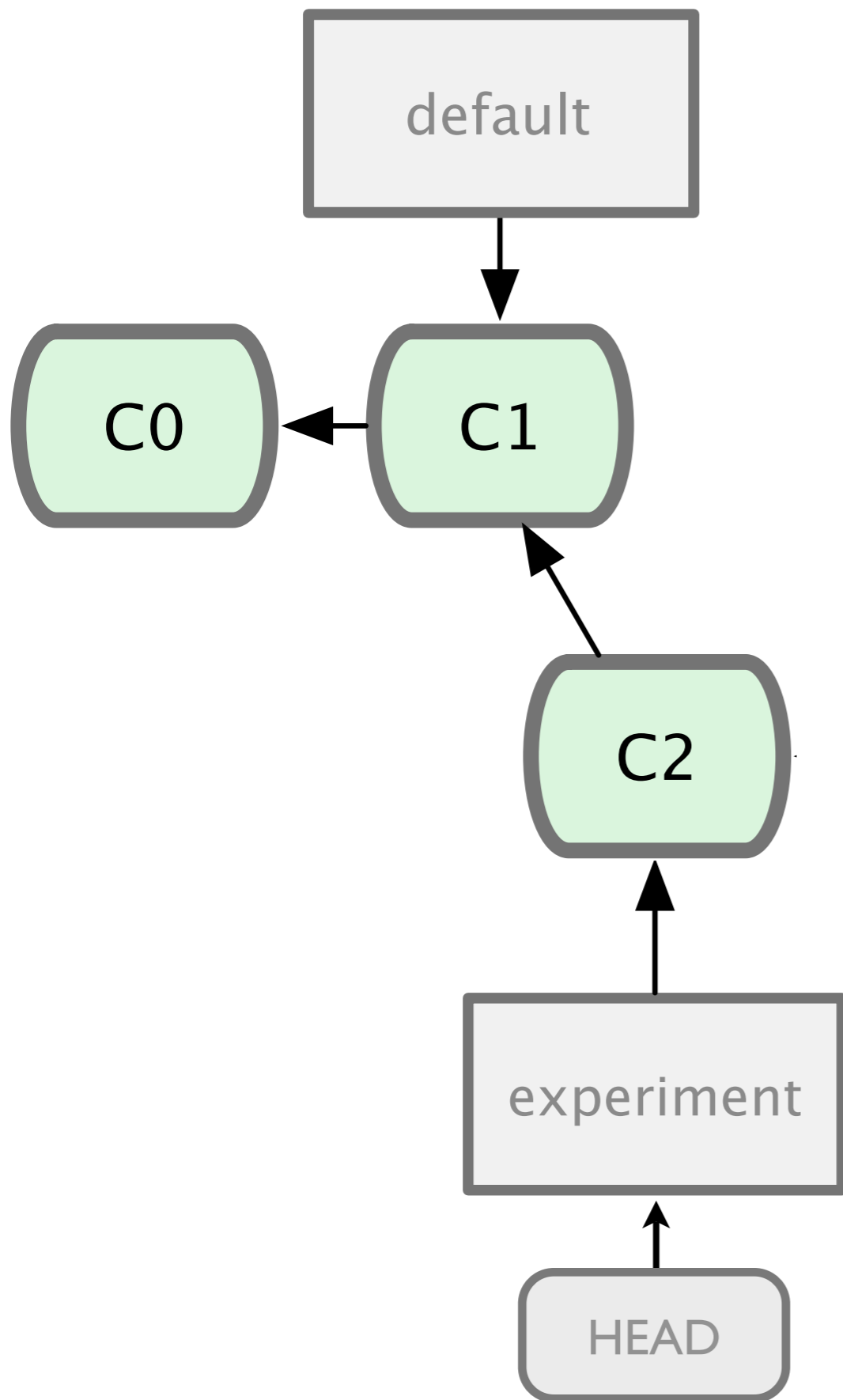
\$ git branch
* default
experiment



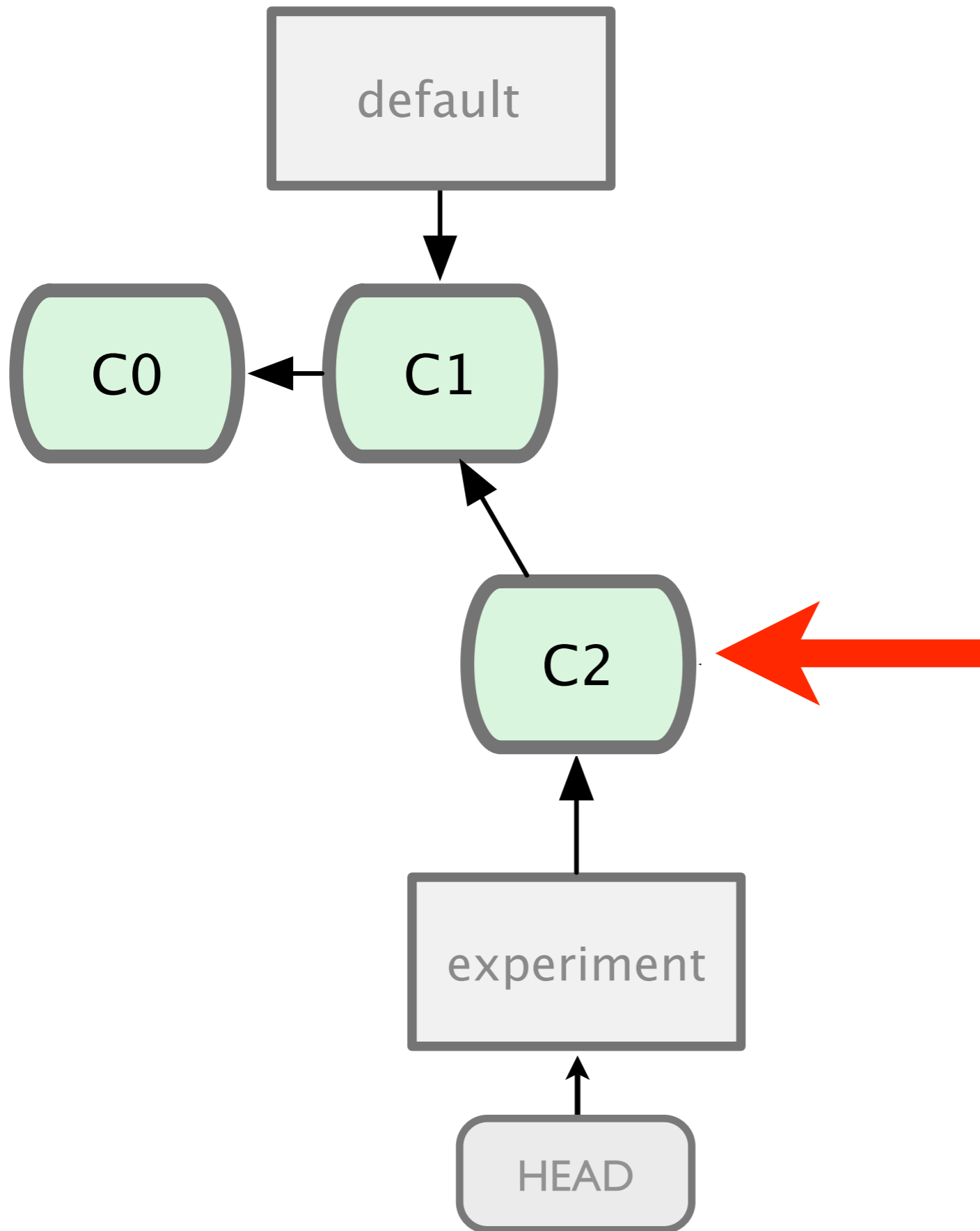
\$ git branch
* default
experiment



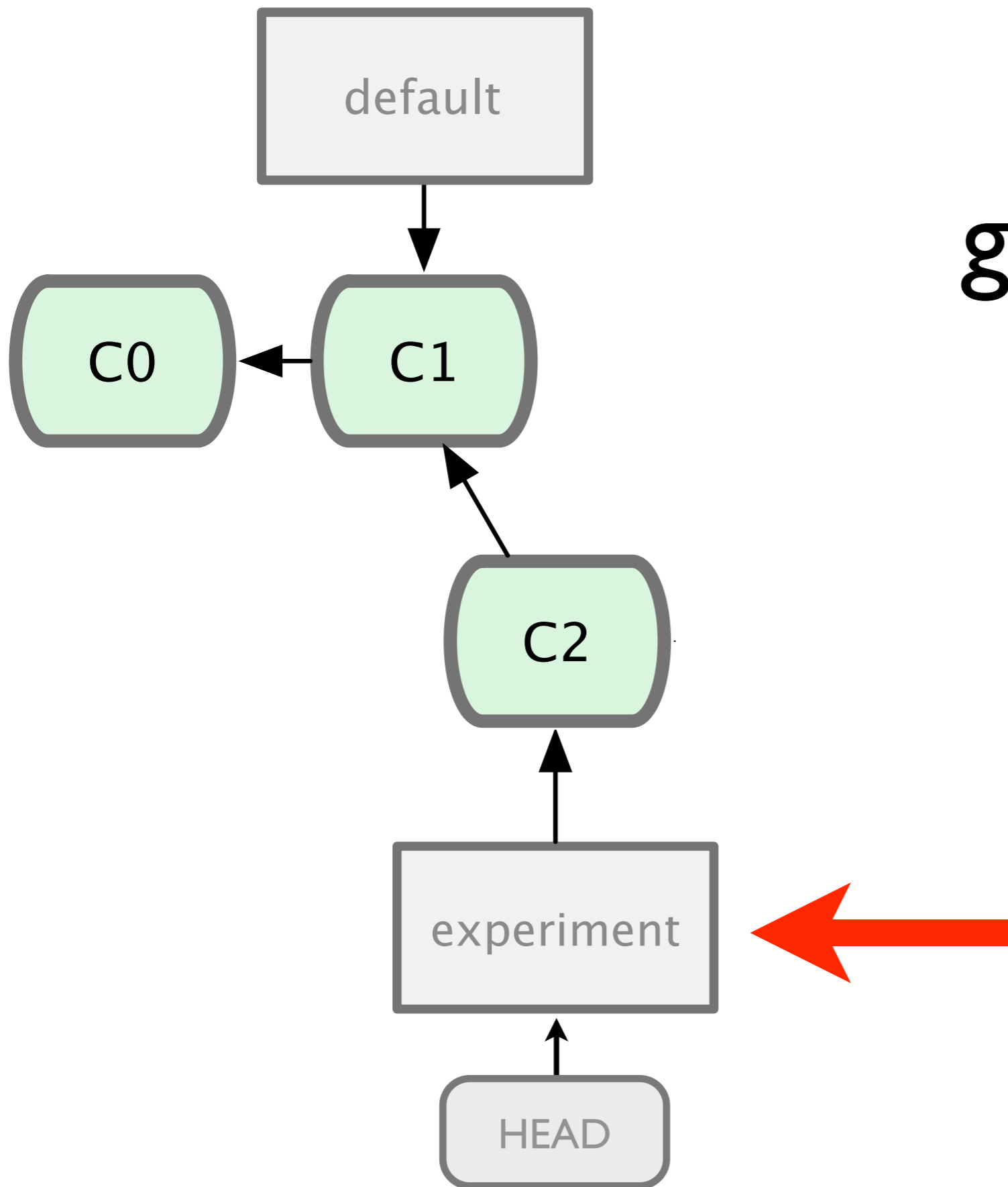
git checkout experiment



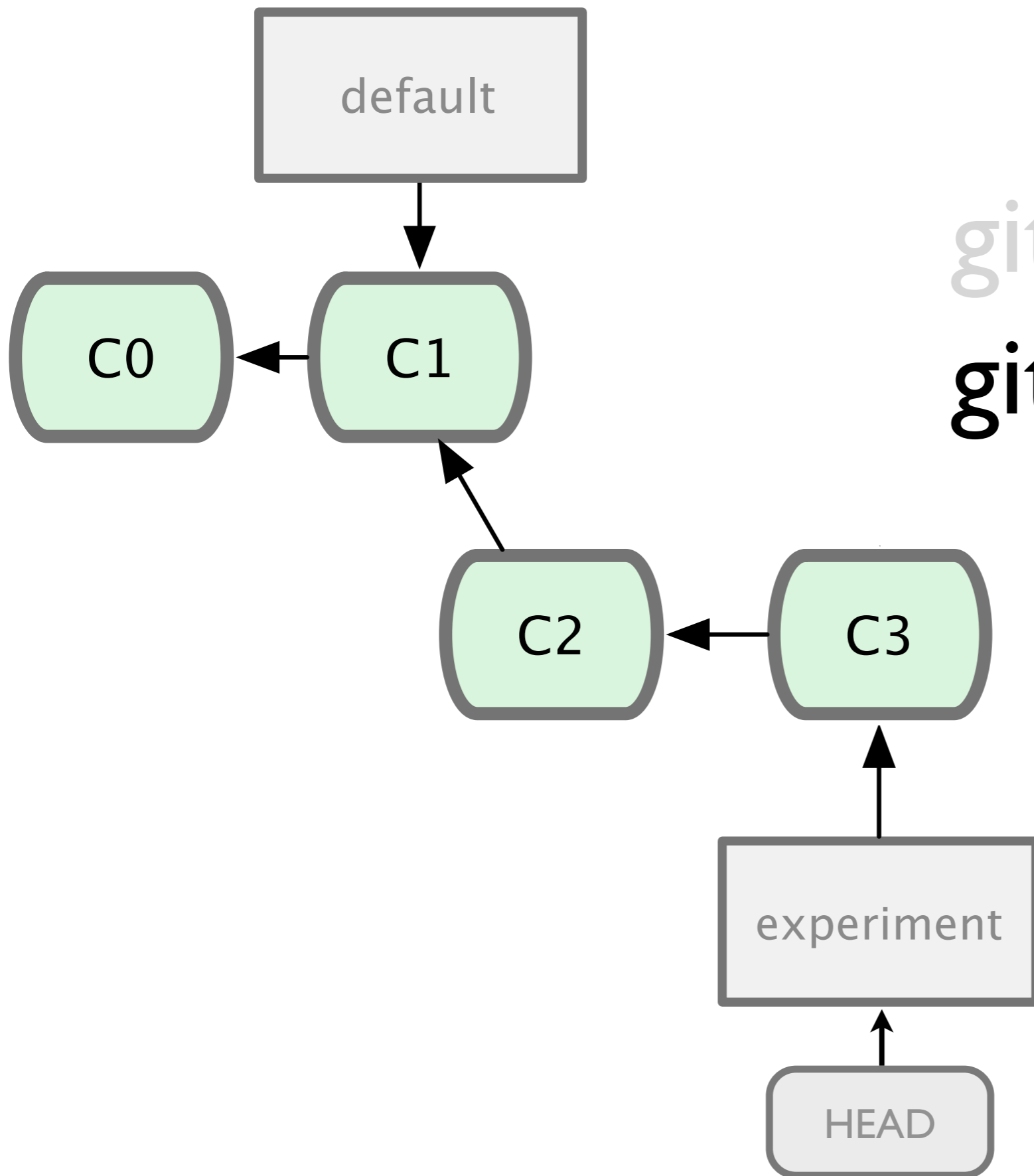
git commit



git commit

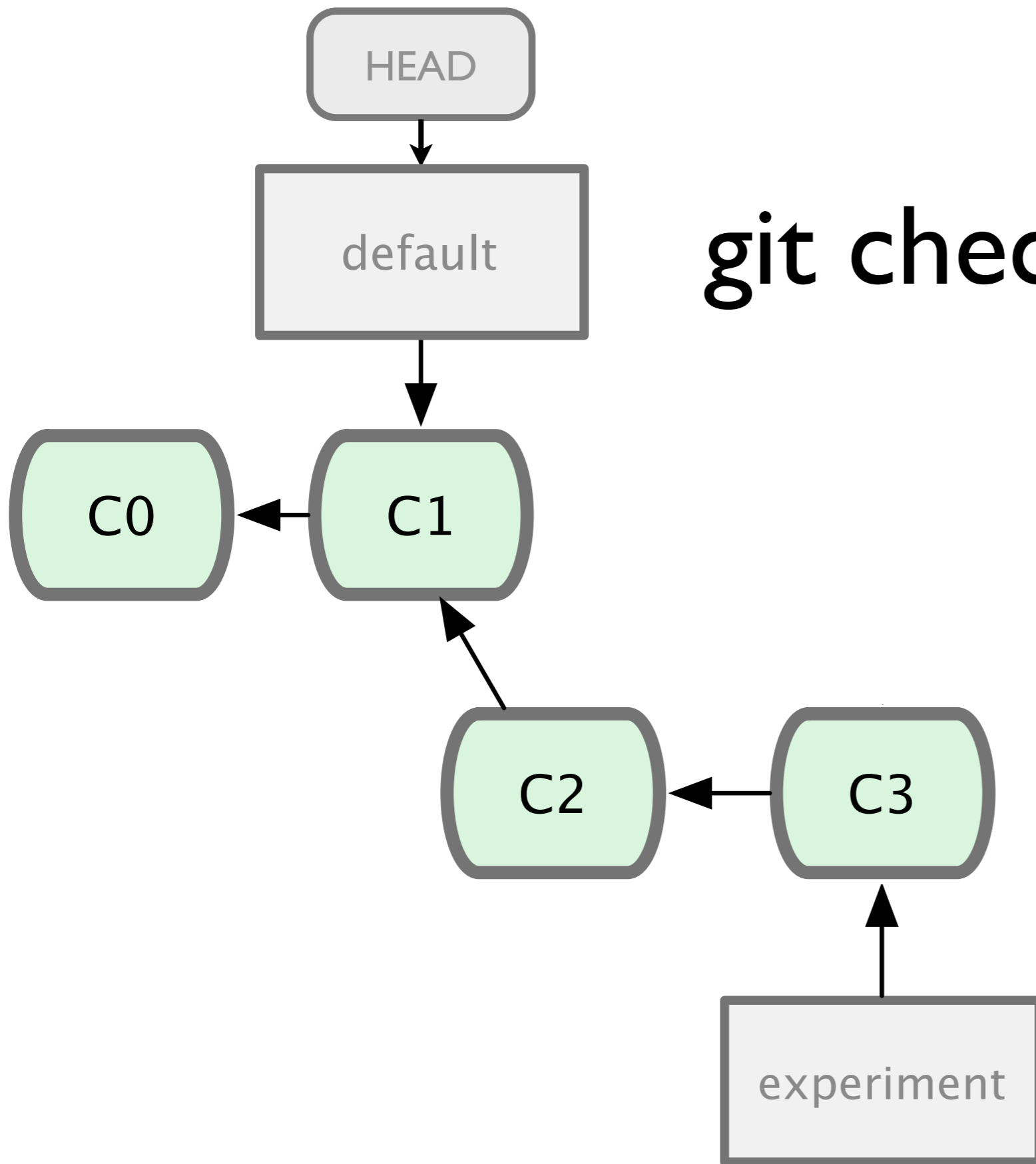


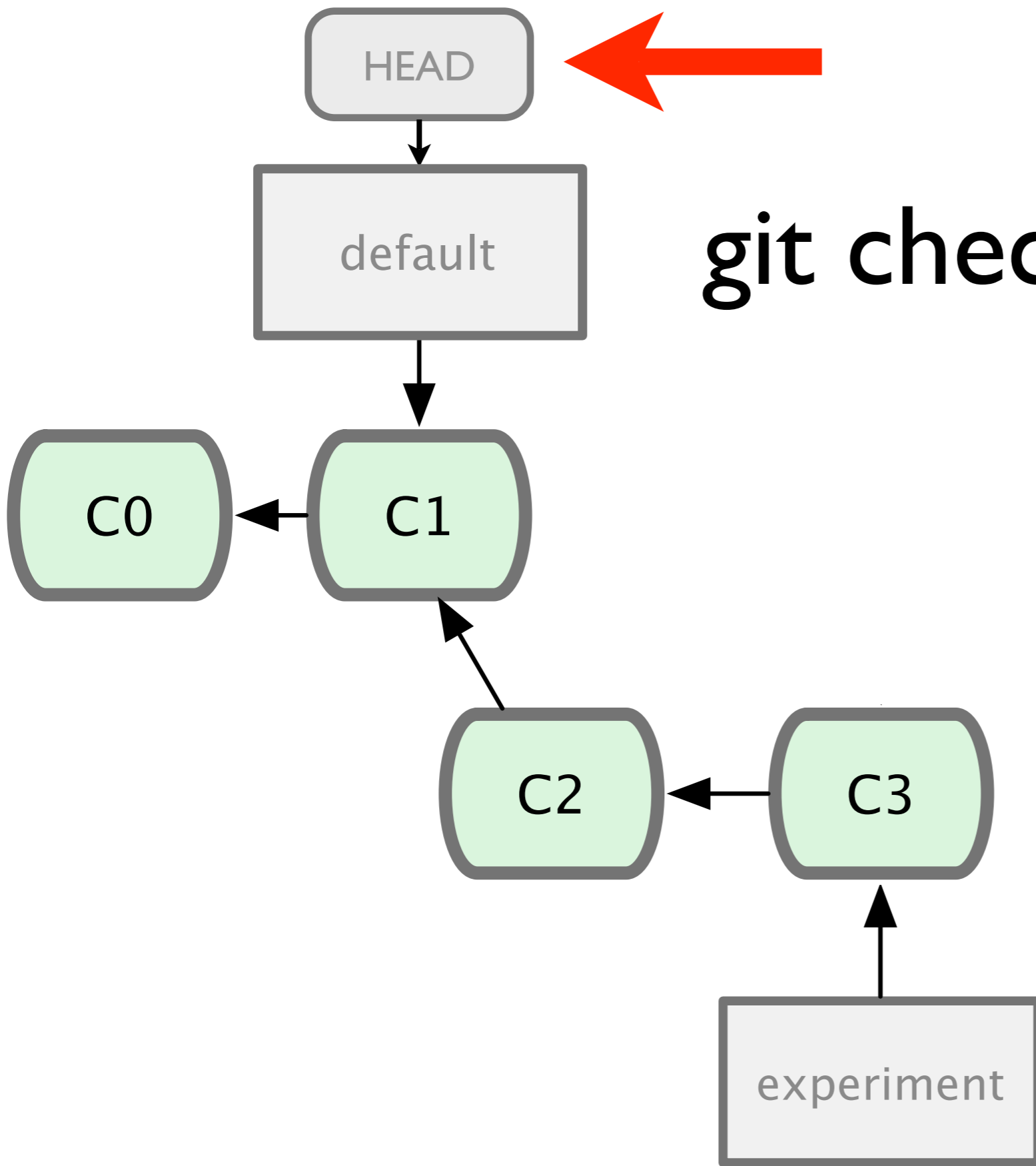
git commit



git commit
git commit

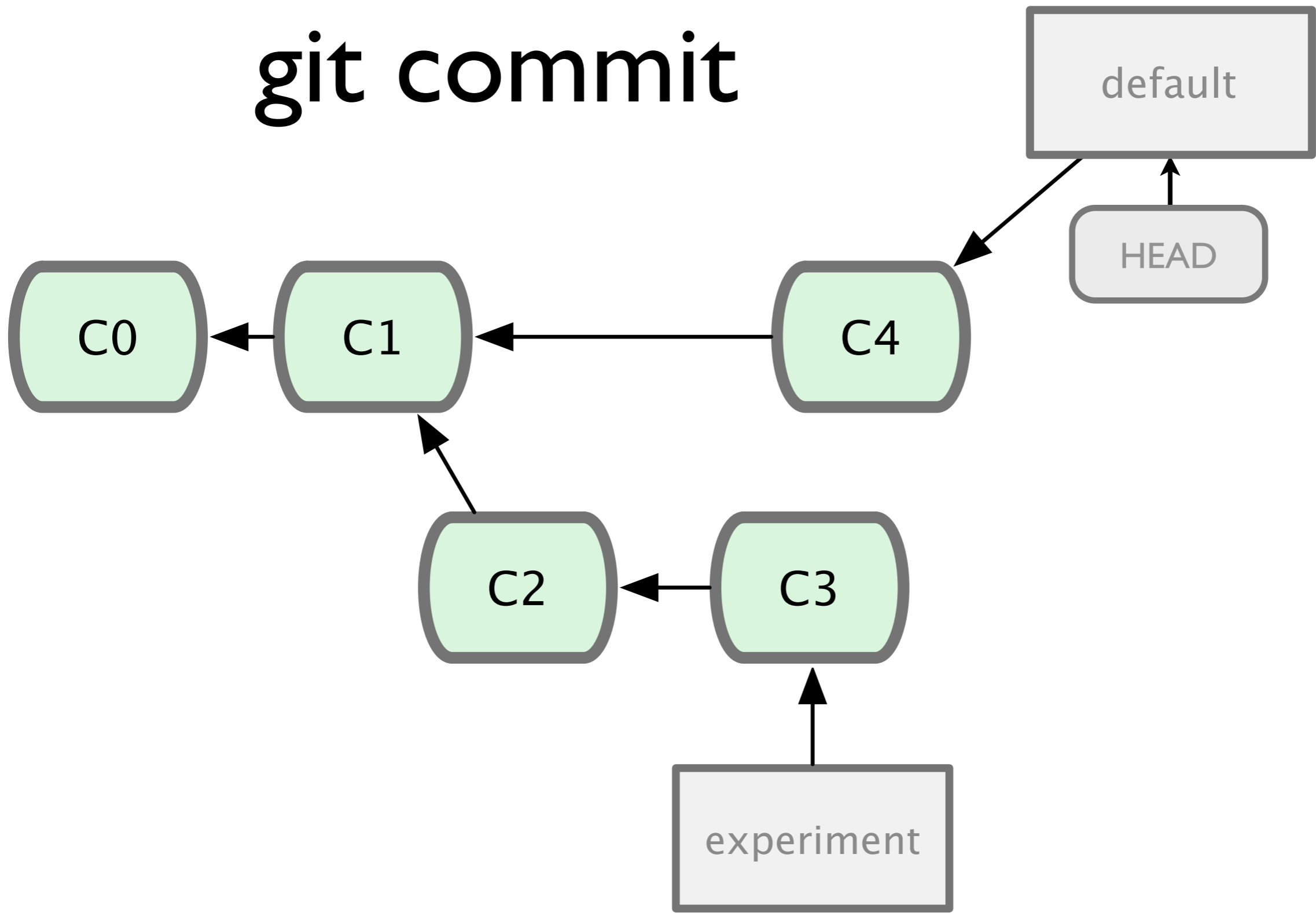
git checkout default



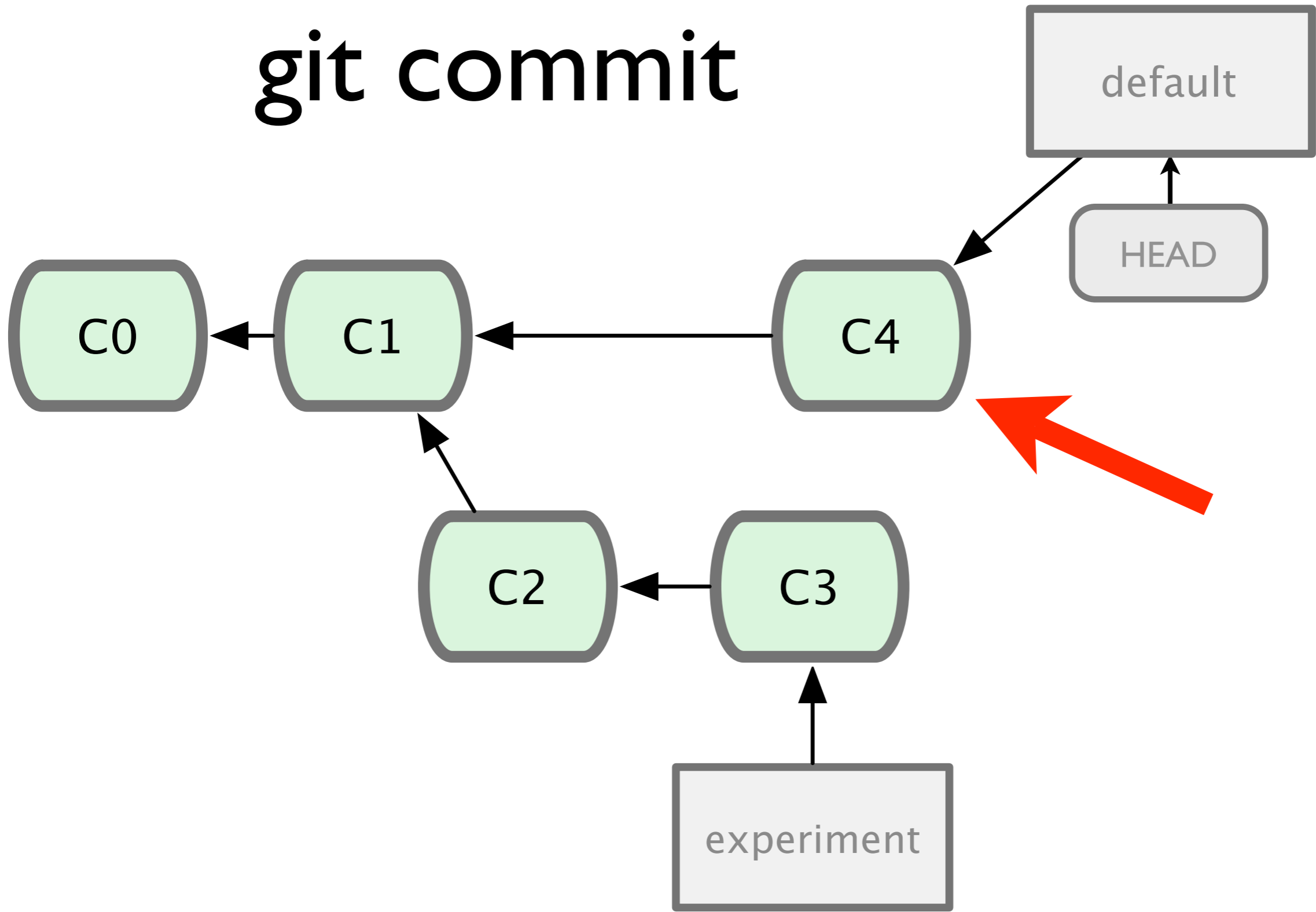


git checkout default

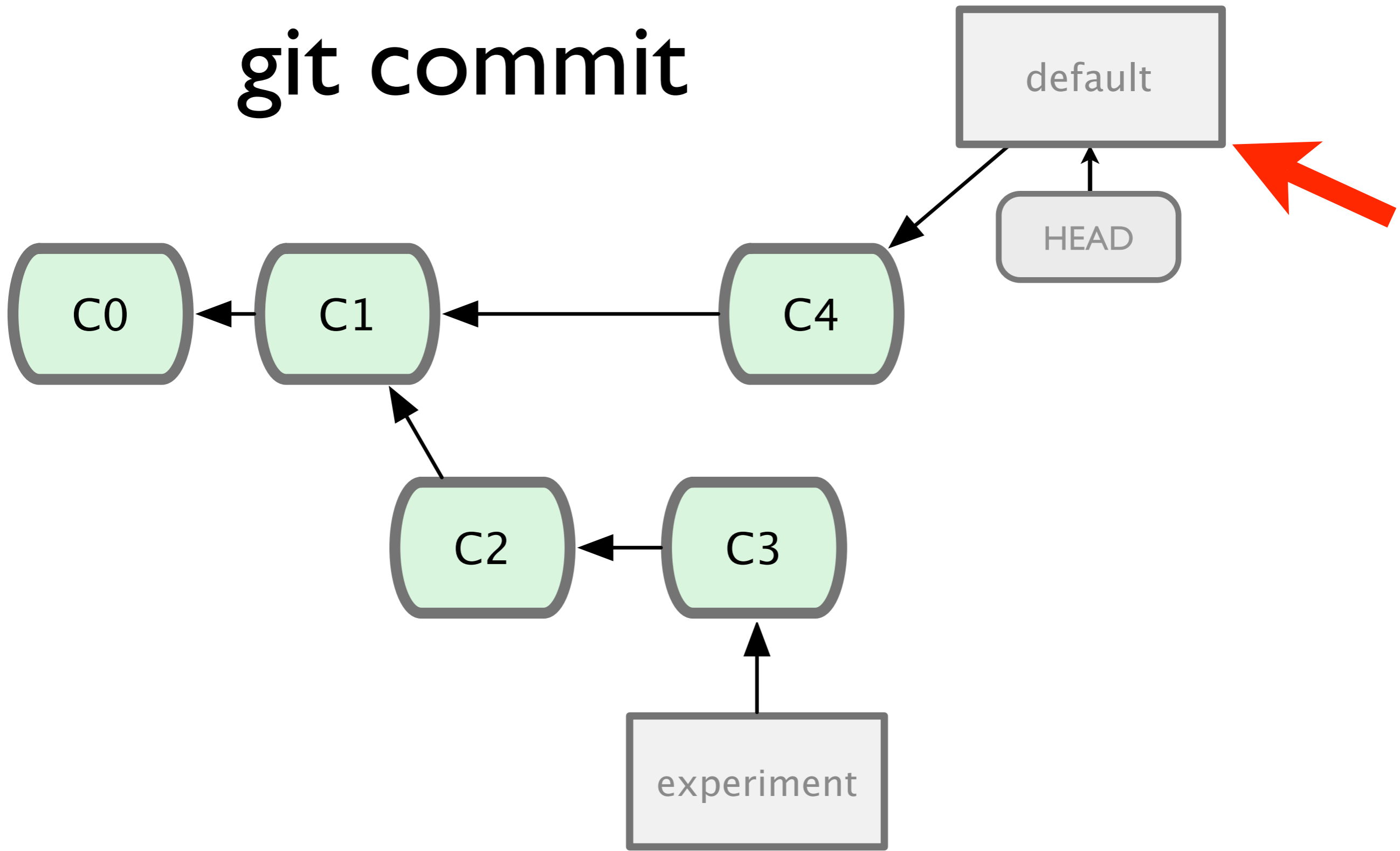
git commit

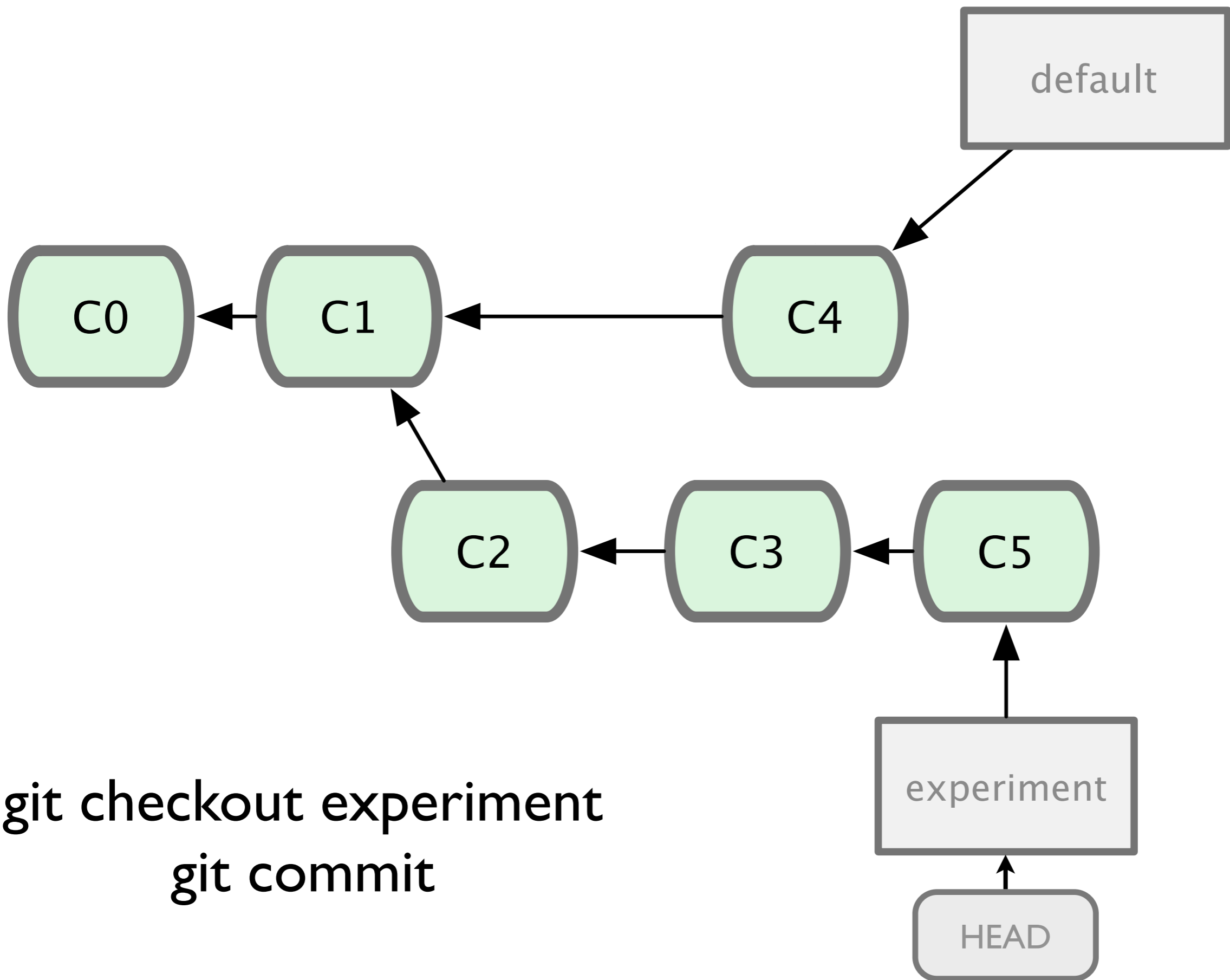


git commit



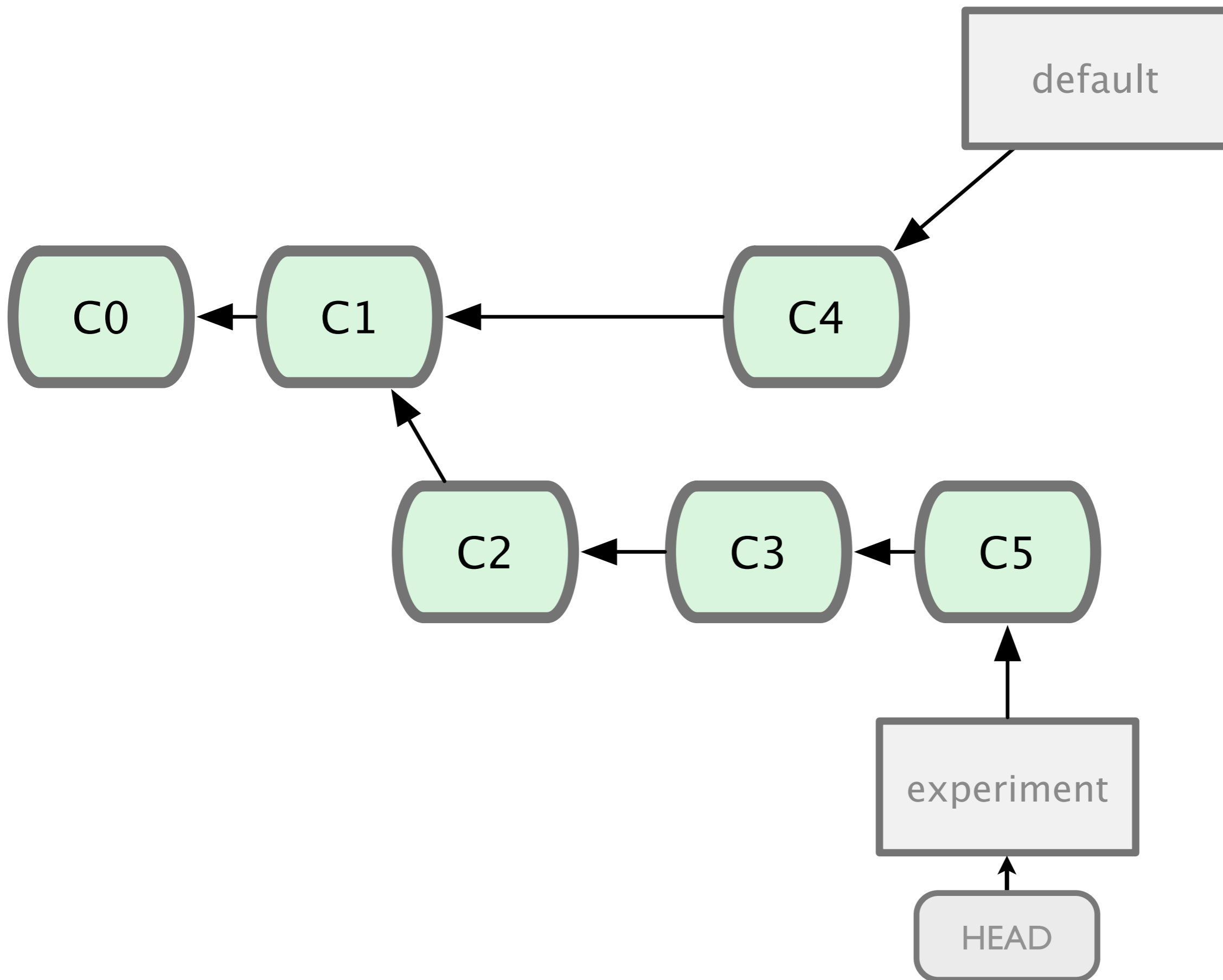
git commit

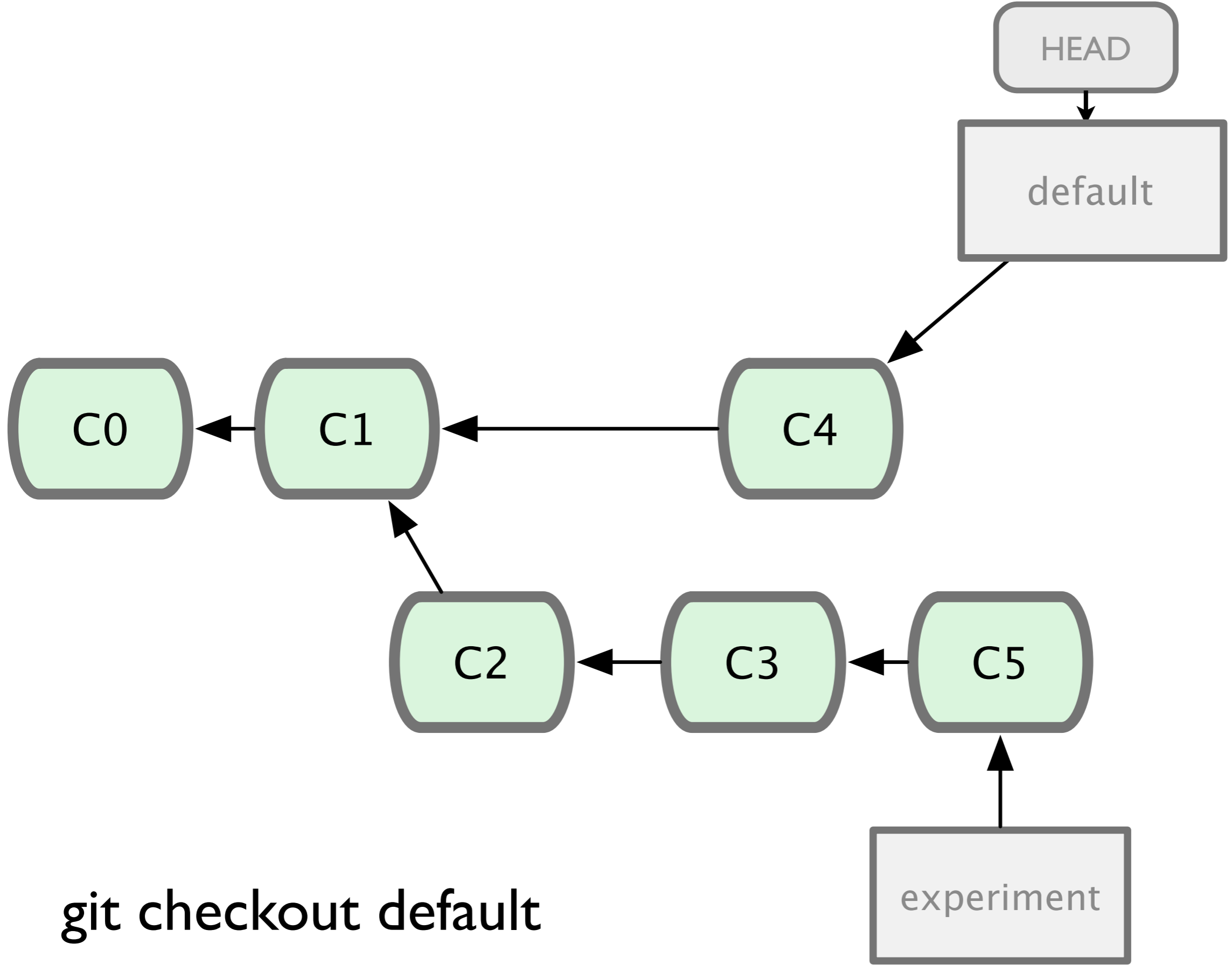




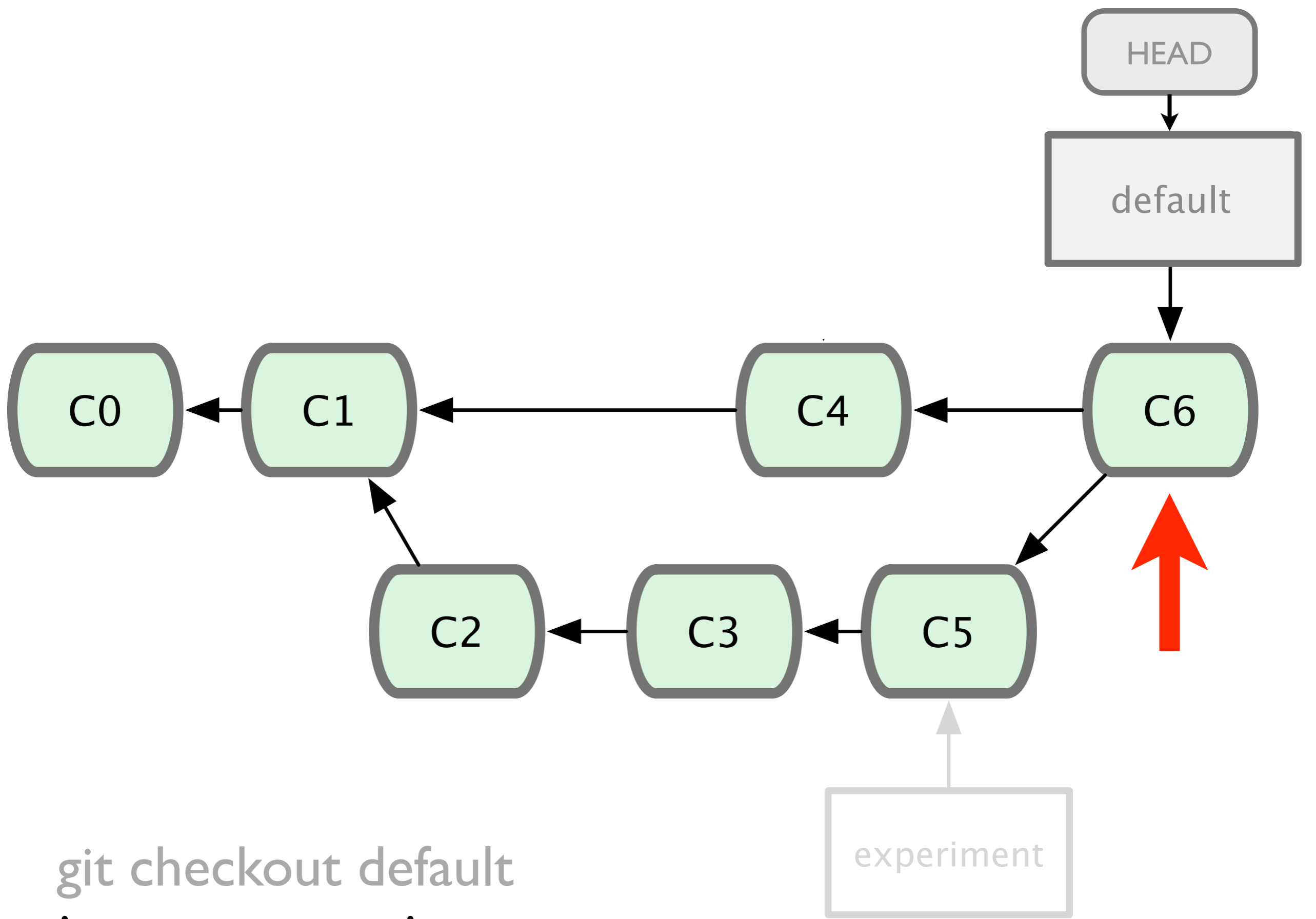
merging

git merge

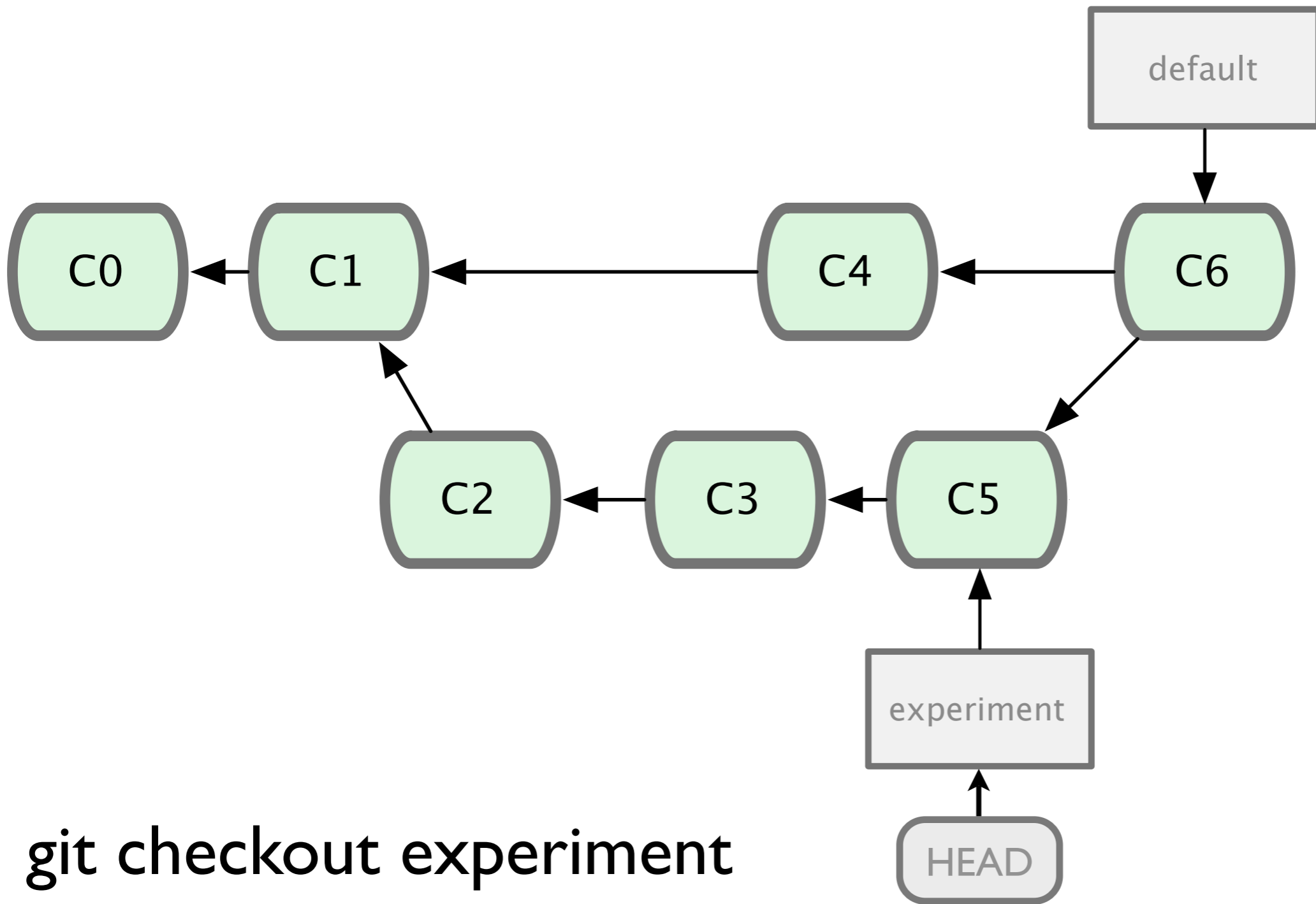


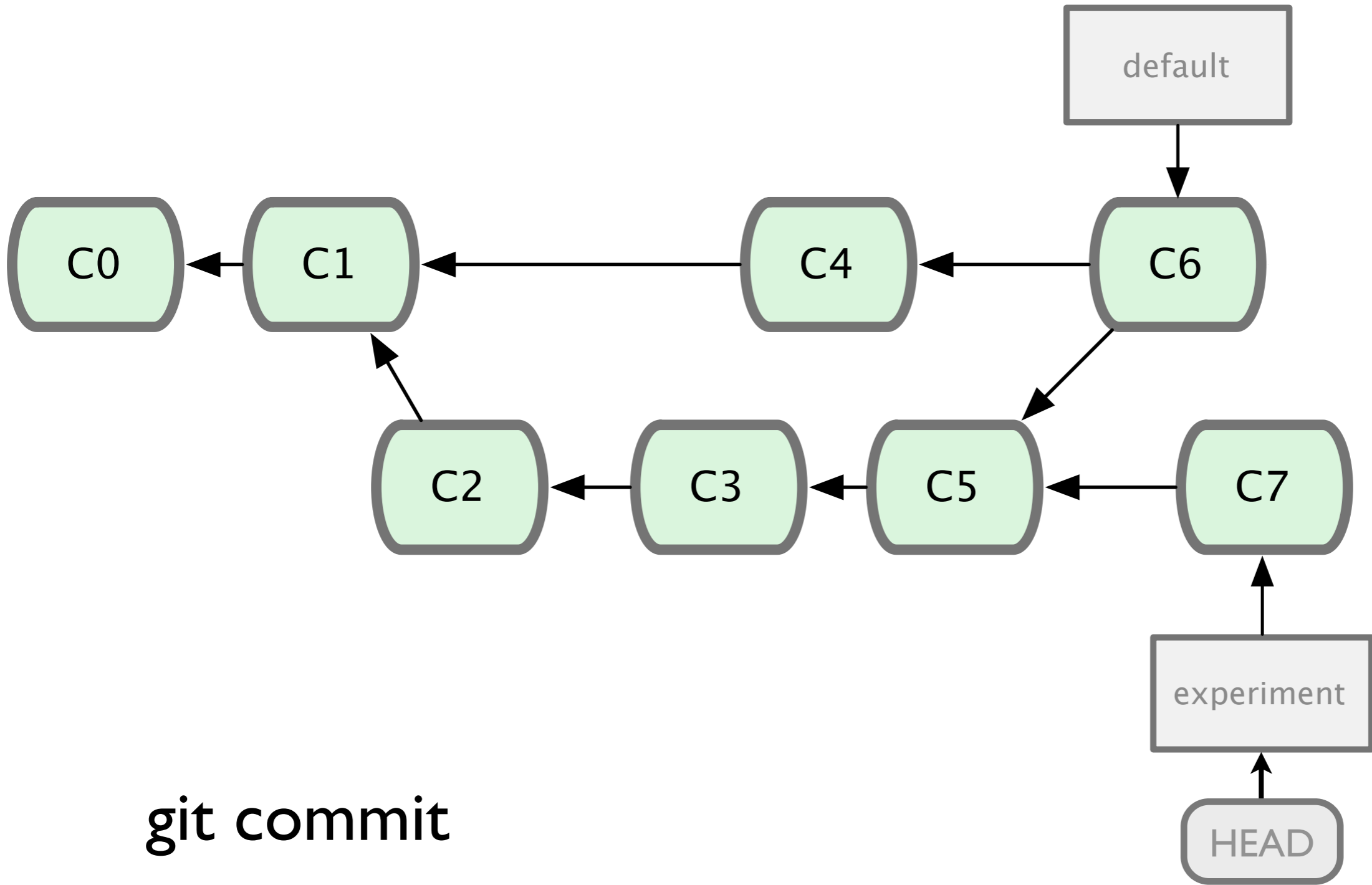


git checkout default

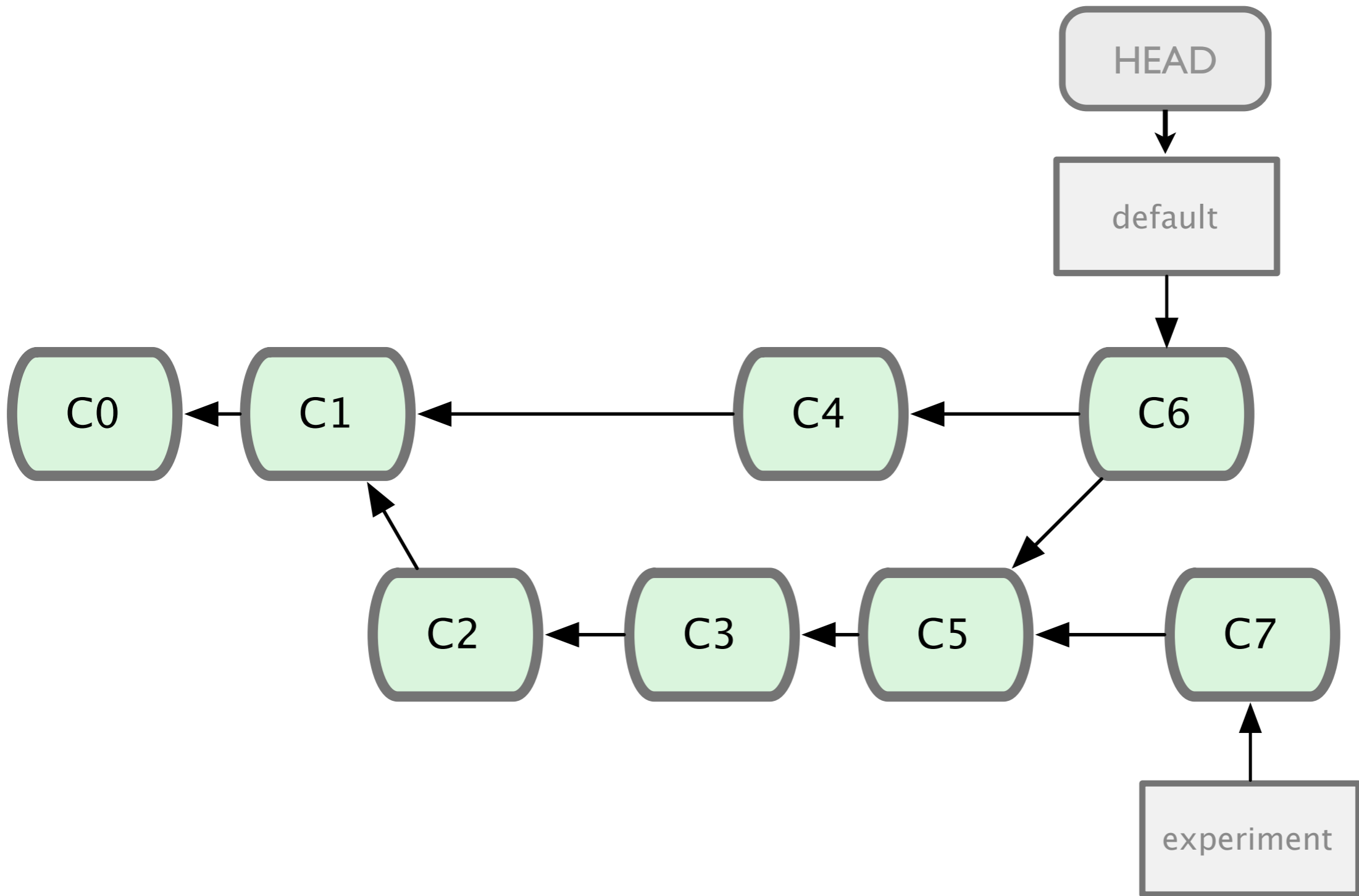


git checkout default
git merge experiment

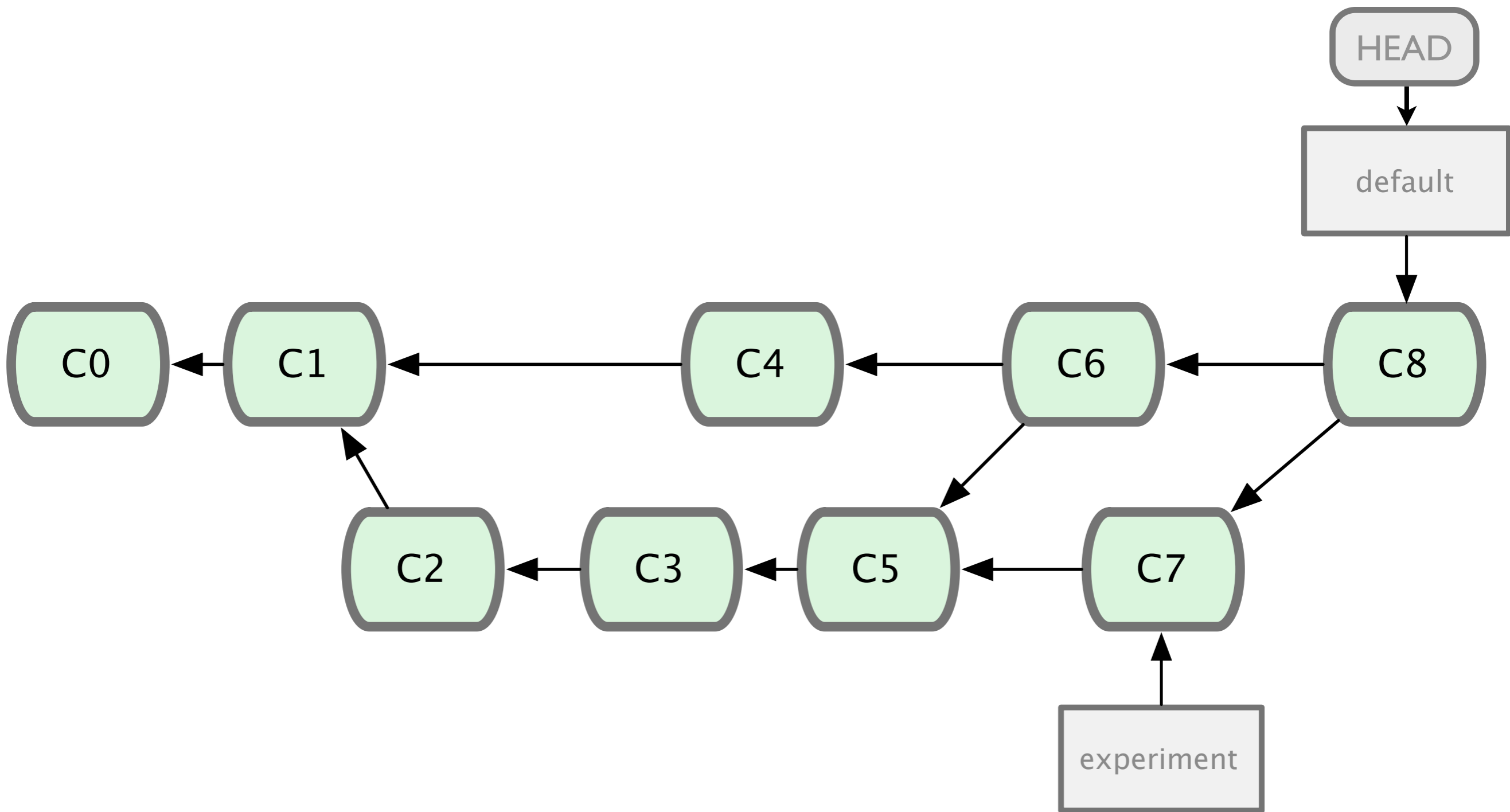




git commit

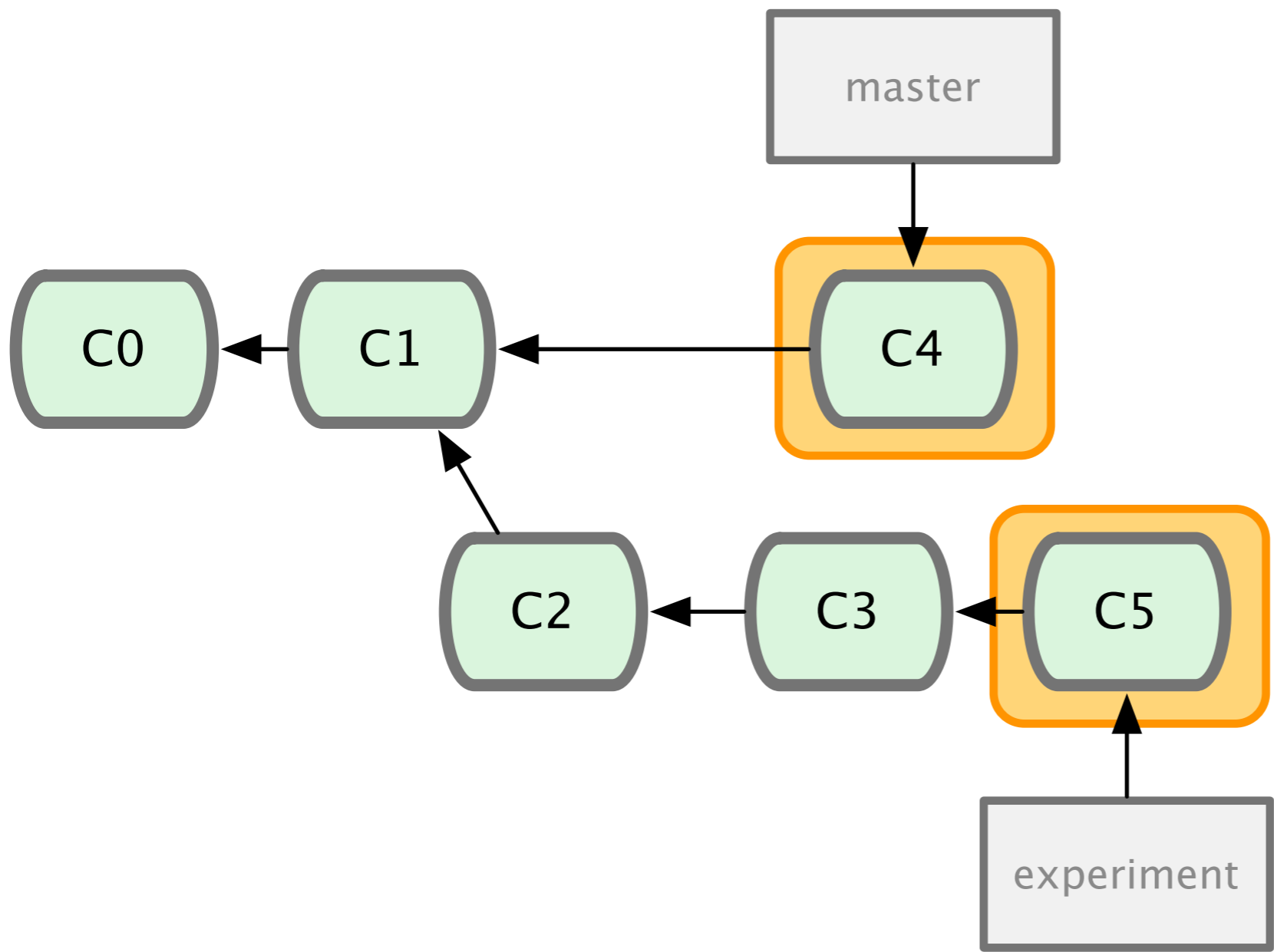


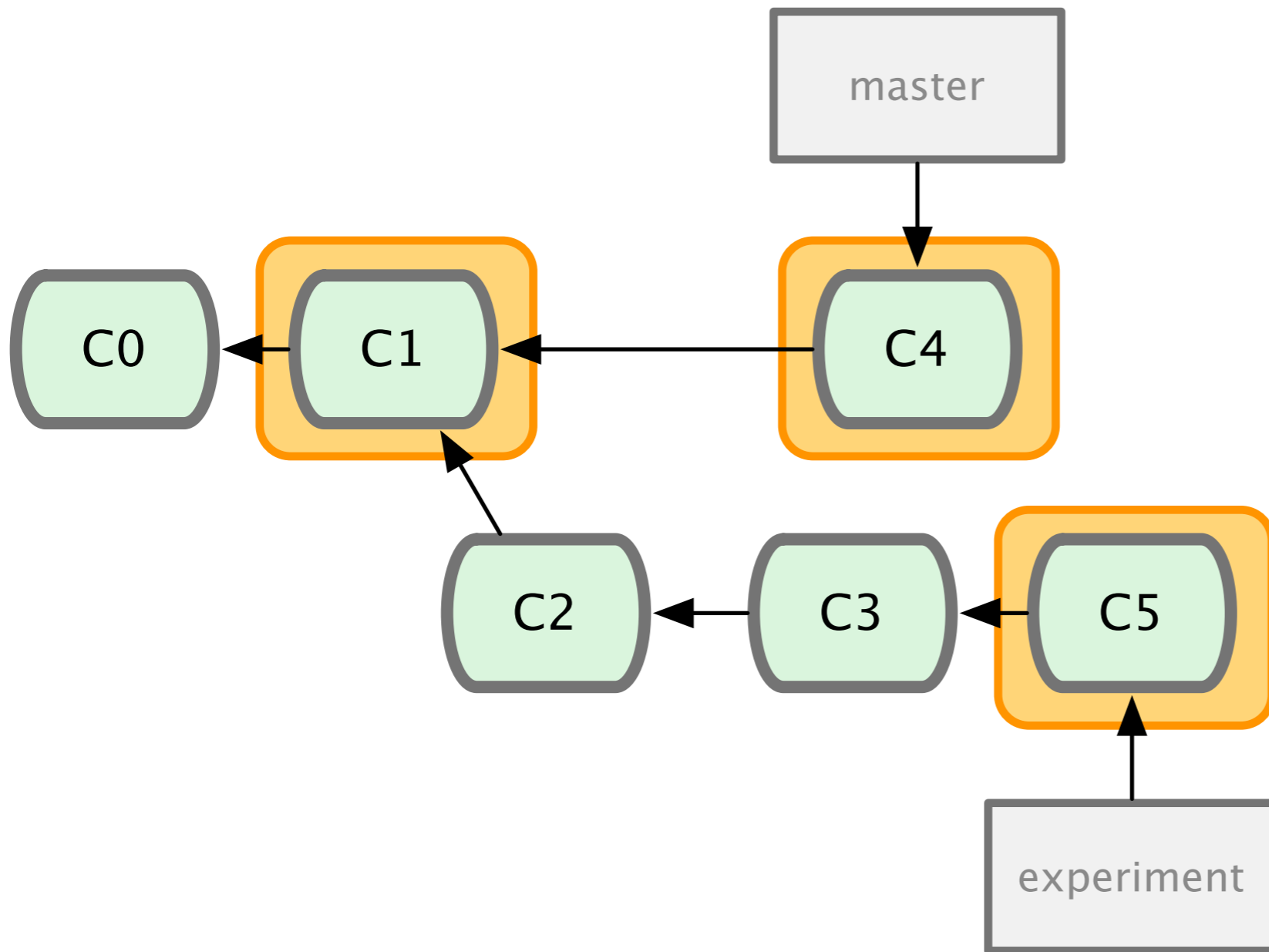
git checkout default

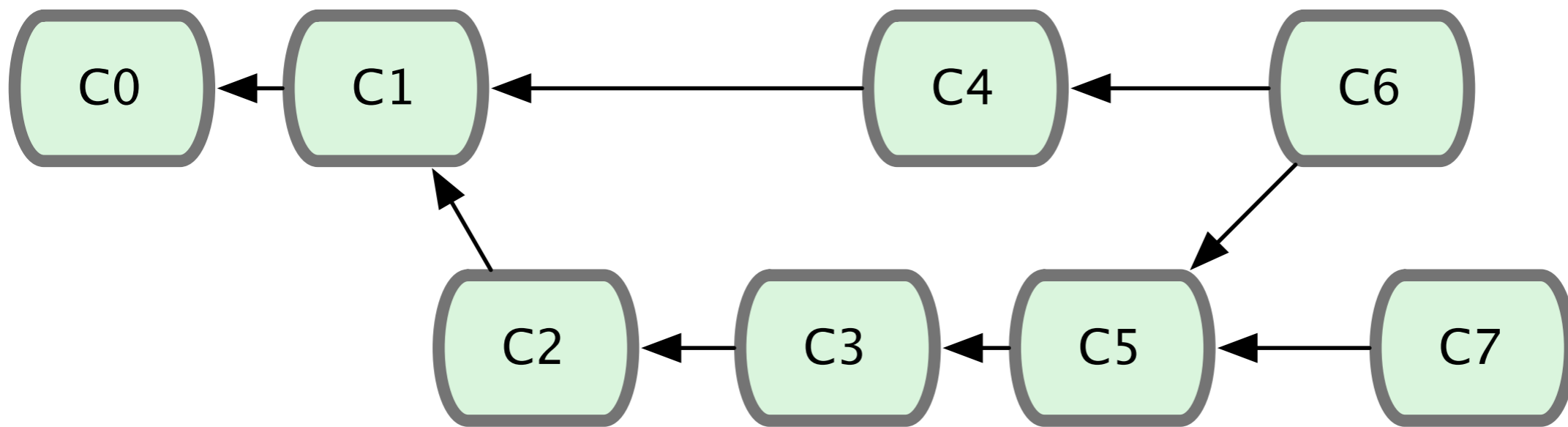


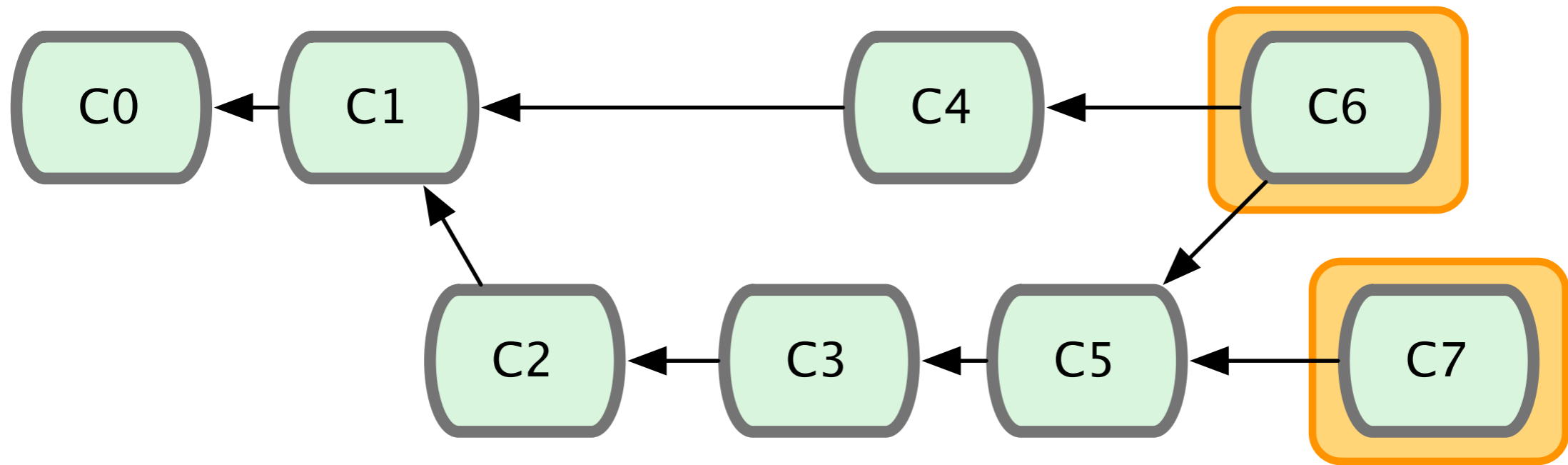
git merge experiment

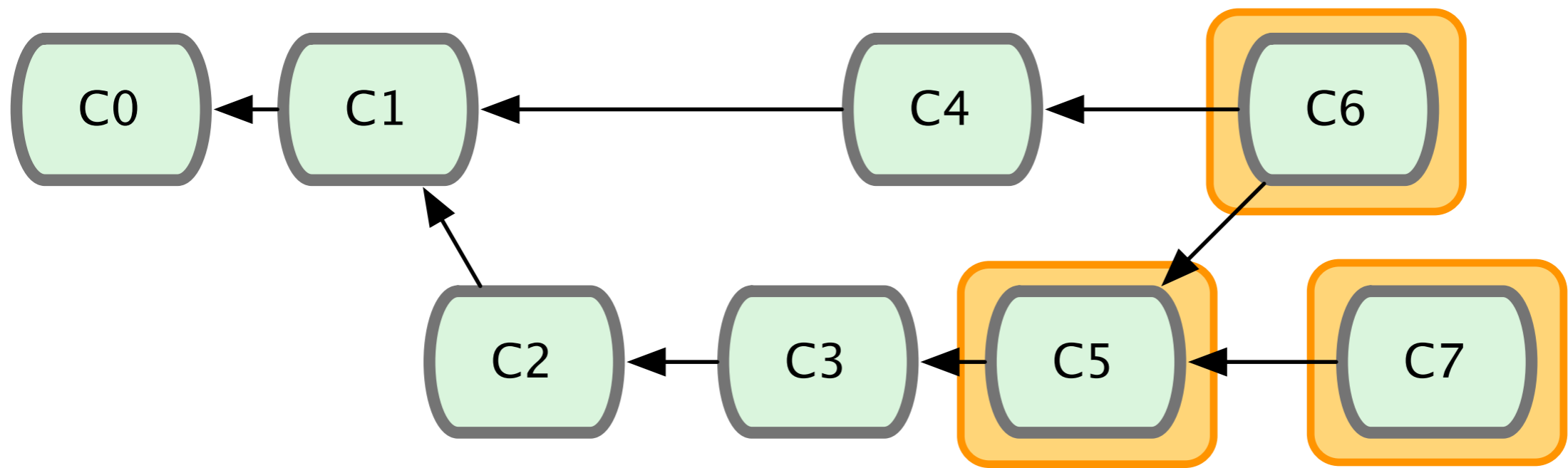
Hot 3 Way Merge Action





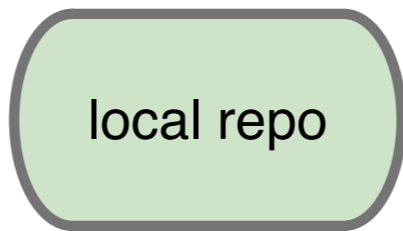
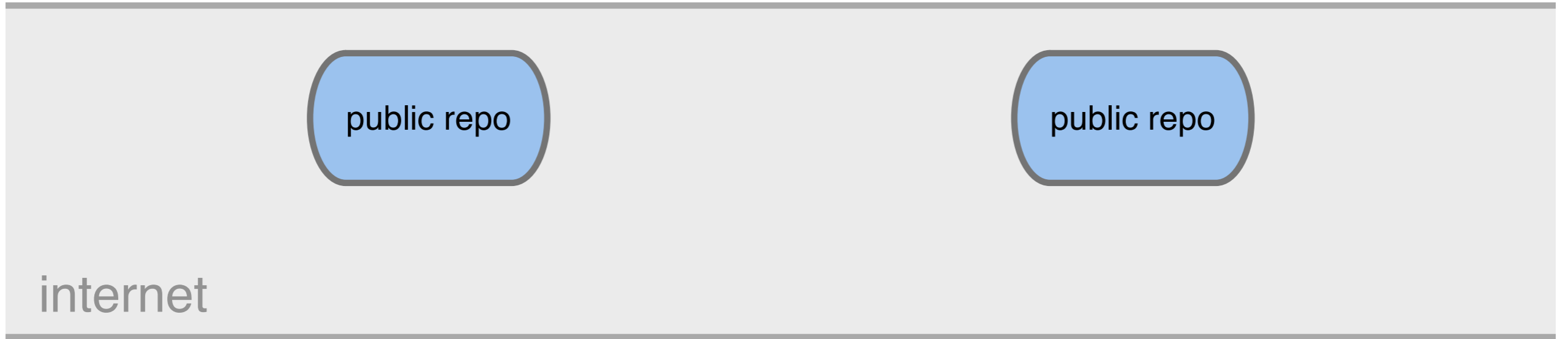


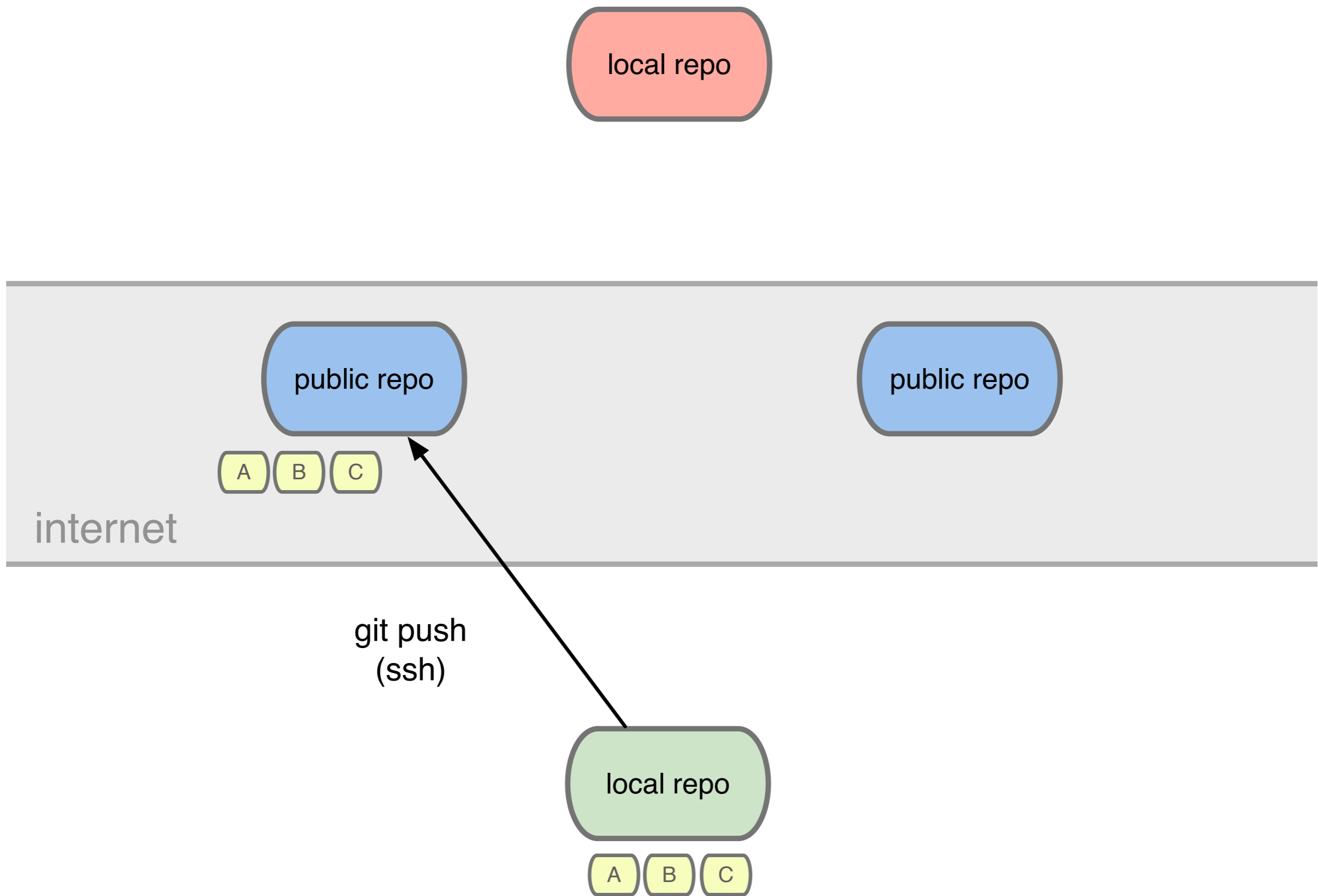


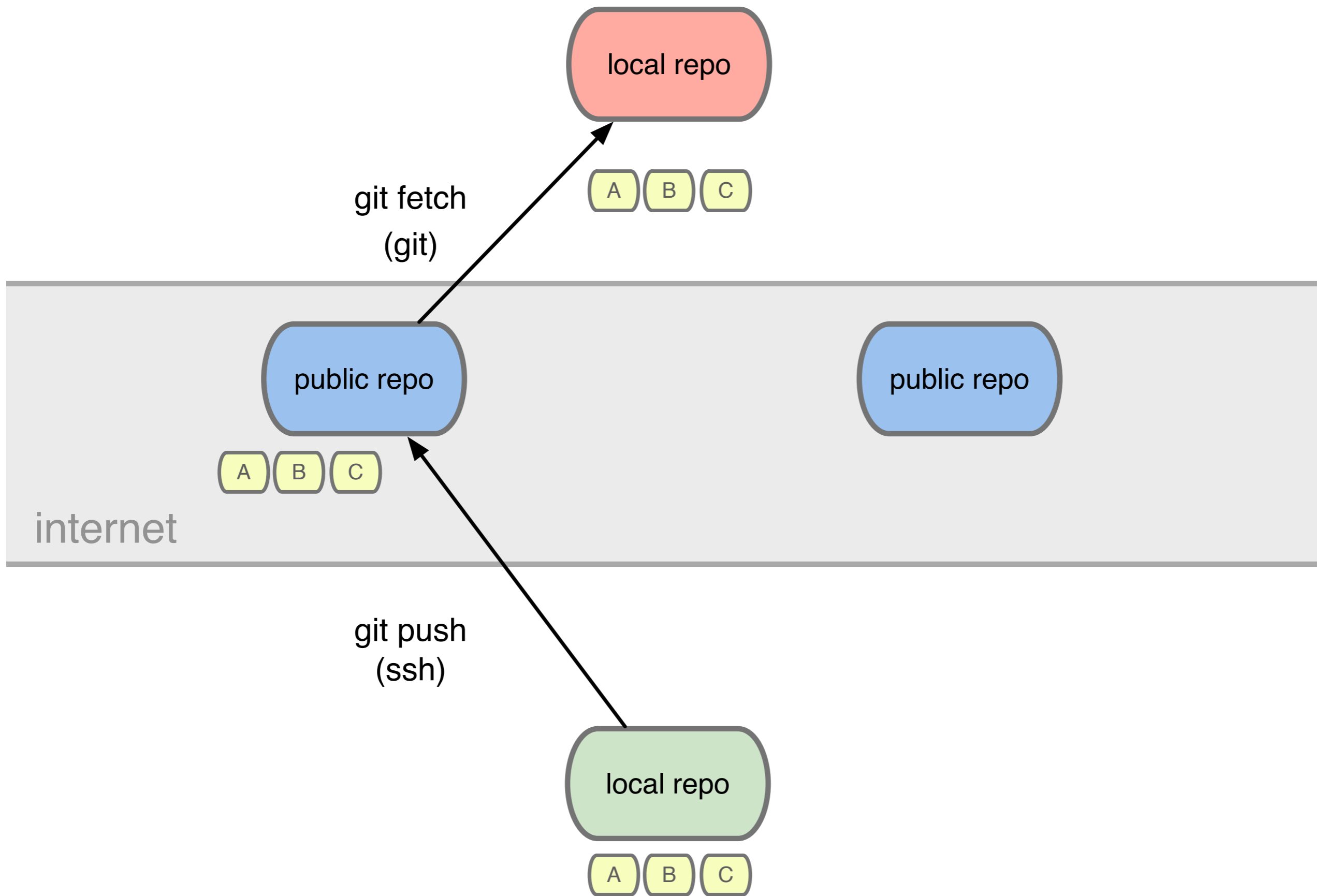


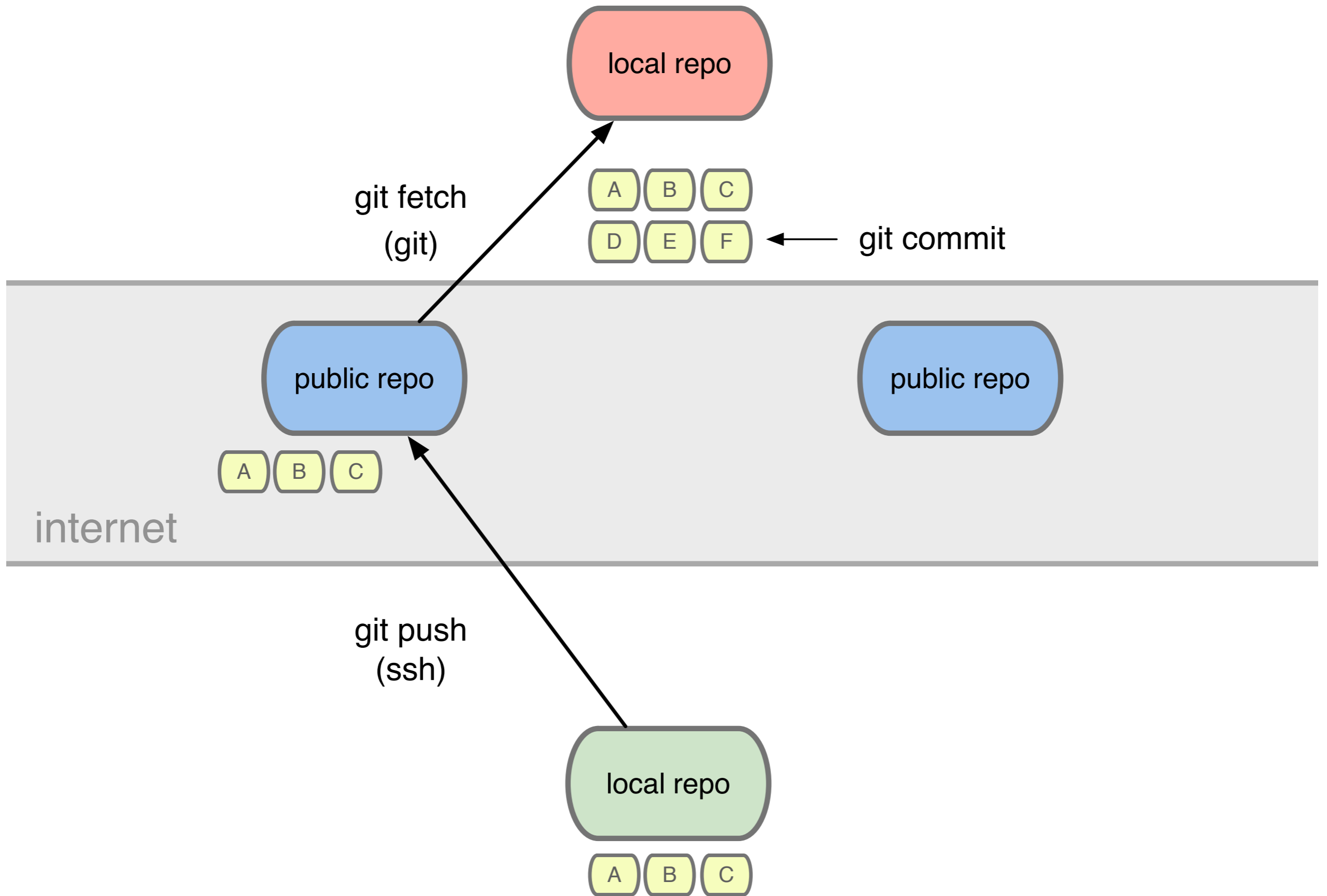
Remotes

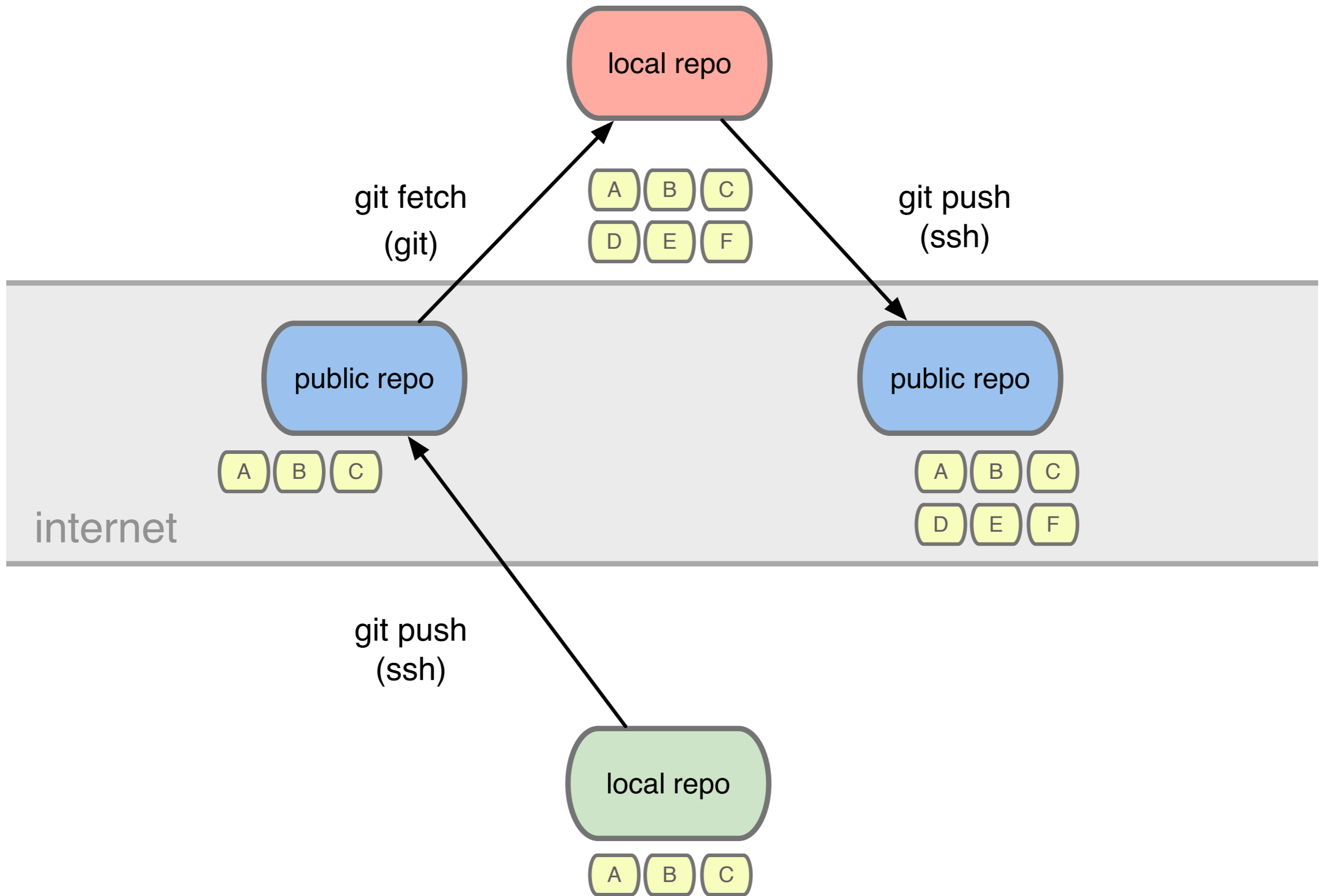
Distributed Workflow

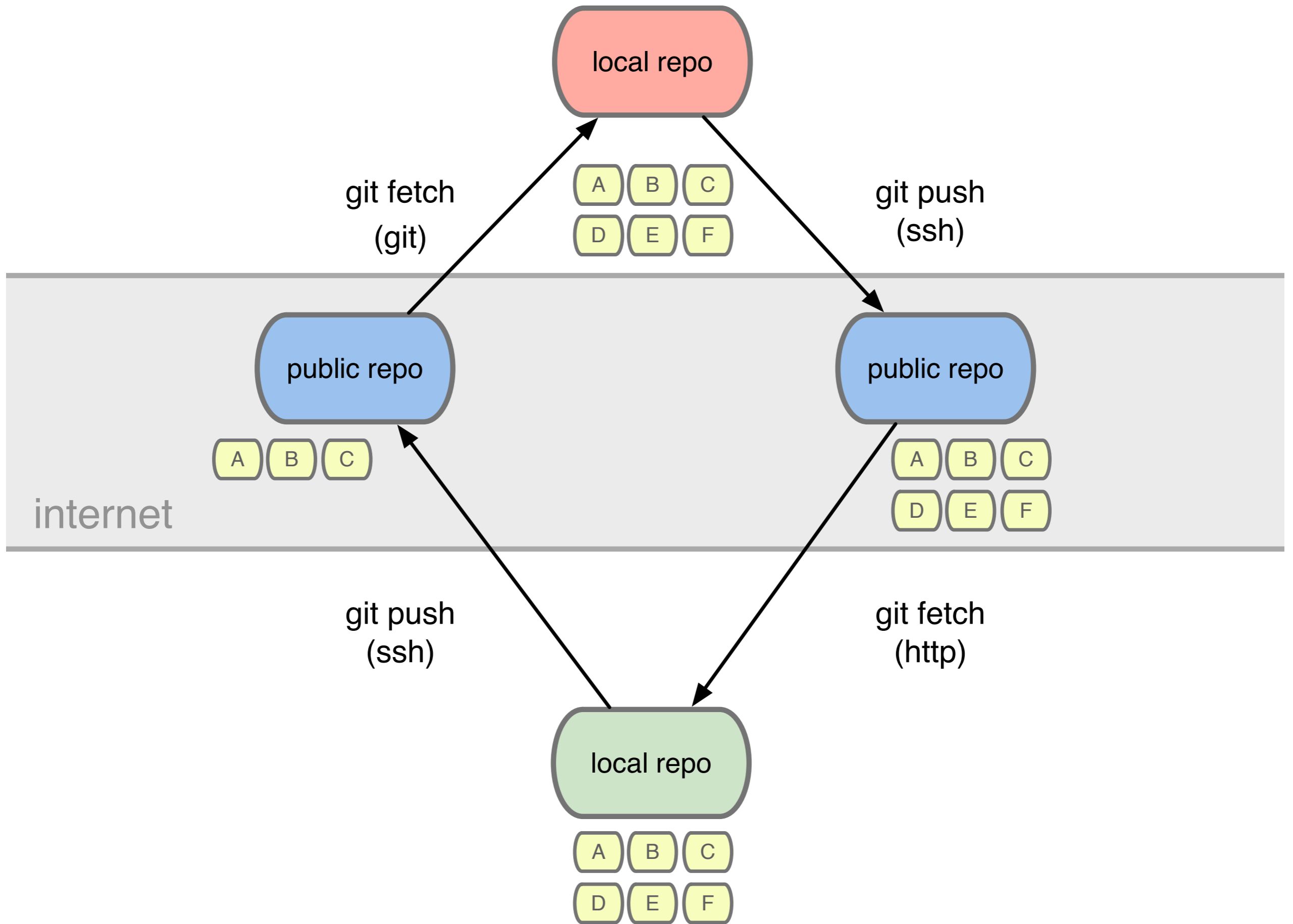


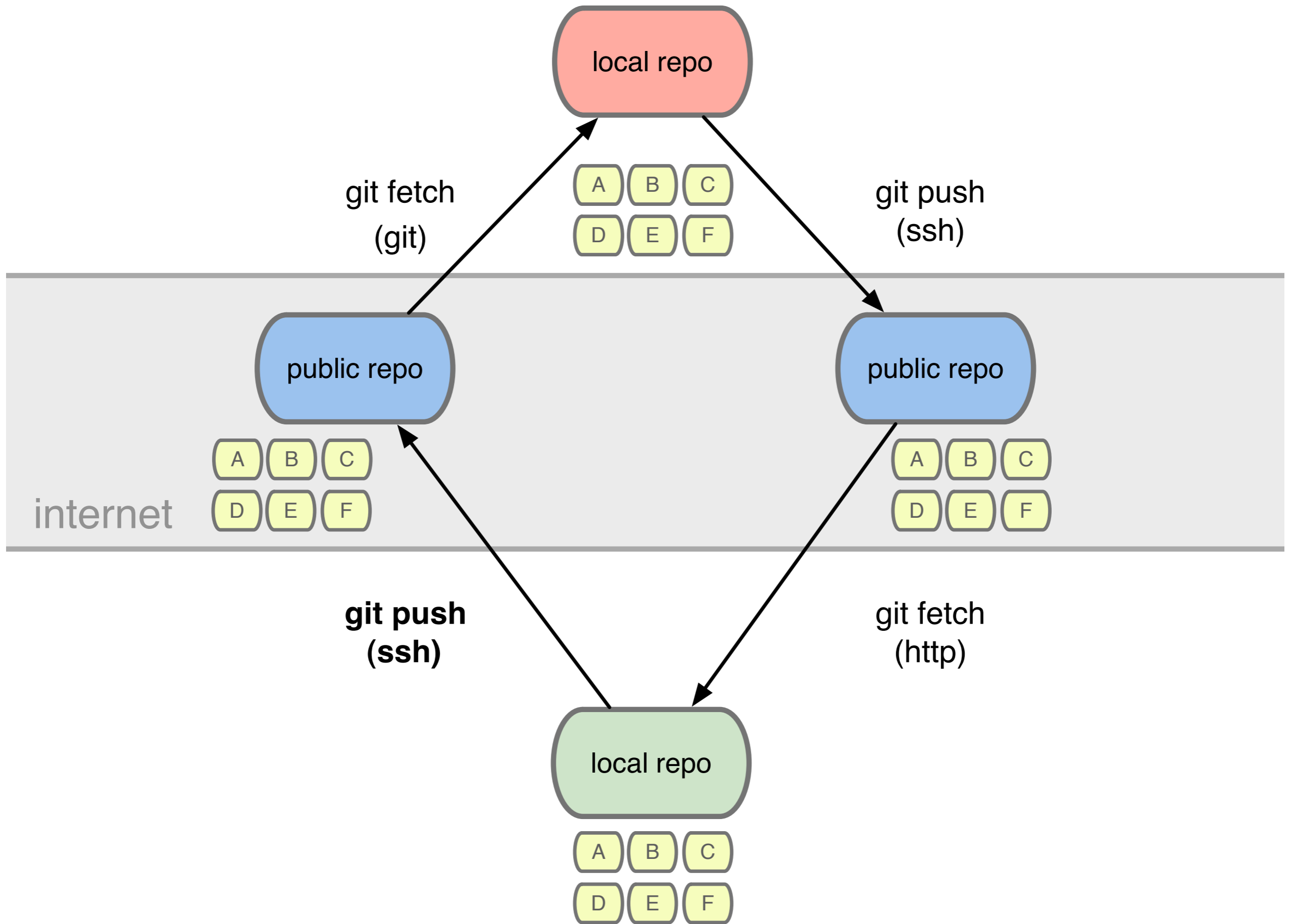








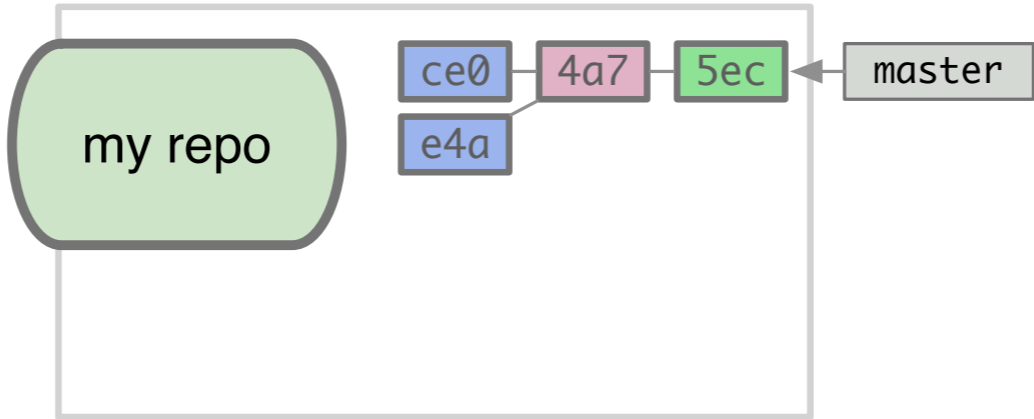




Multiple Remotes

developer
nick

developer
jessica

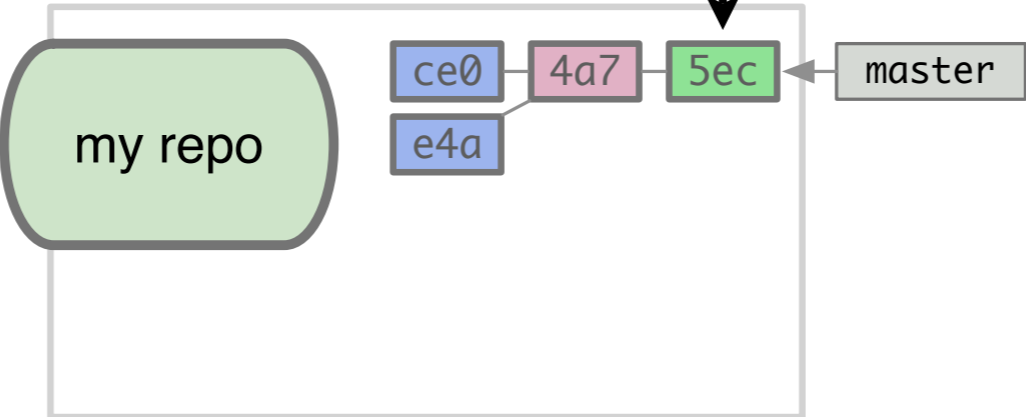


developer
nick

developer
jessica

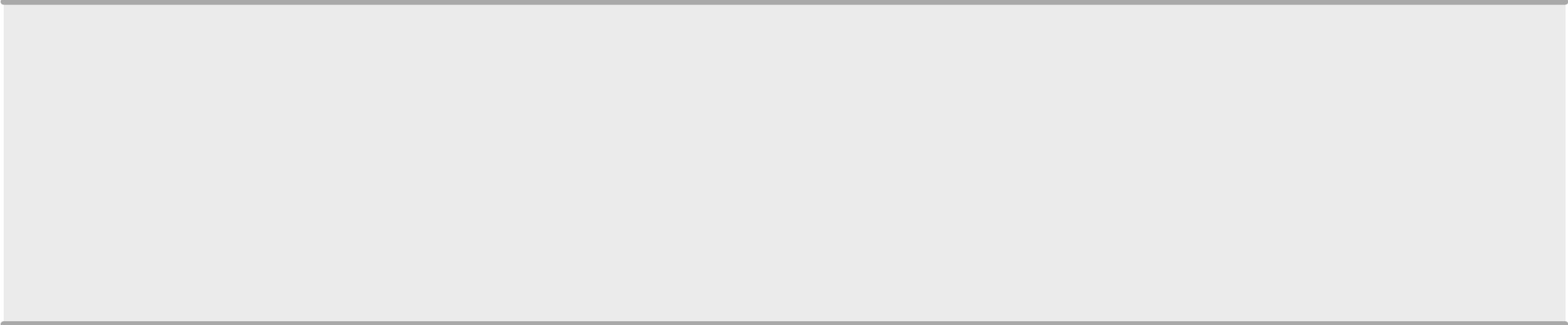


commit

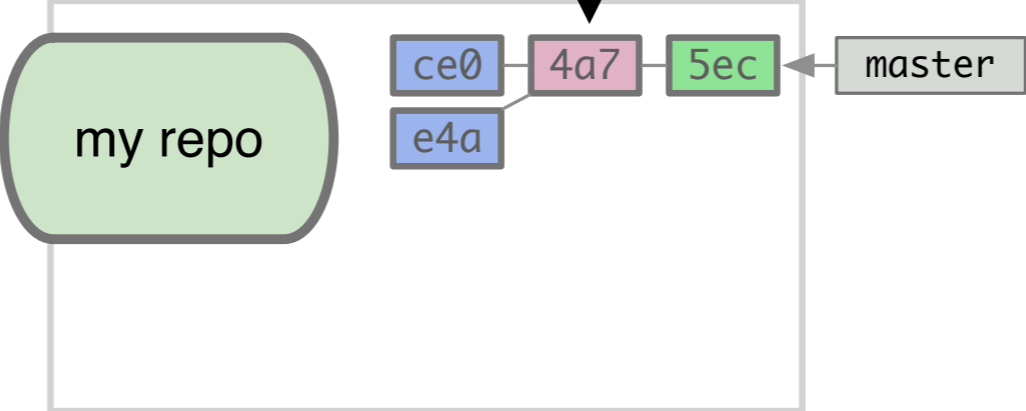


developer
nick

developer
jessica

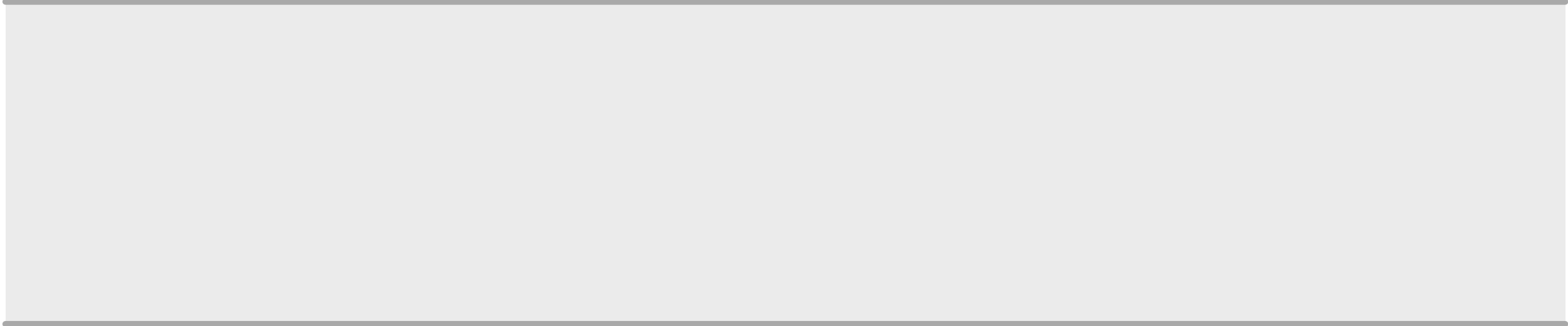


tree

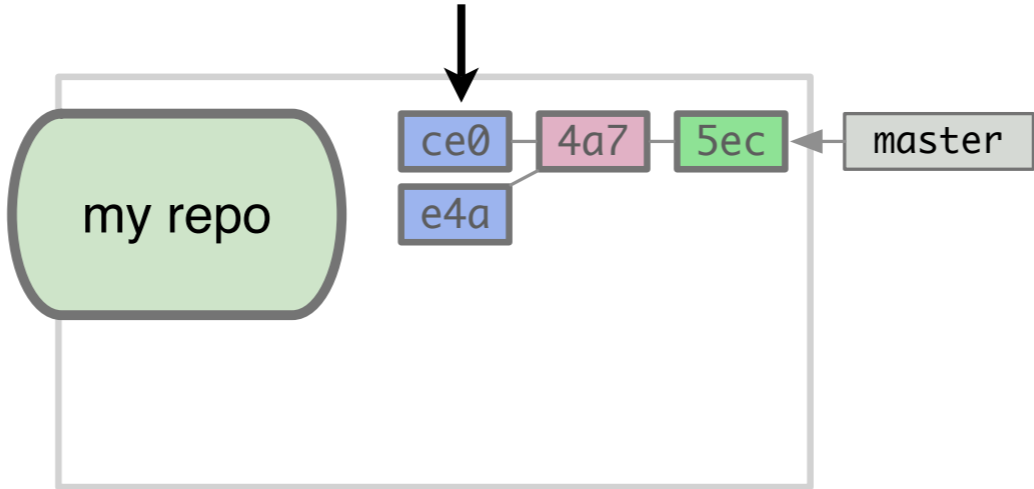


developer
nick

developer
jessica

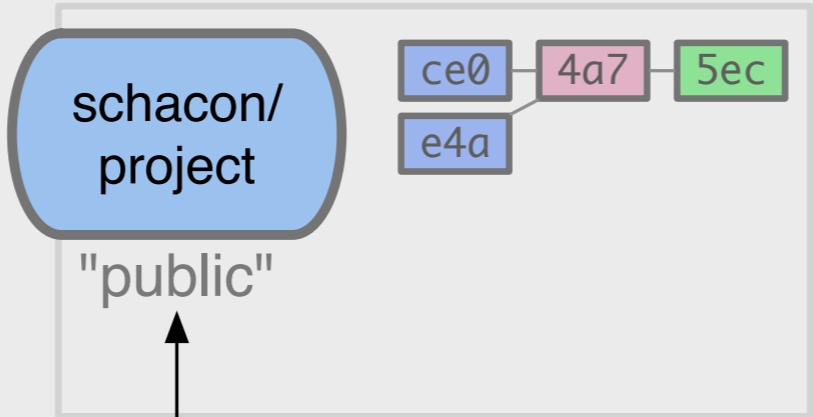


blobs

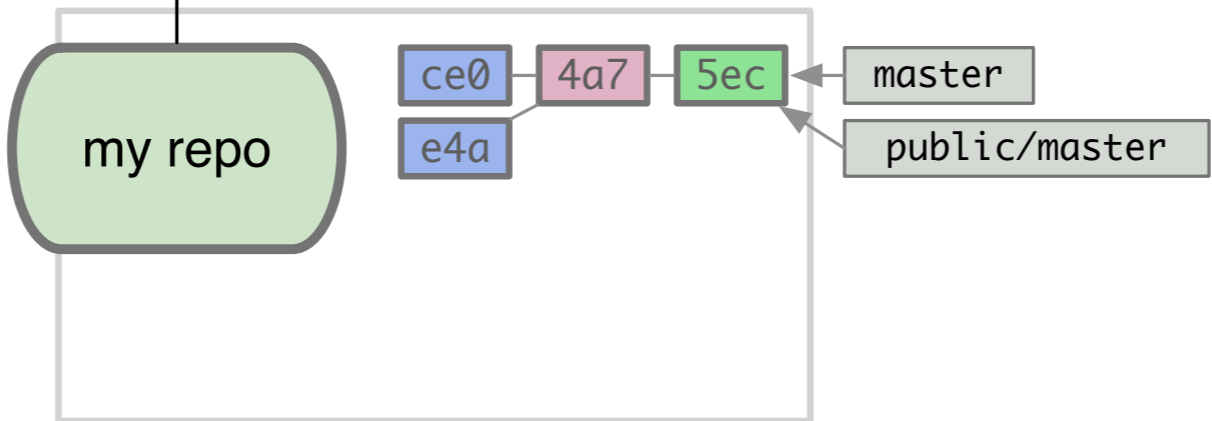


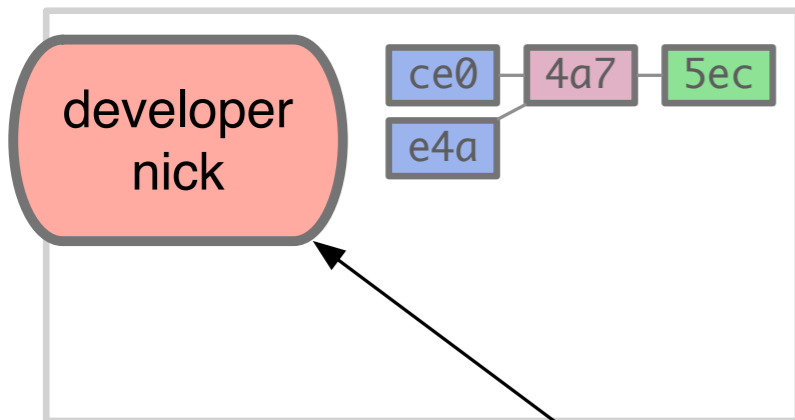
developer
nick

developer
jessica

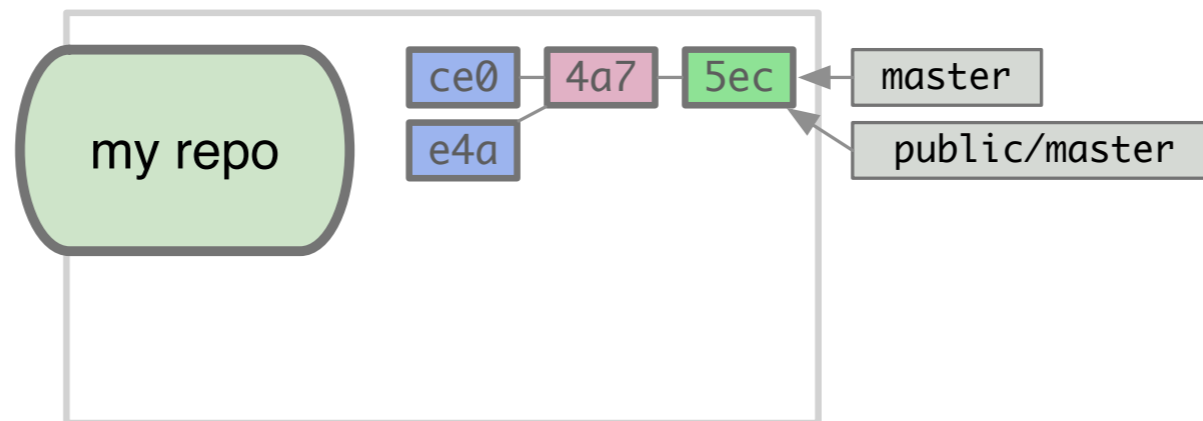
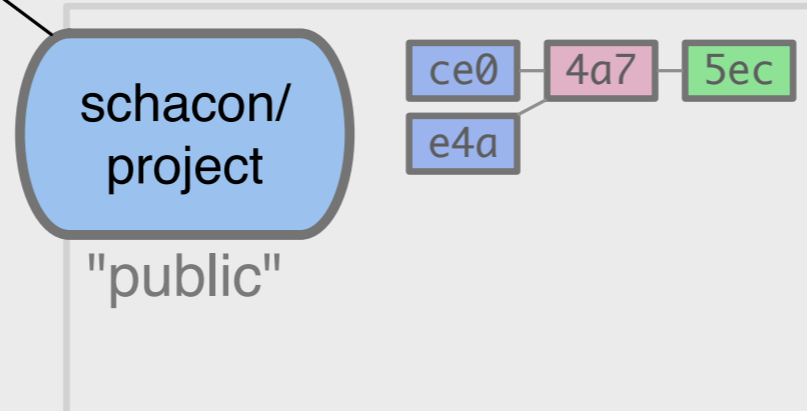


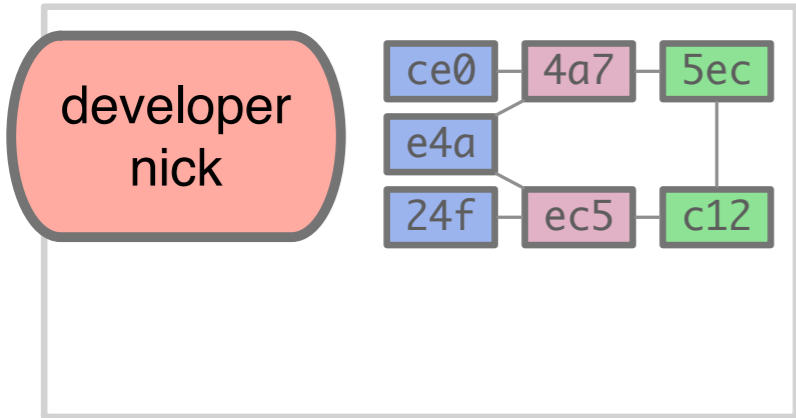
git push public



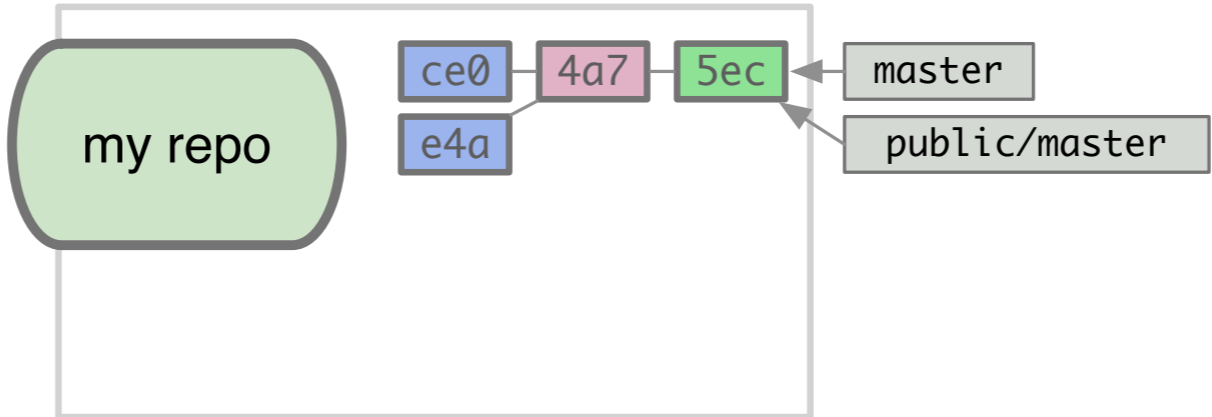
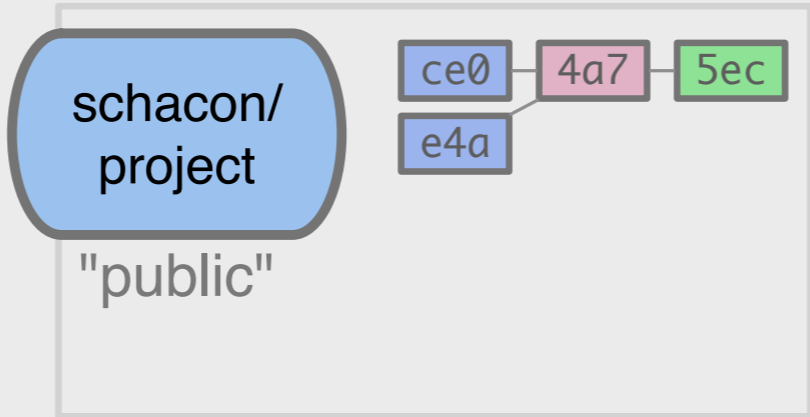


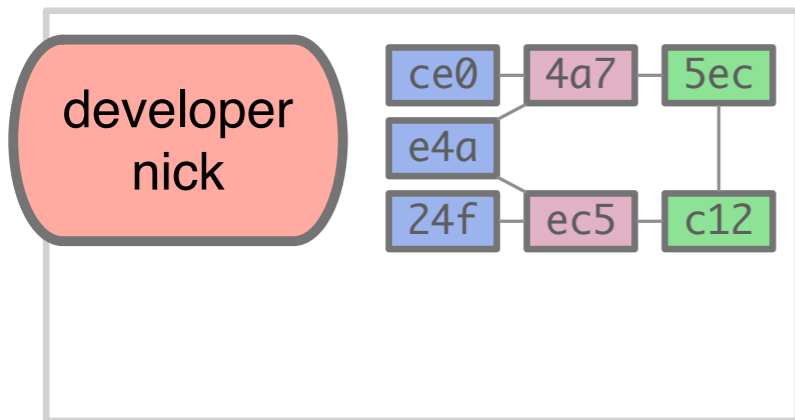
git clone (url)



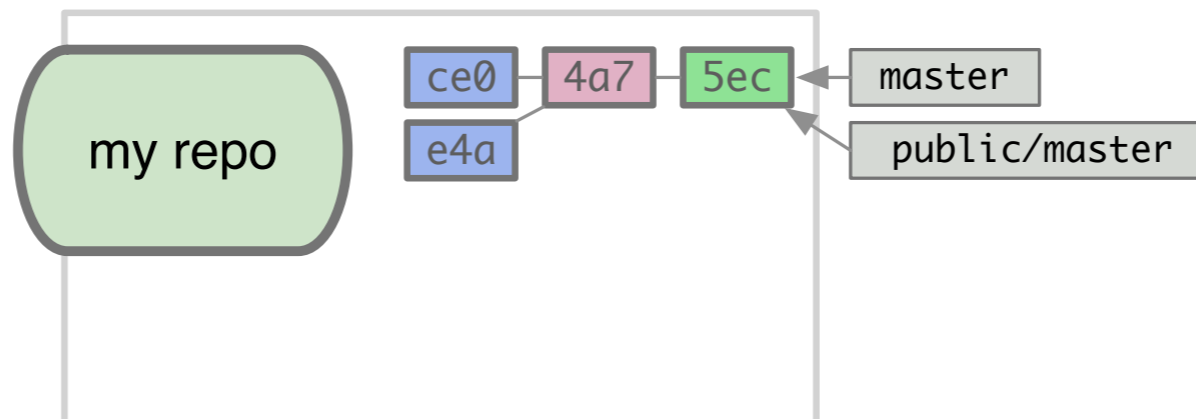
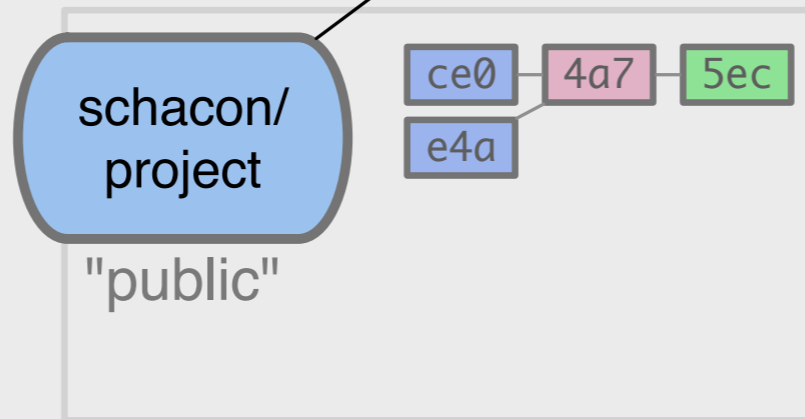
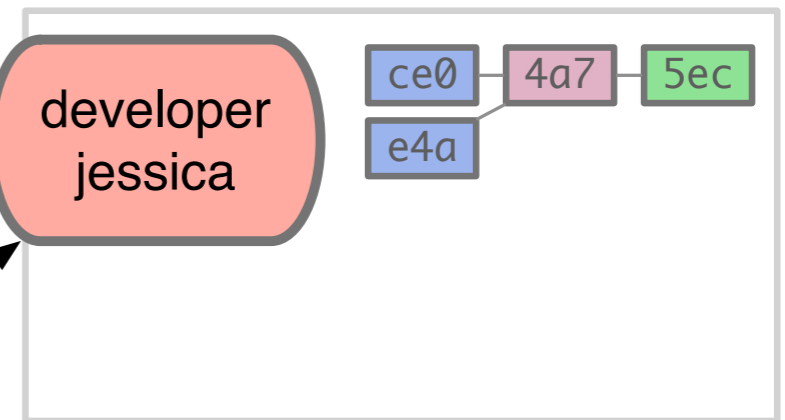


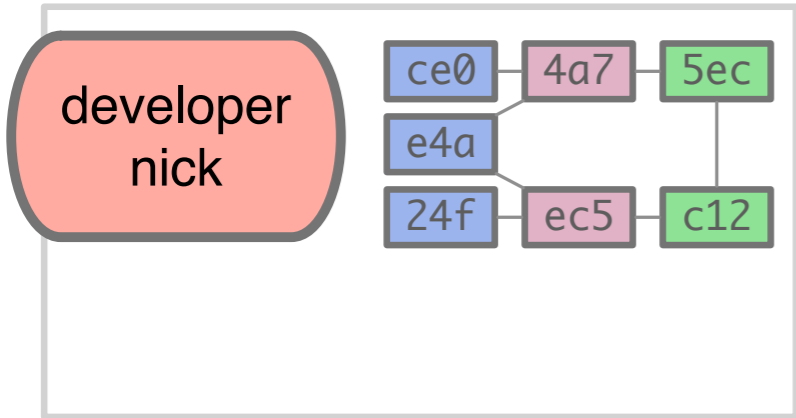
git commit



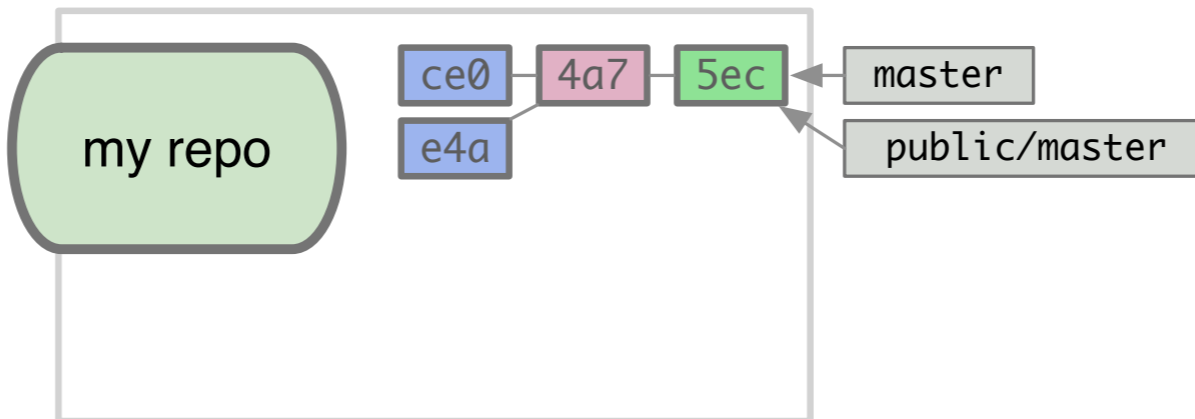
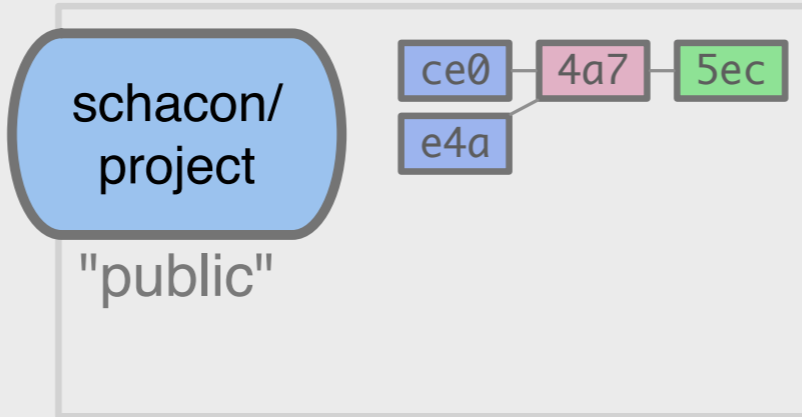
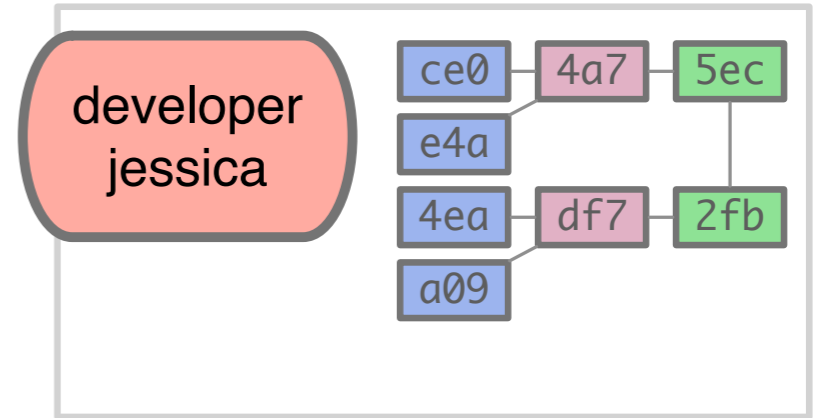


git clone (url)

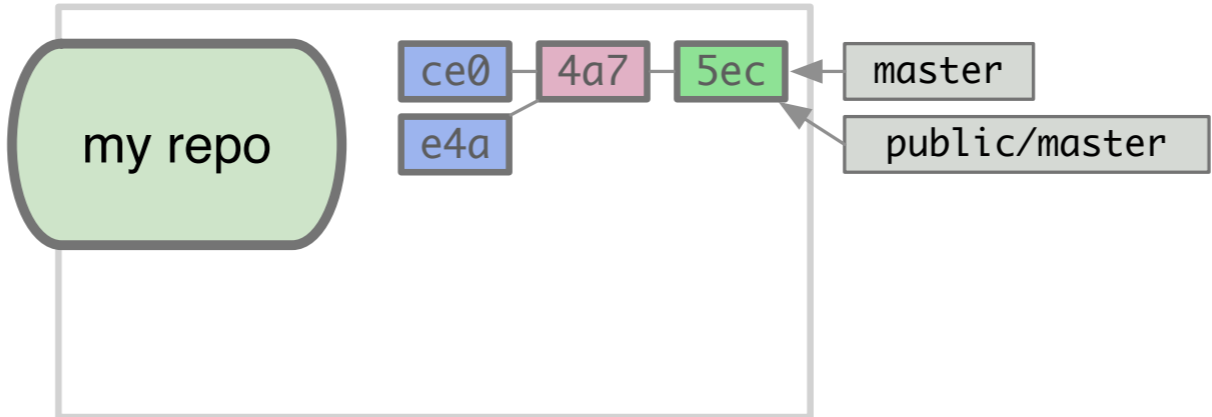
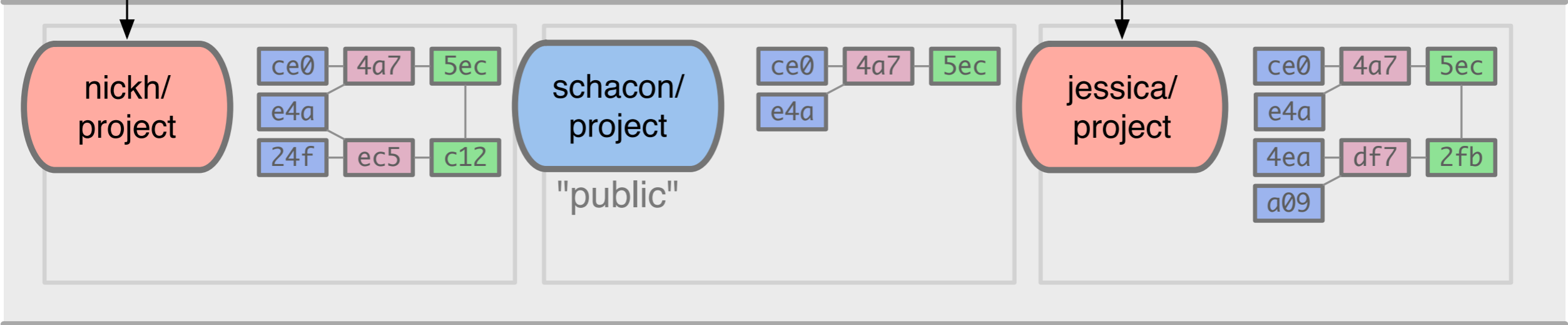
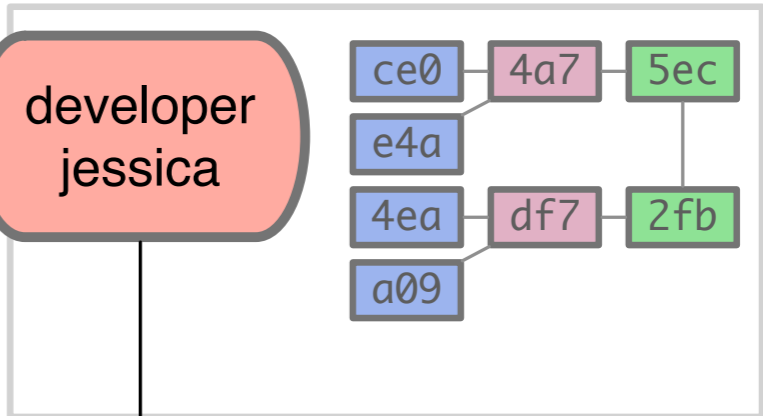
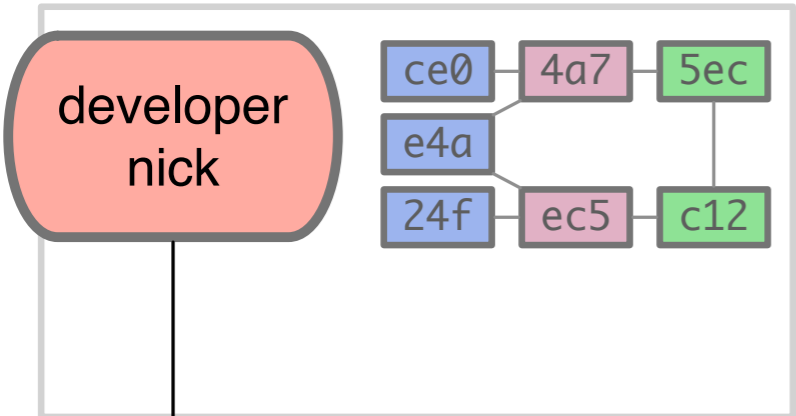


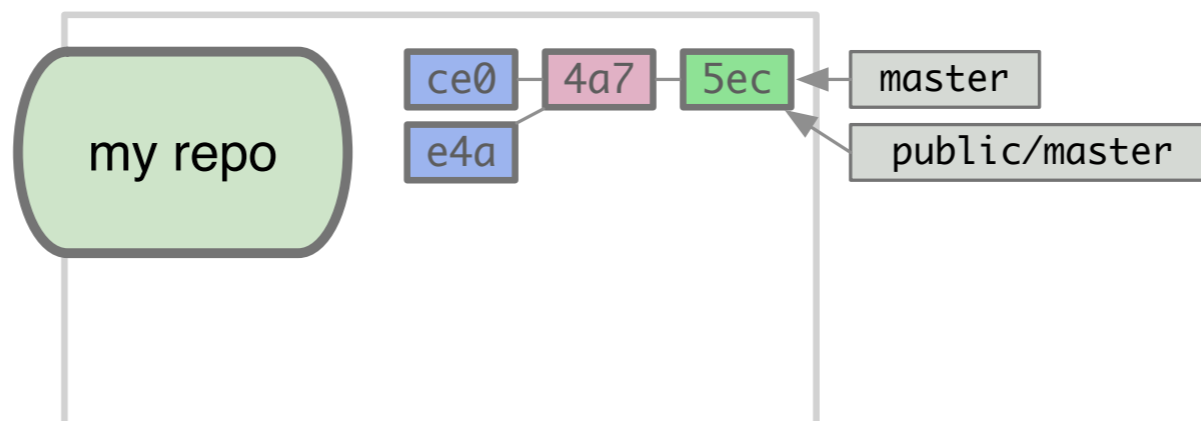
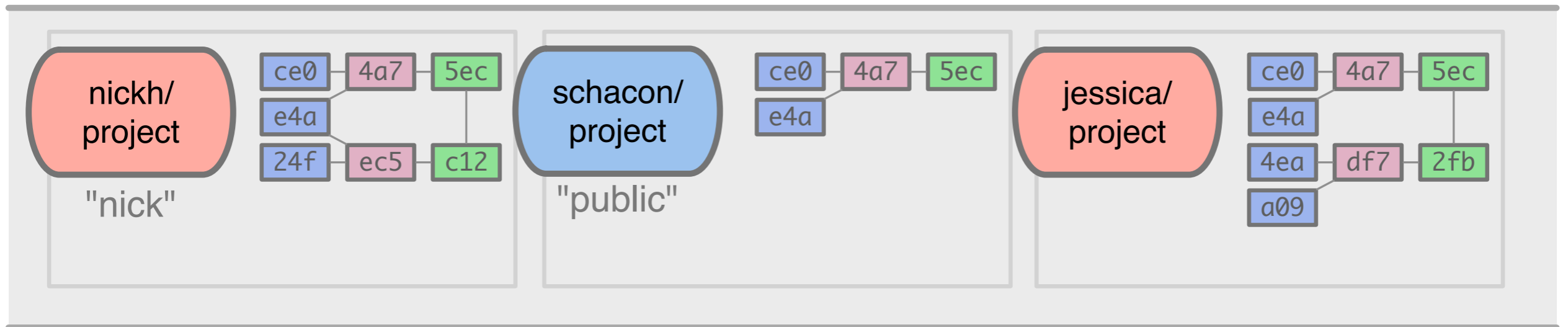
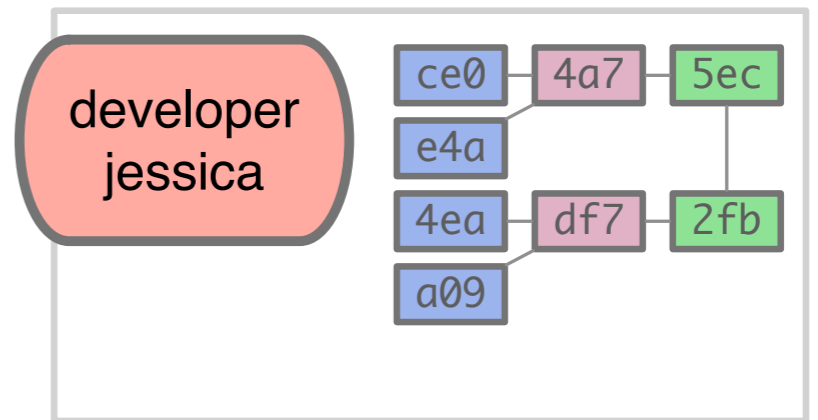
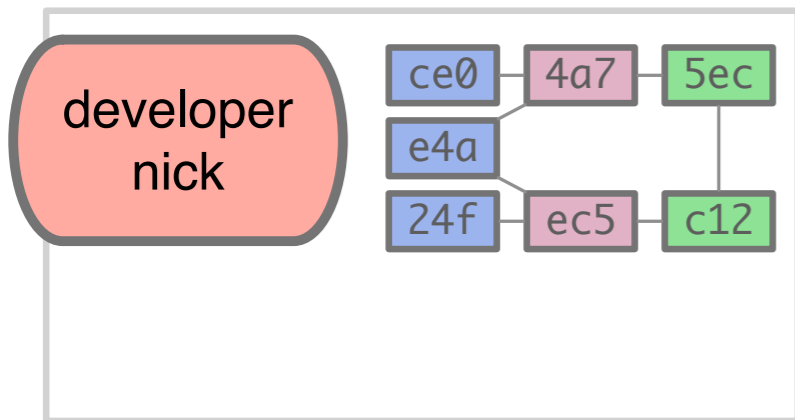


git commit

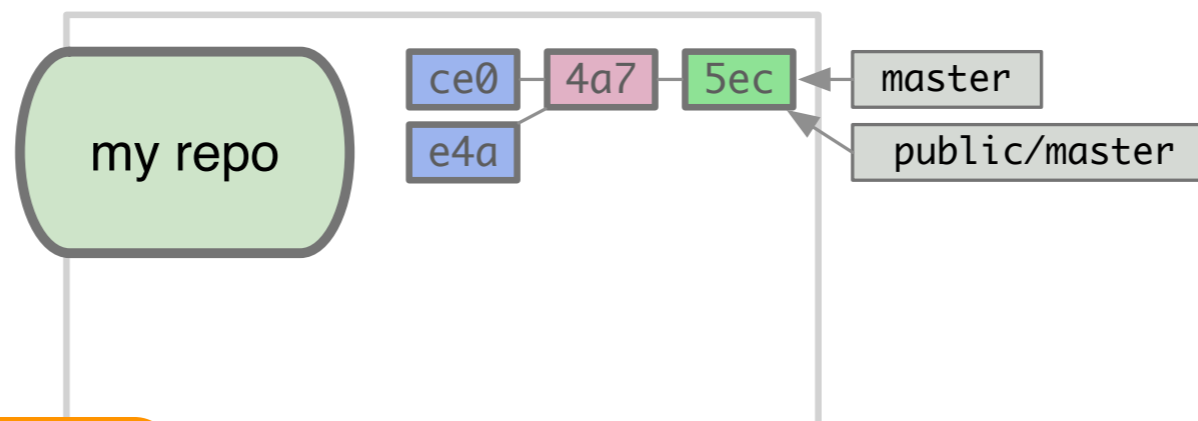
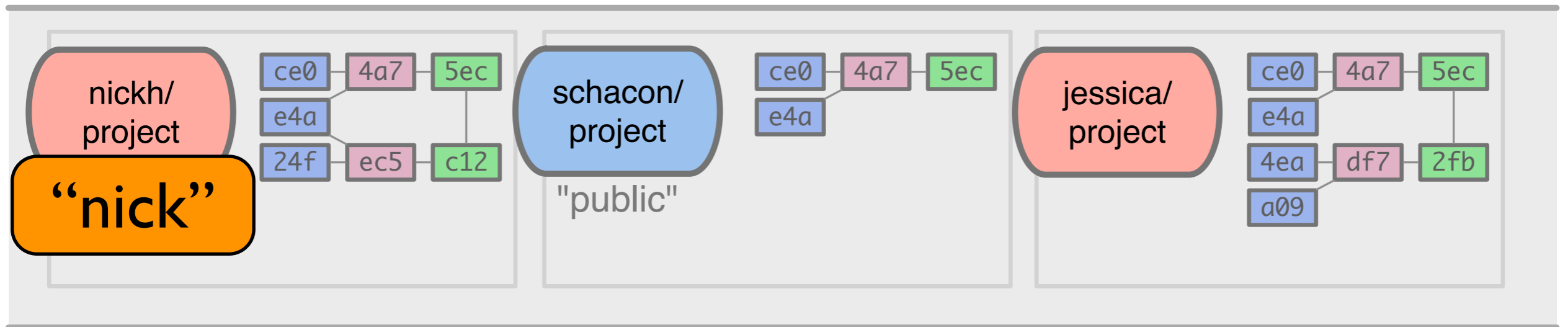
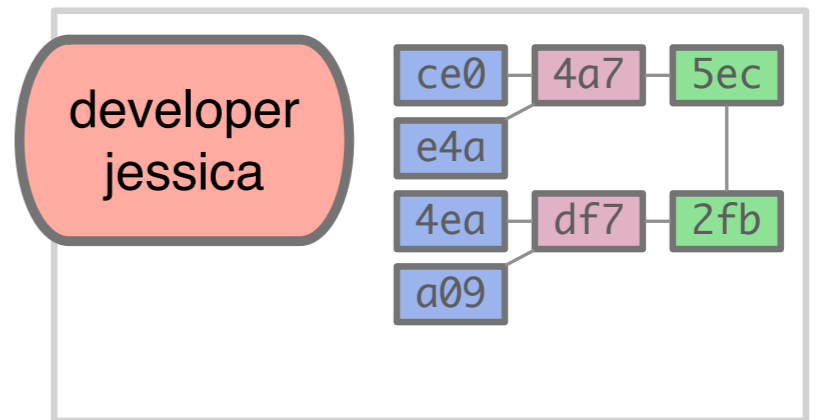
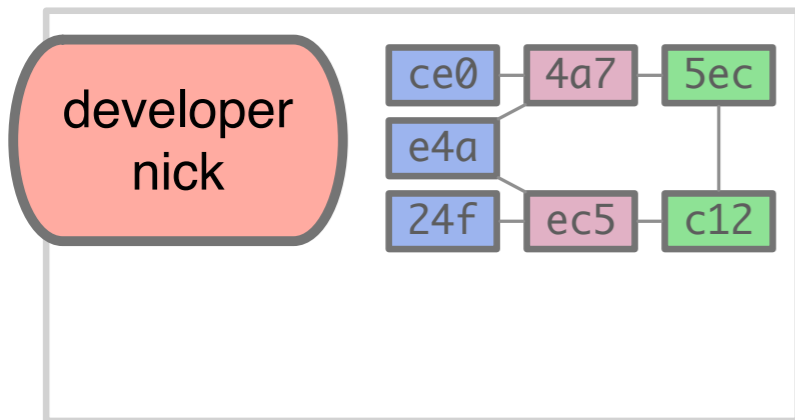


git push

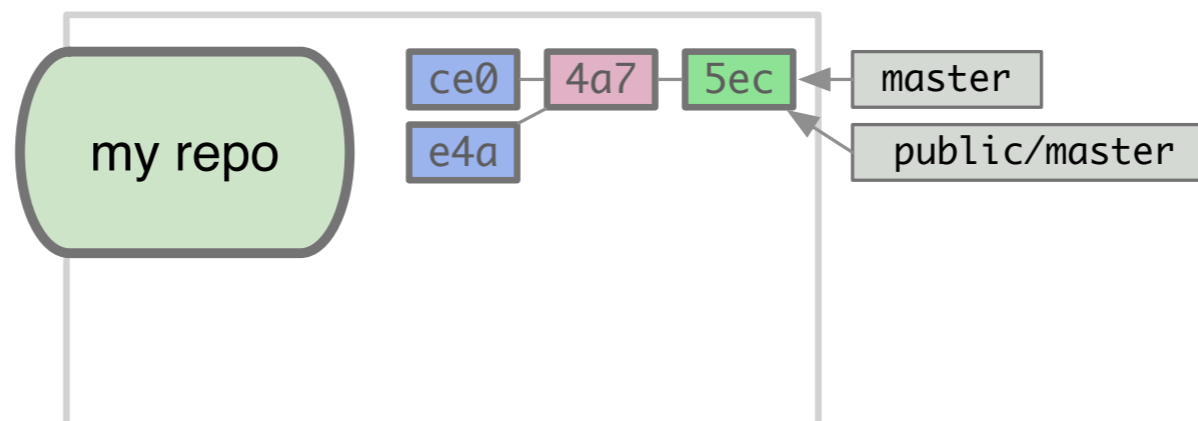
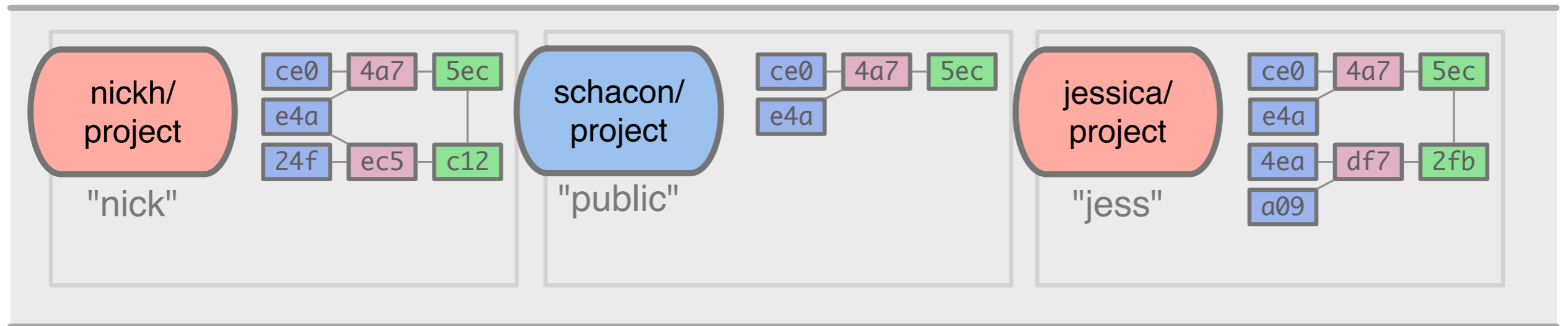
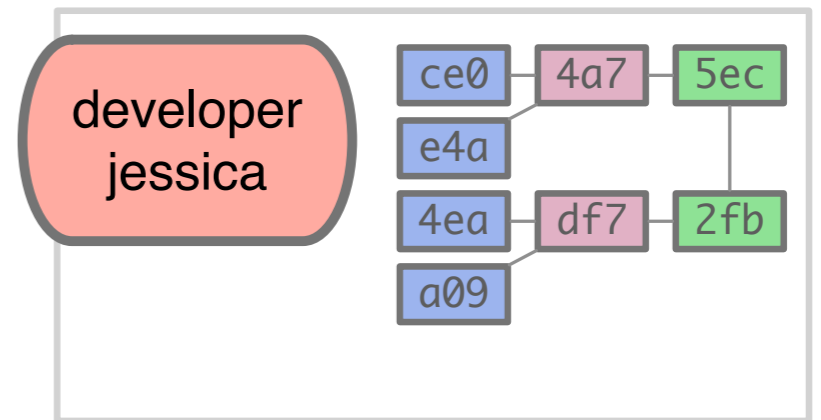
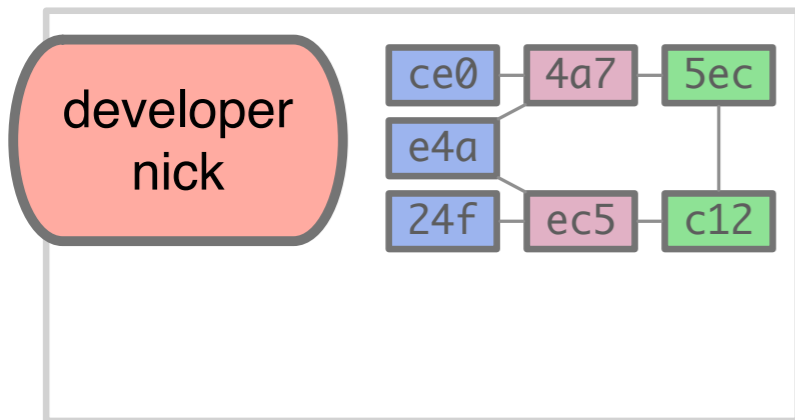




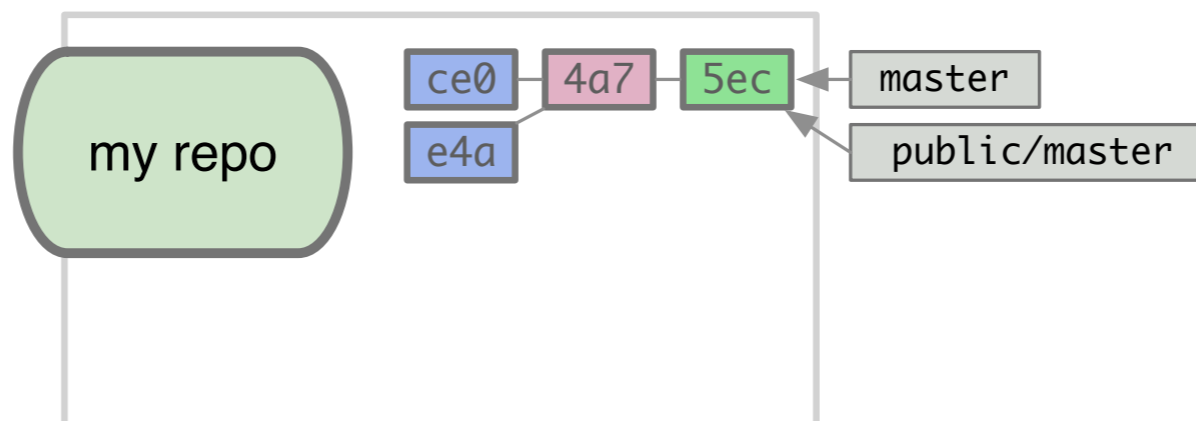
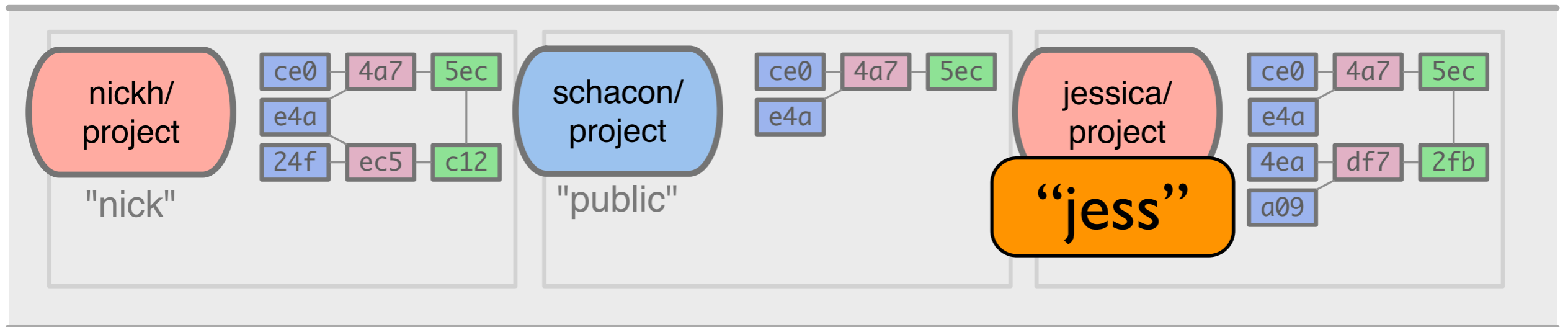
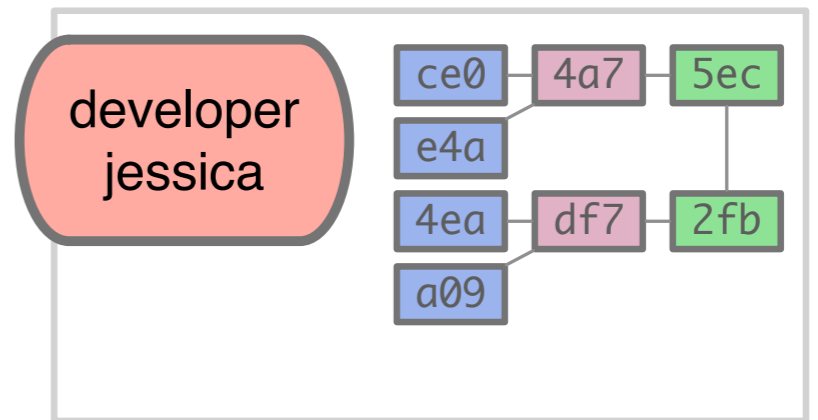
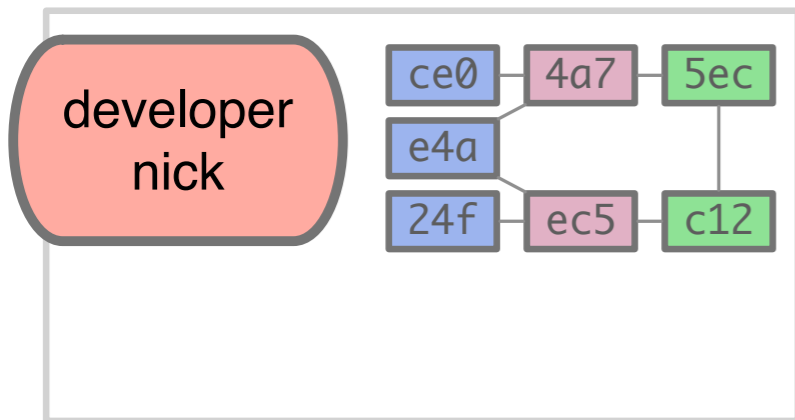
`git remote add nick git://github.com/nickh/project.git`



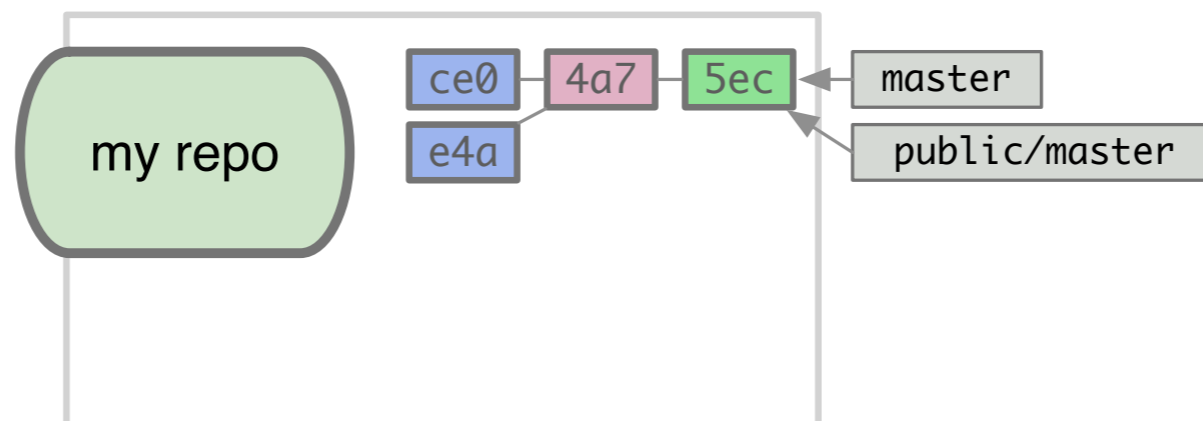
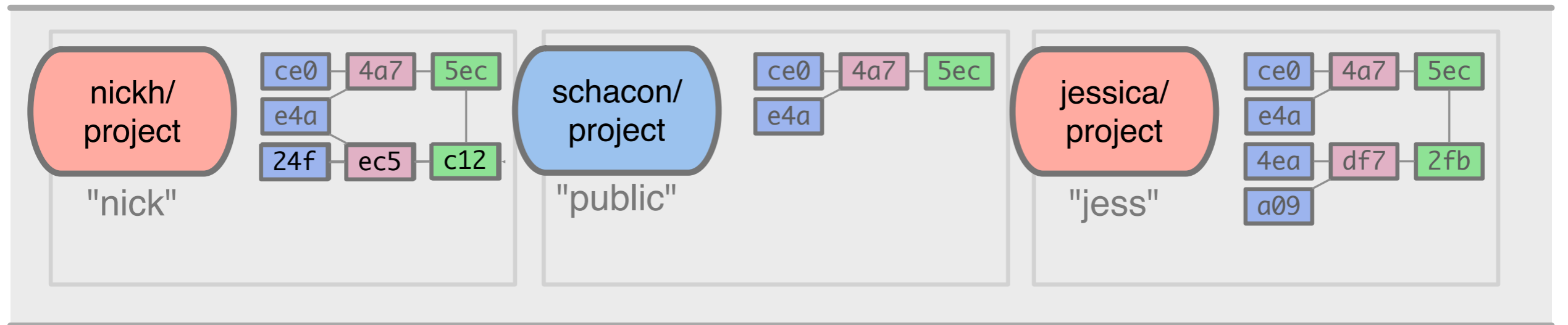
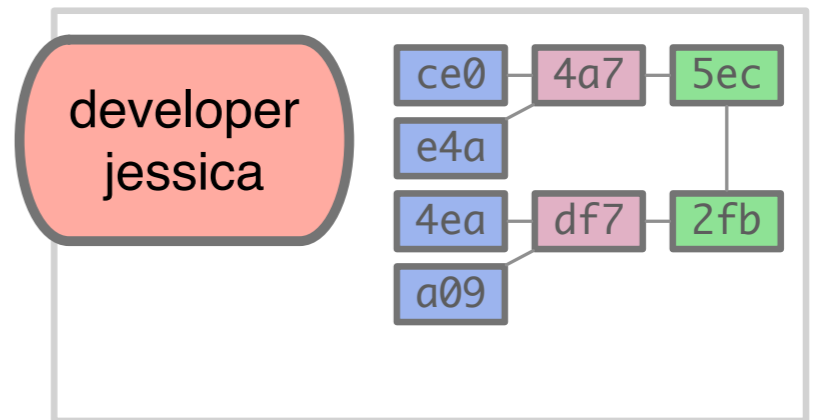
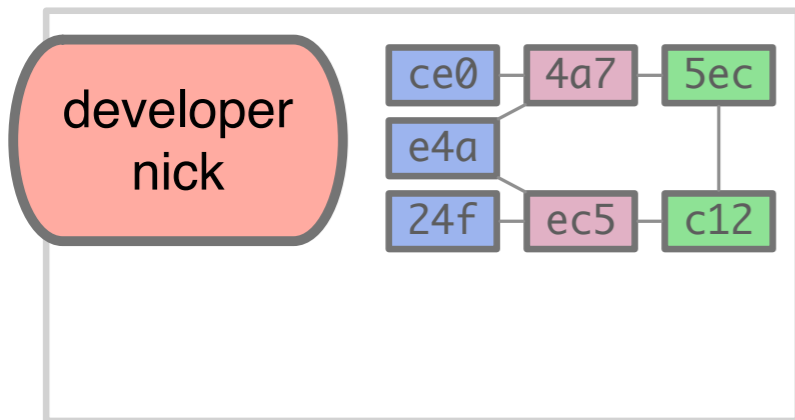
git remote add **nick** git://github.com/nickh/project.git



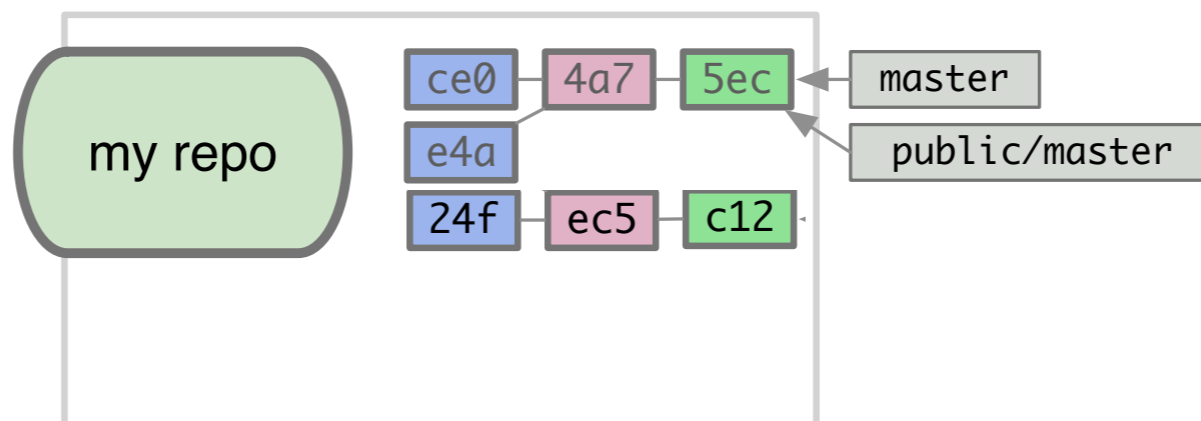
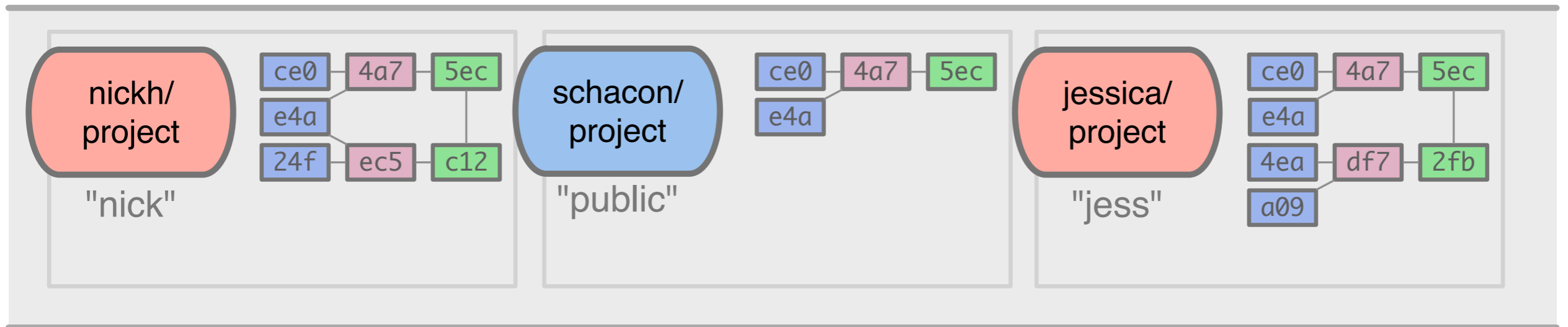
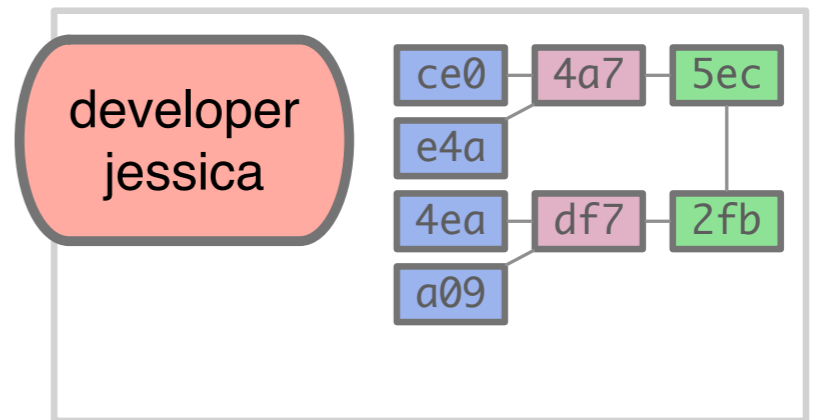
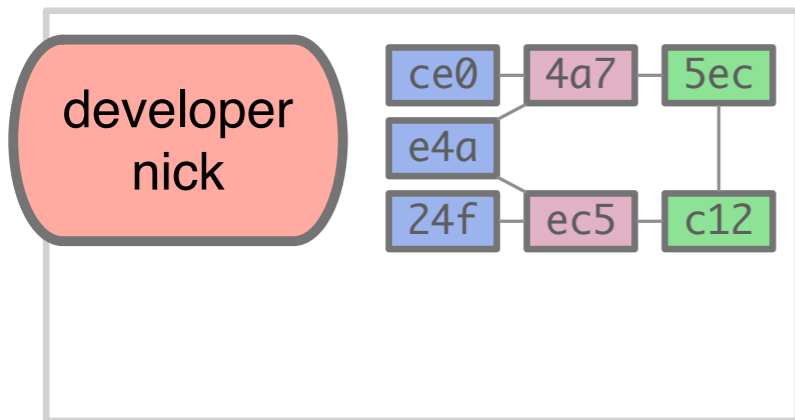
`git remote add jess git://github.com/jessica/project.git`



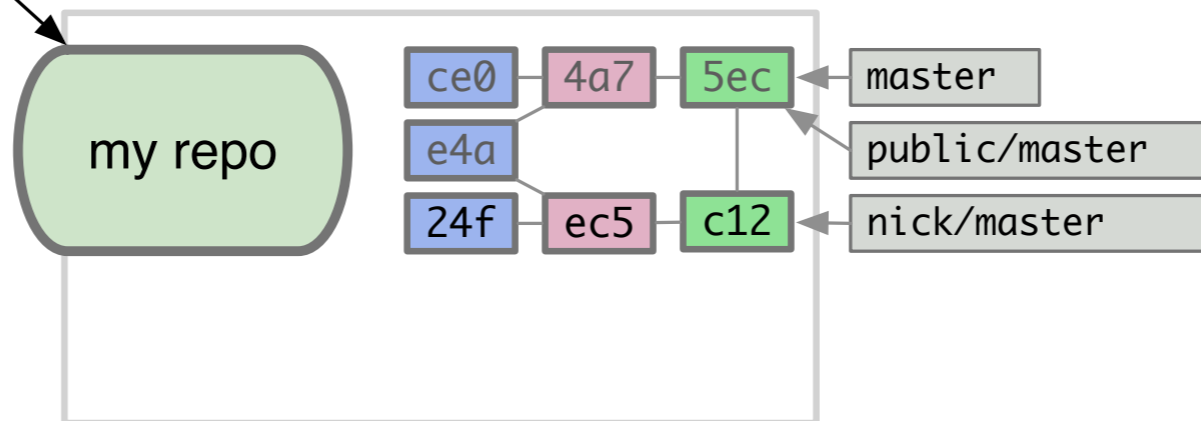
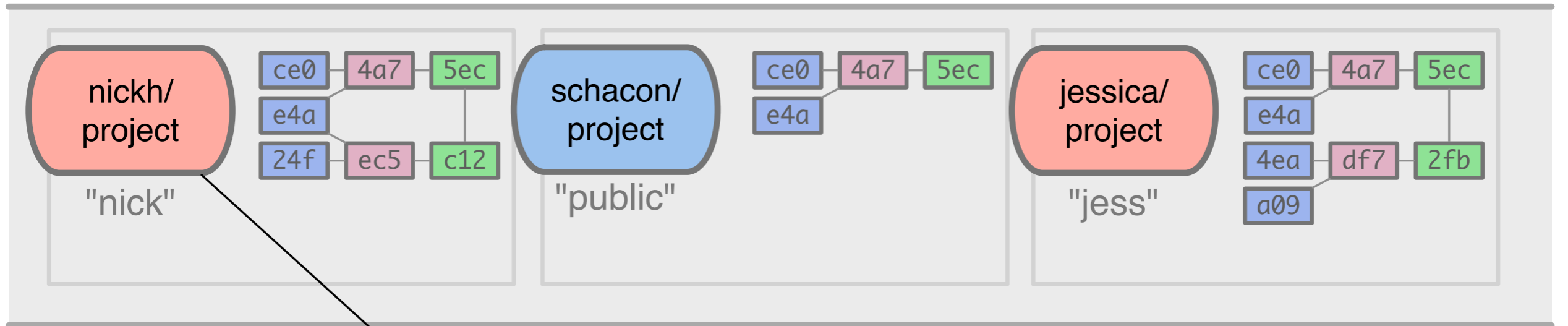
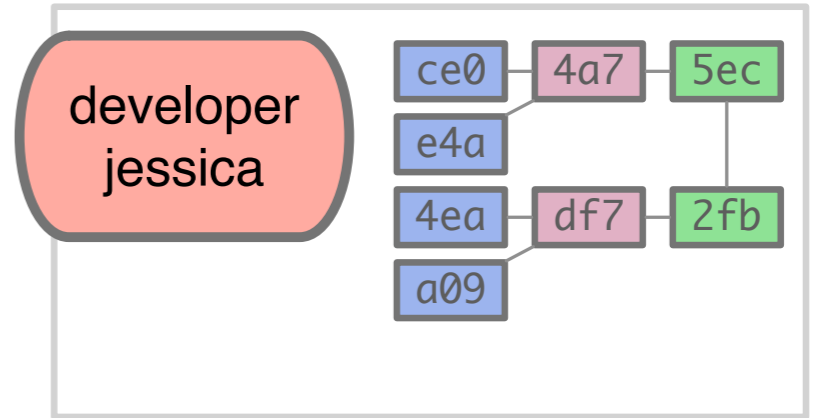
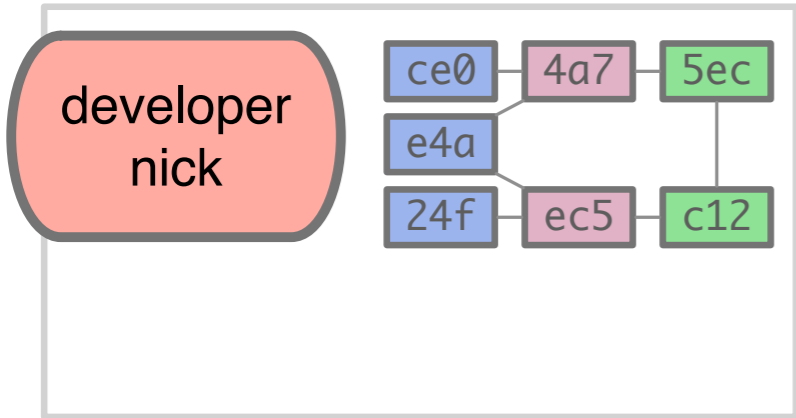
git remote add **jess** git://github.com/jessica/project.git

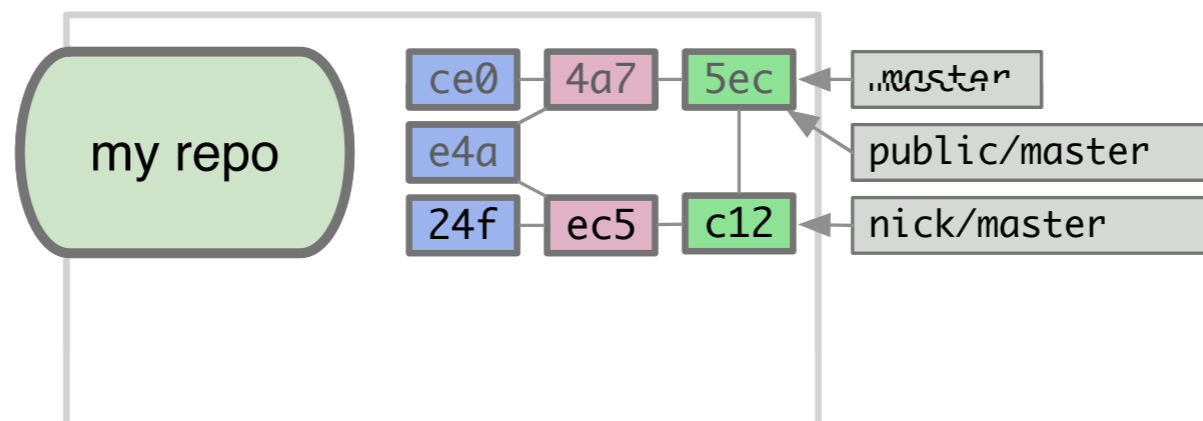
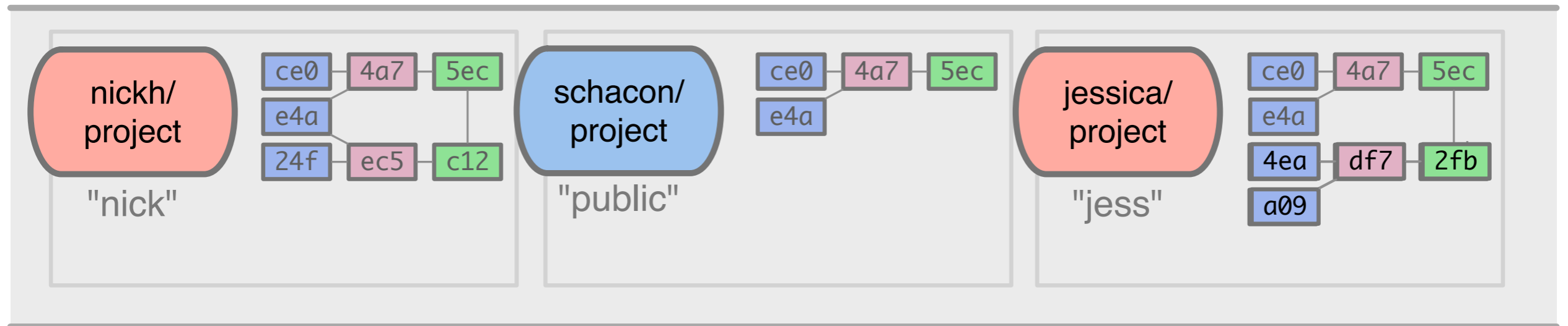
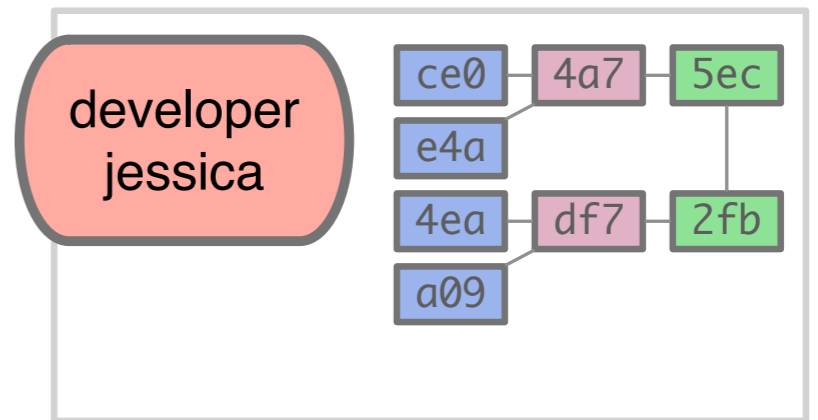
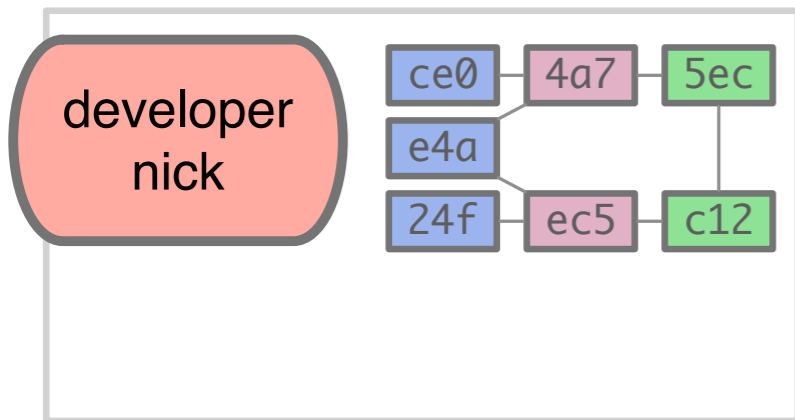


git fetch nick

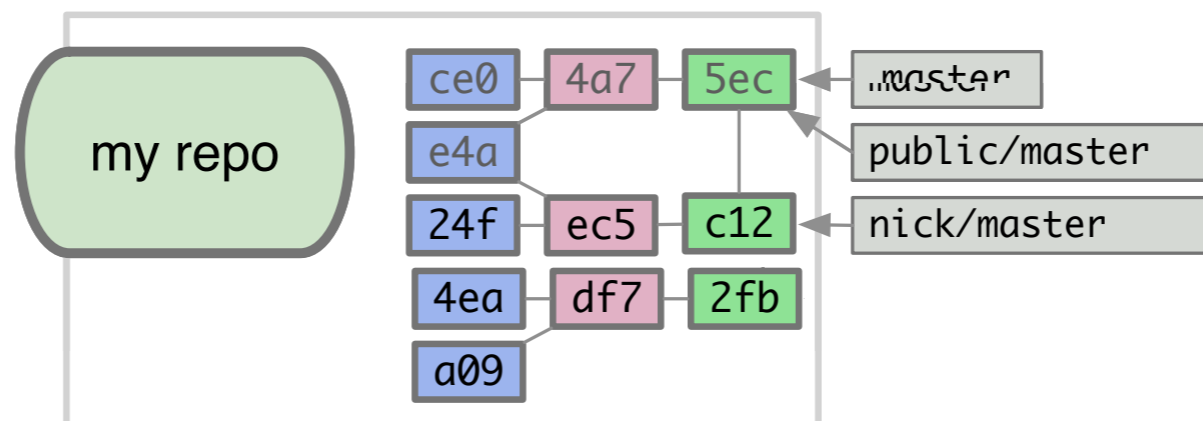
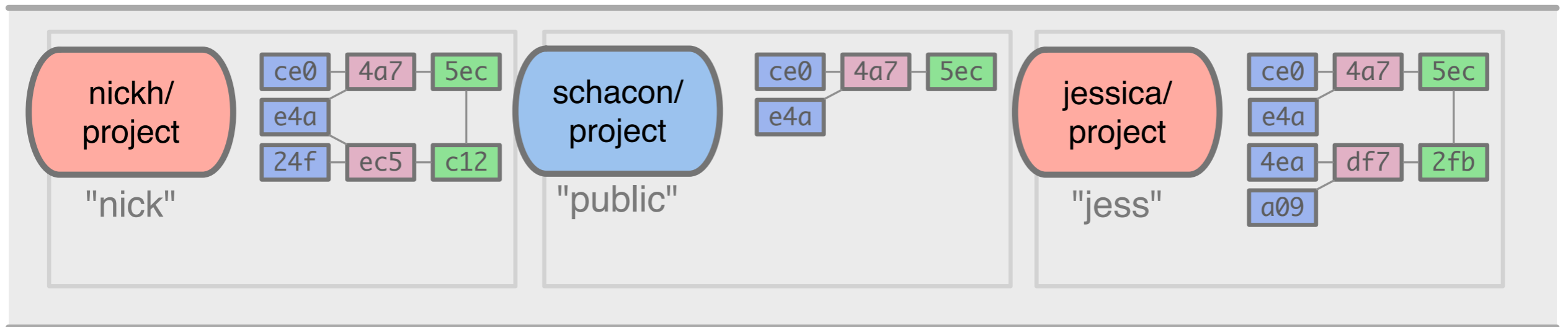
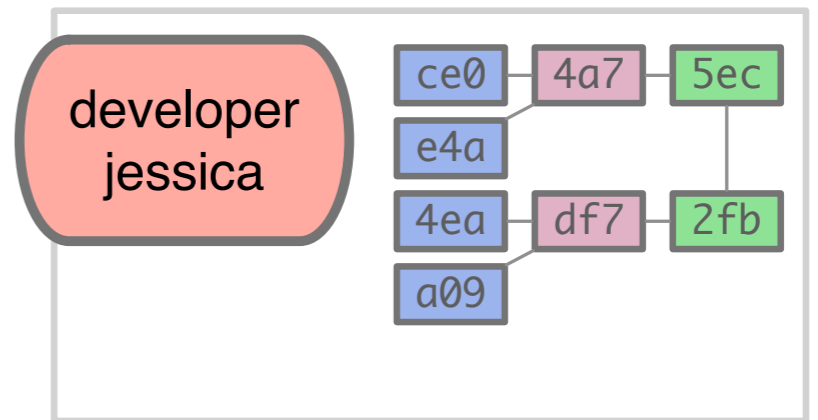
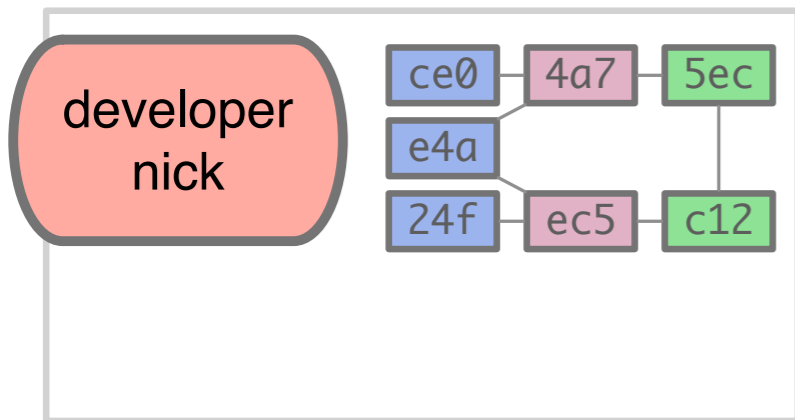


git fetch nick

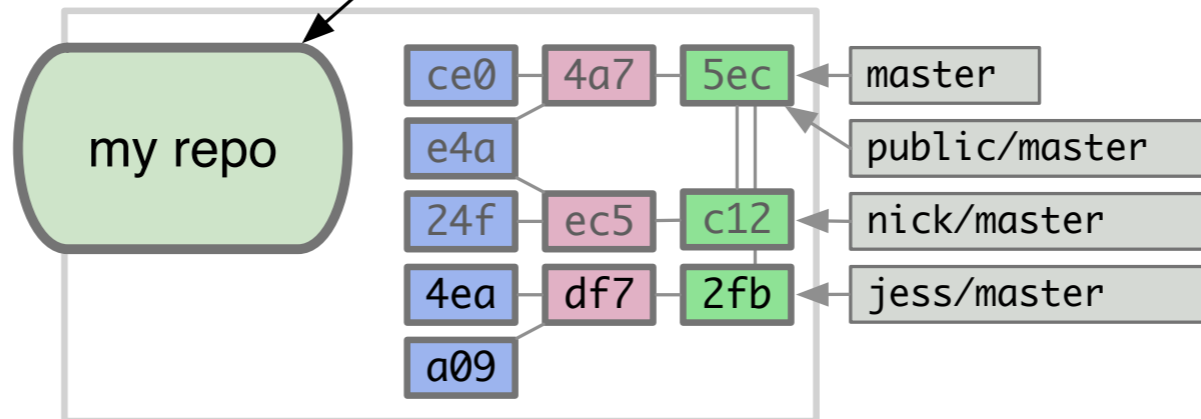
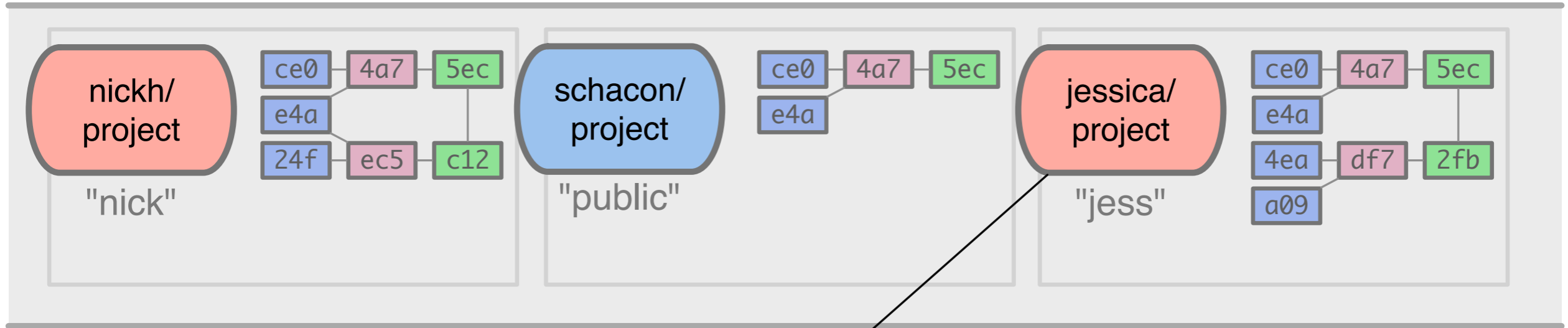
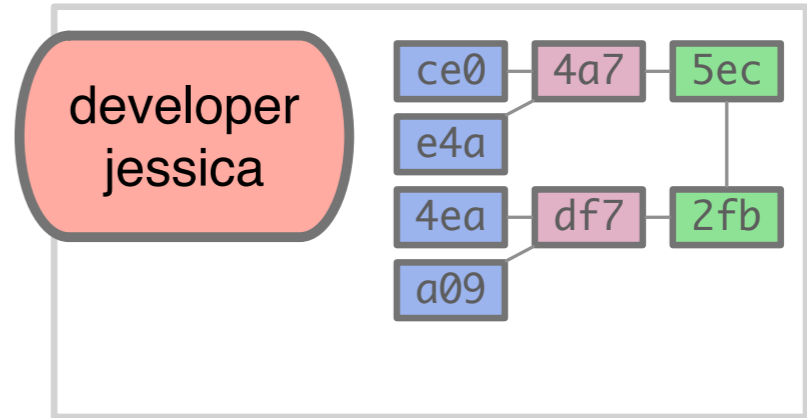
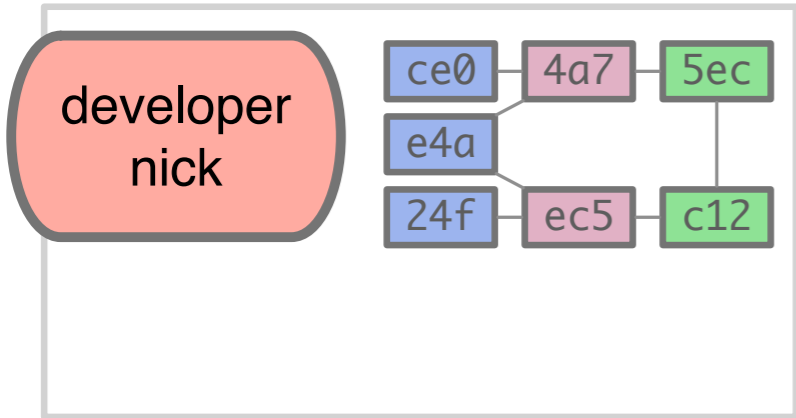


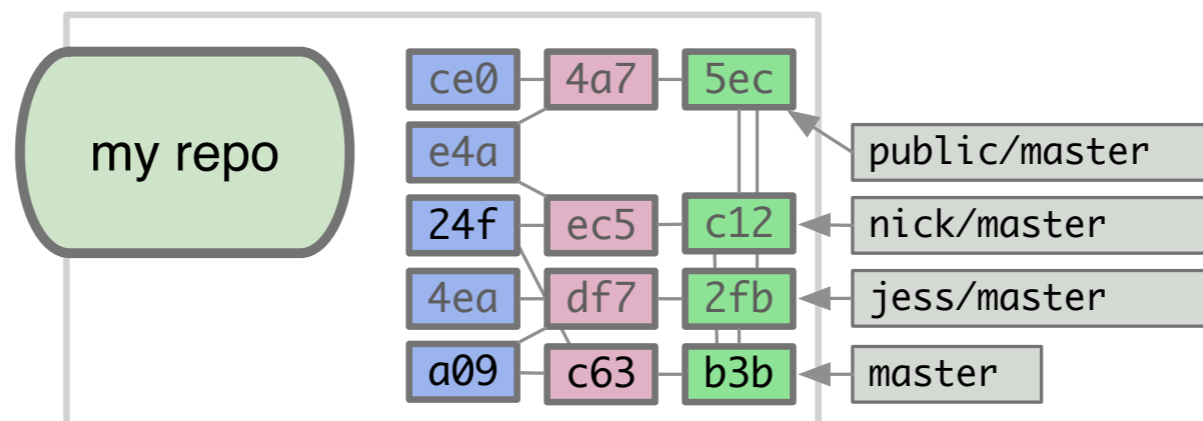
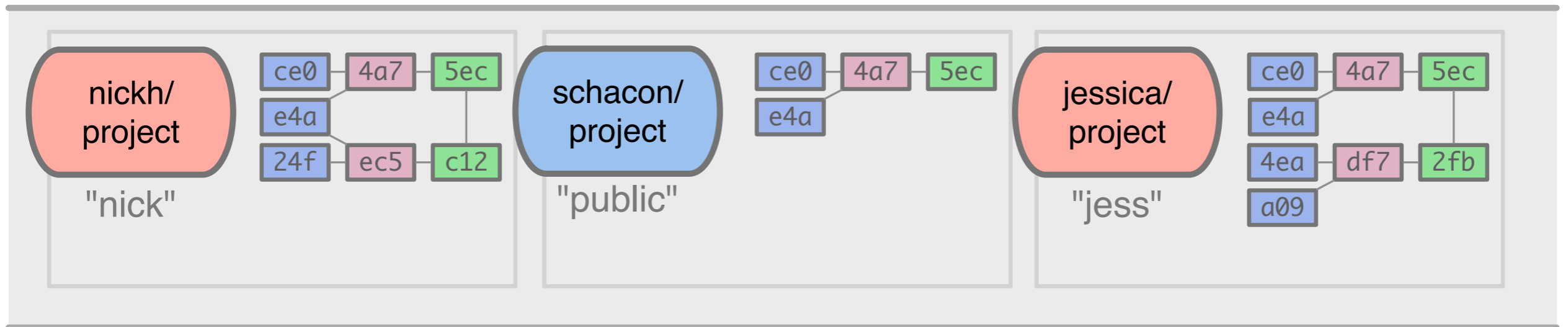
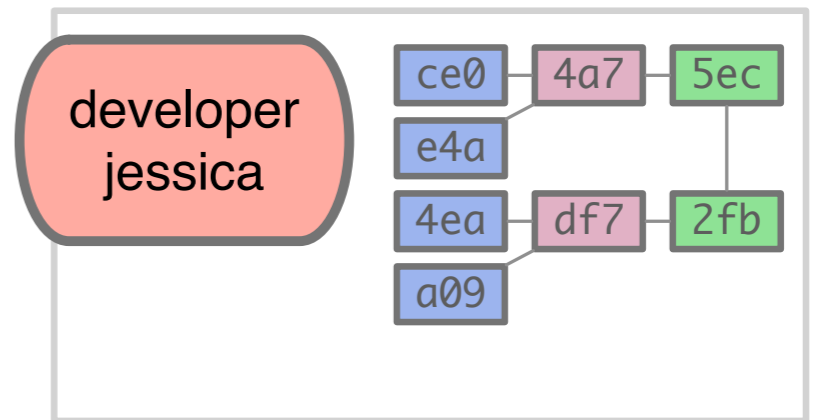
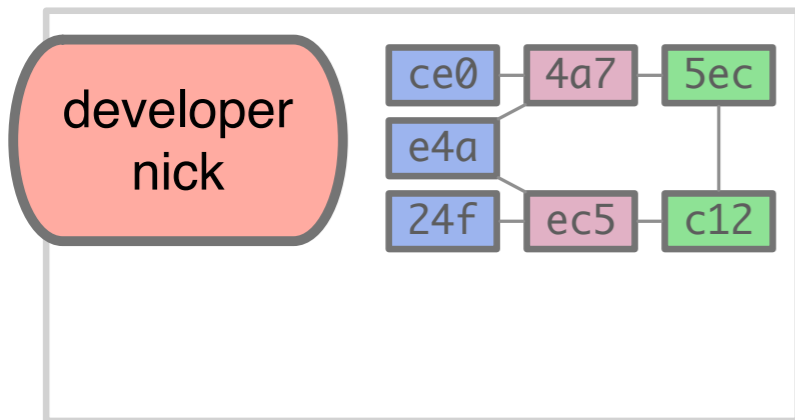


git fetch jess

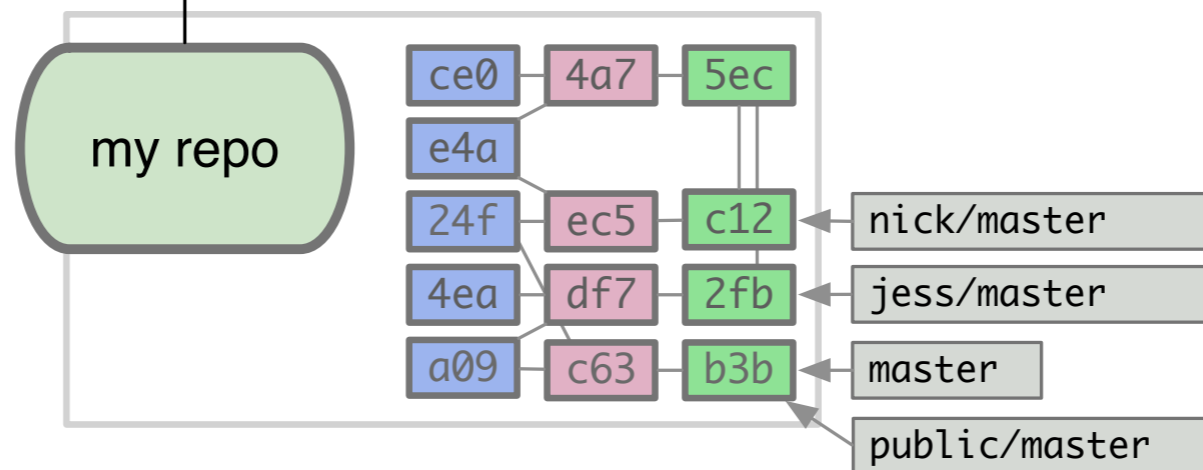
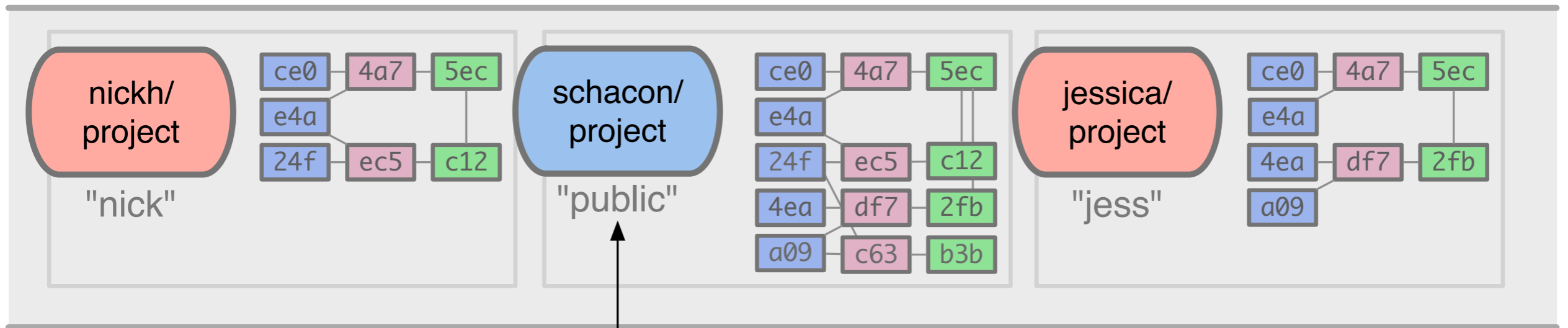
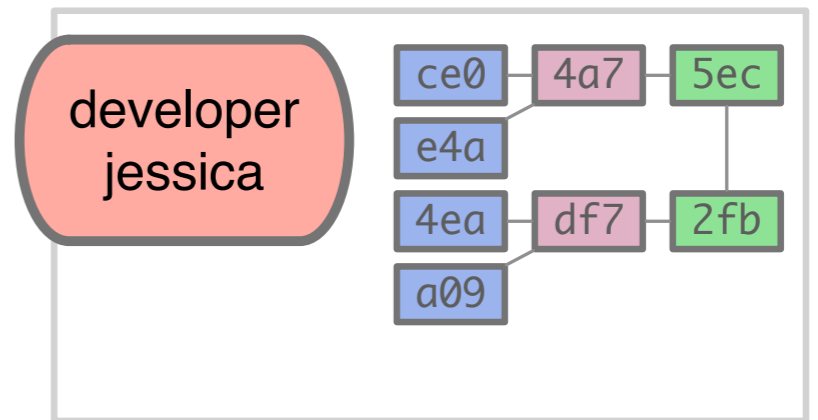
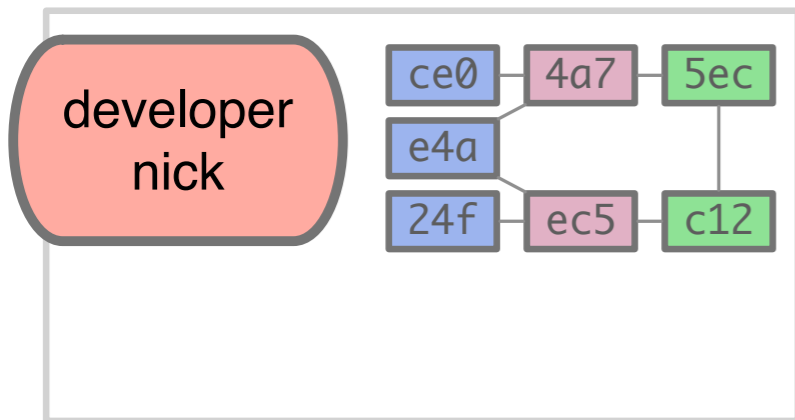


git fetch jess

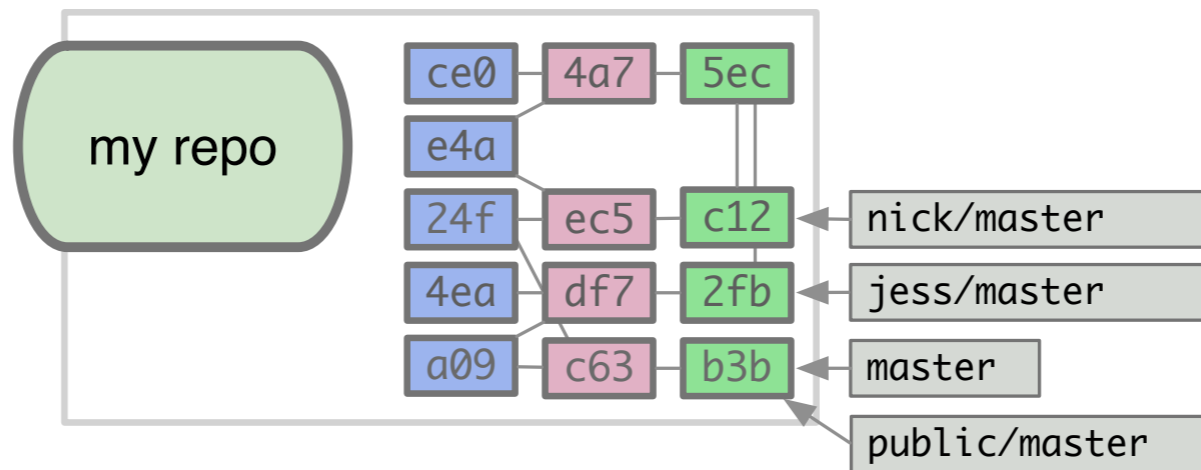
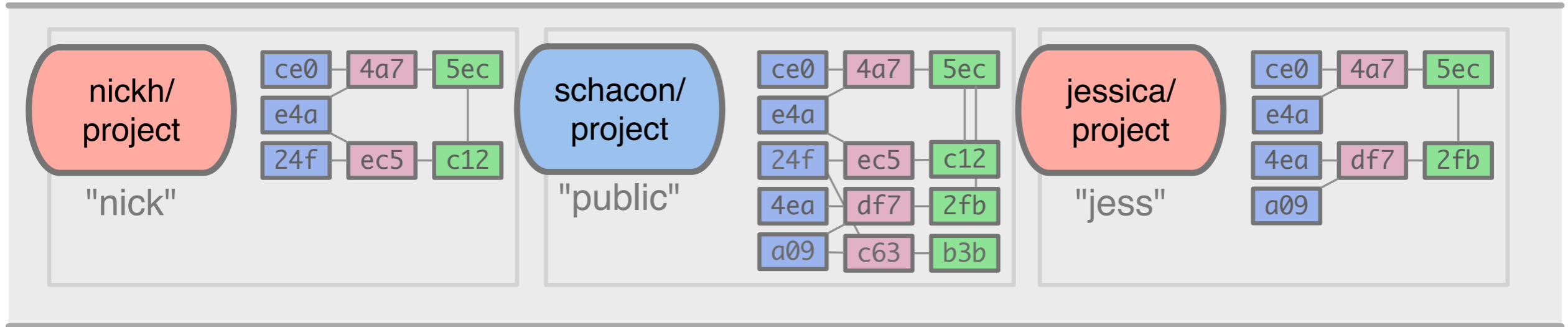
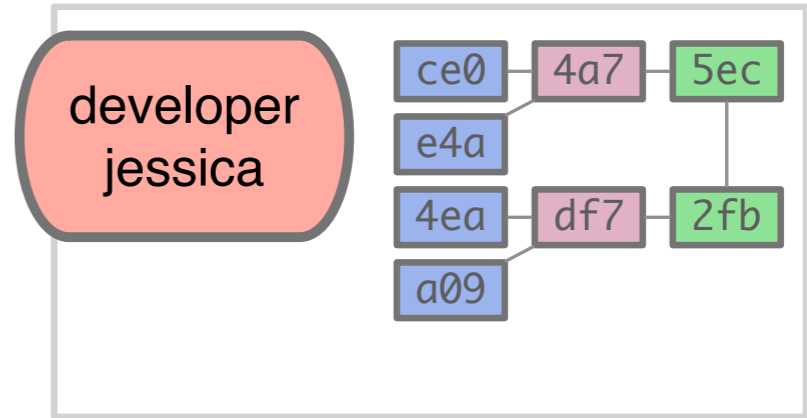
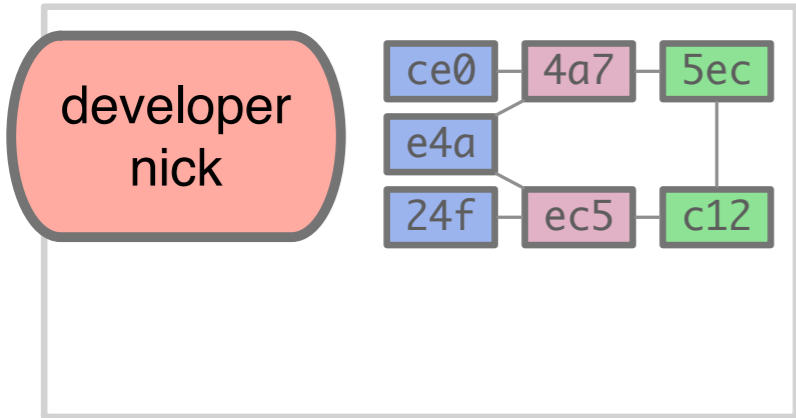


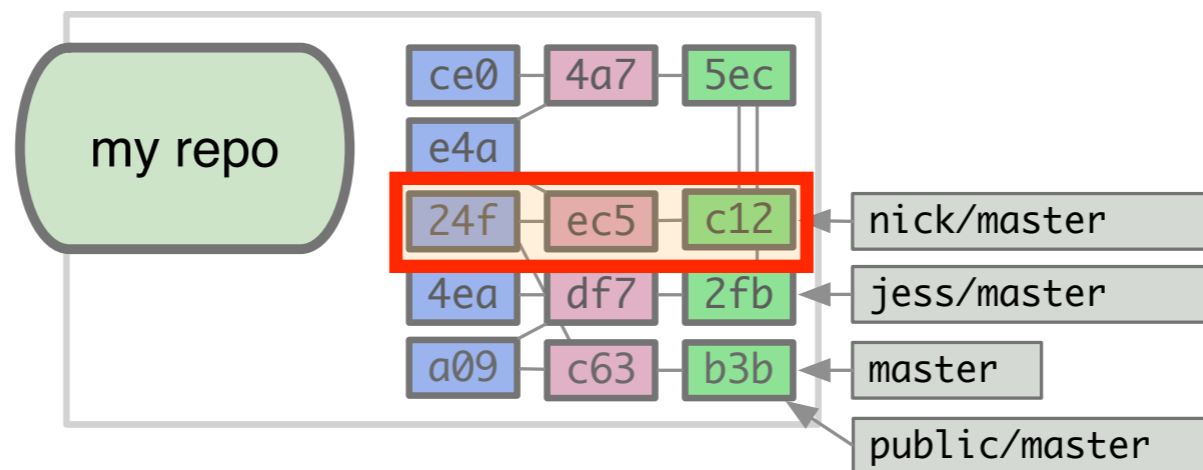
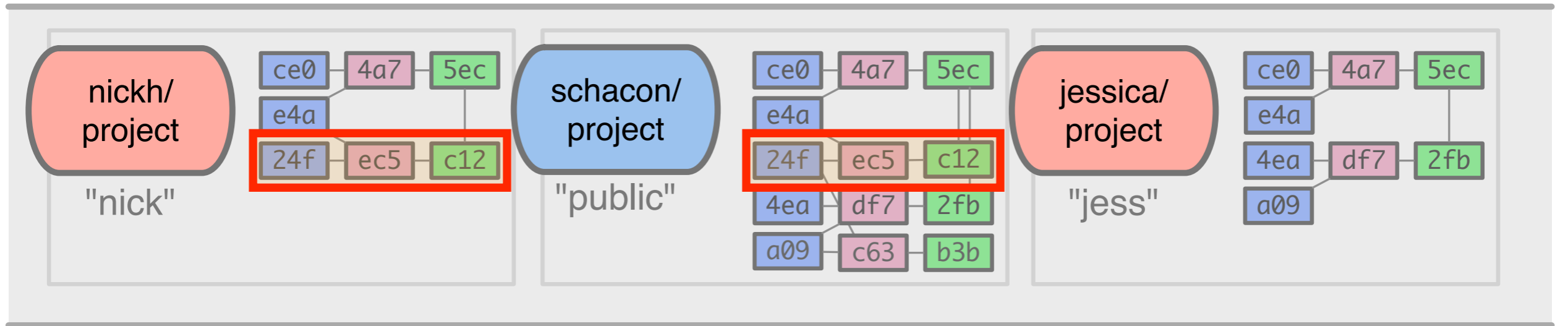
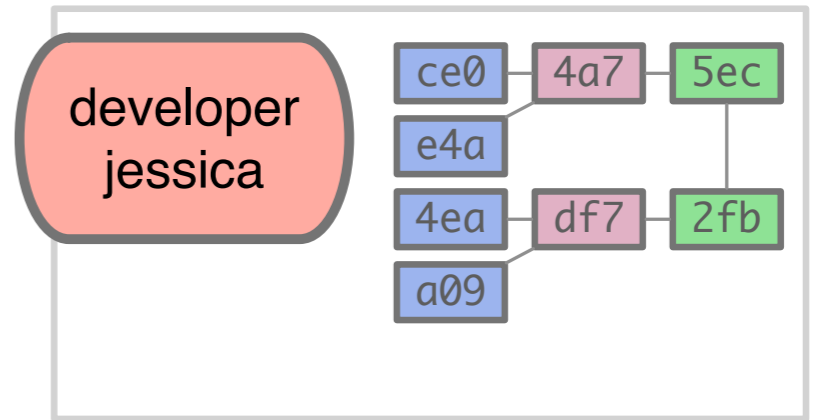
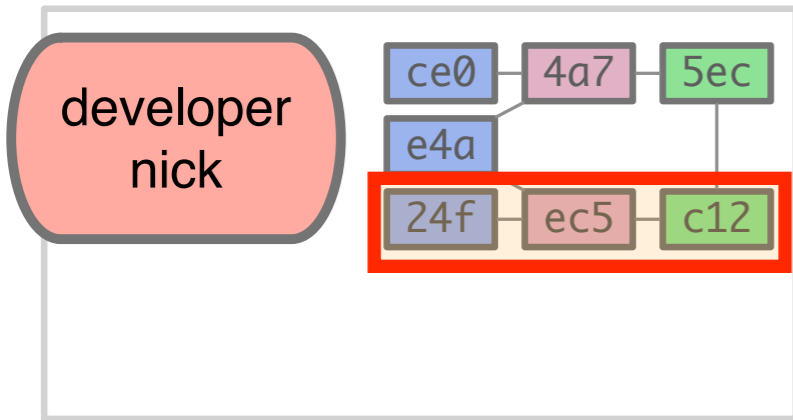


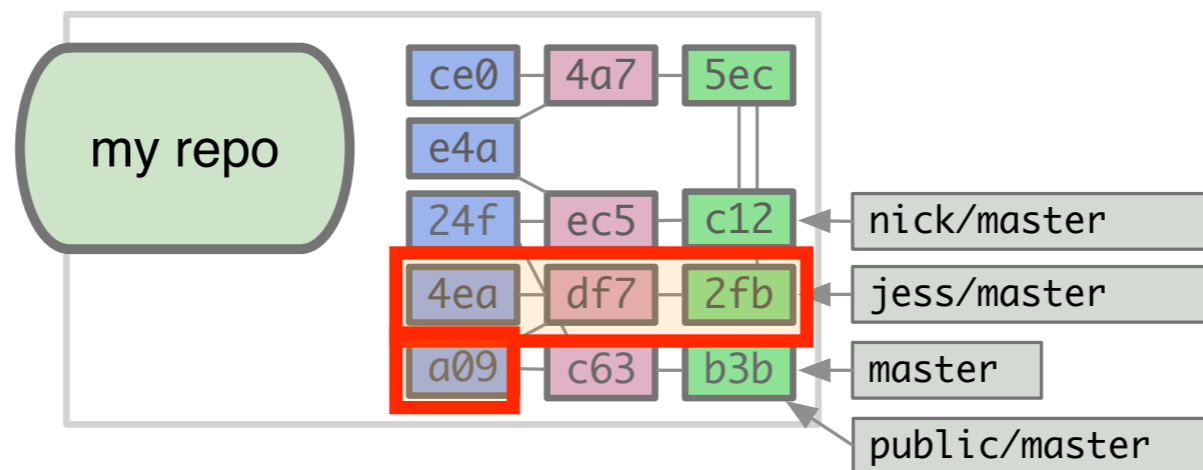
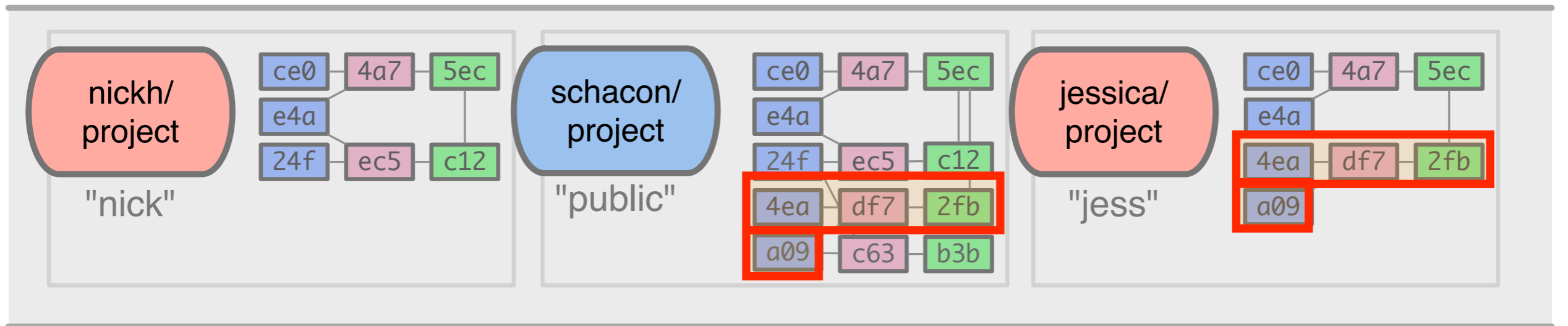
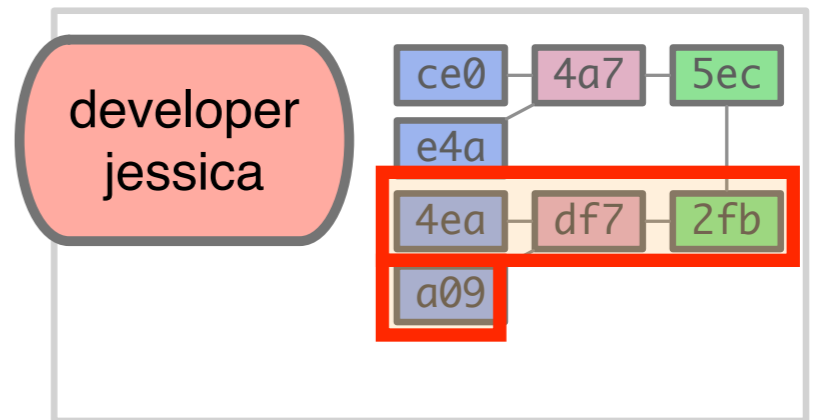
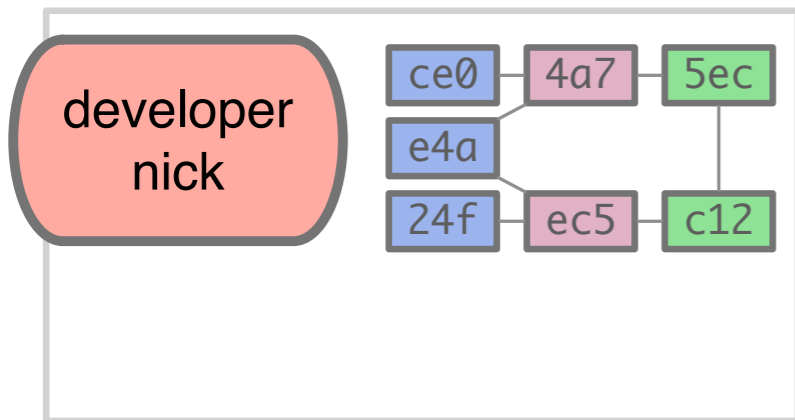
git merge nick jess

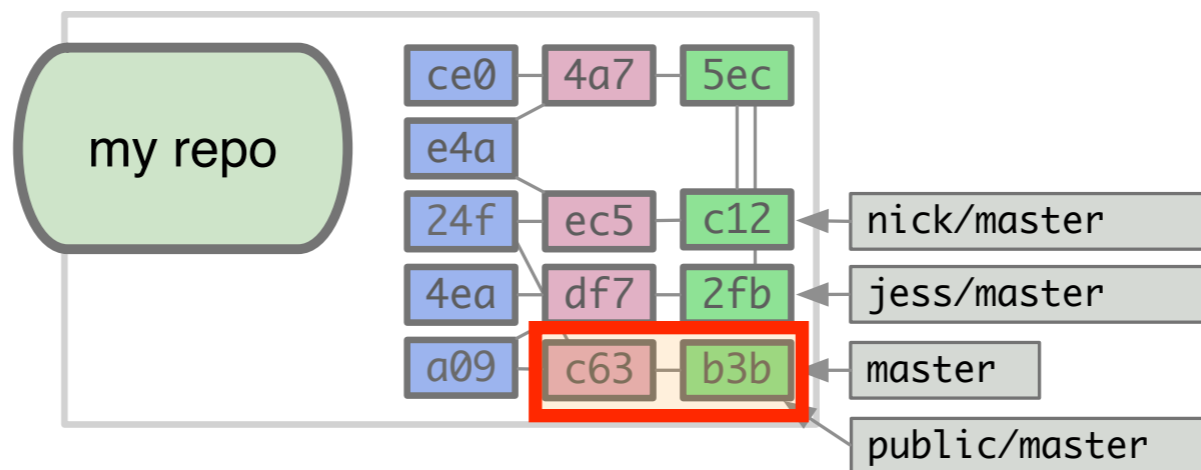
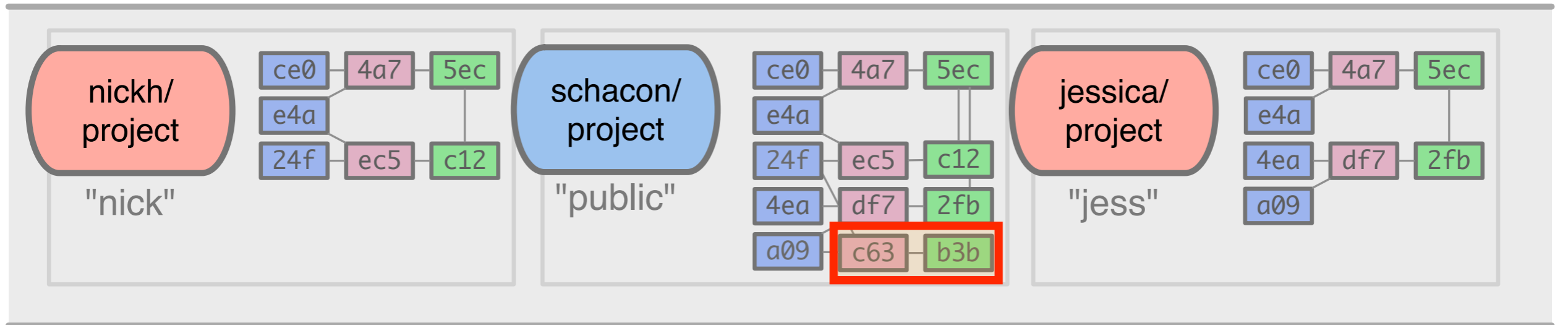
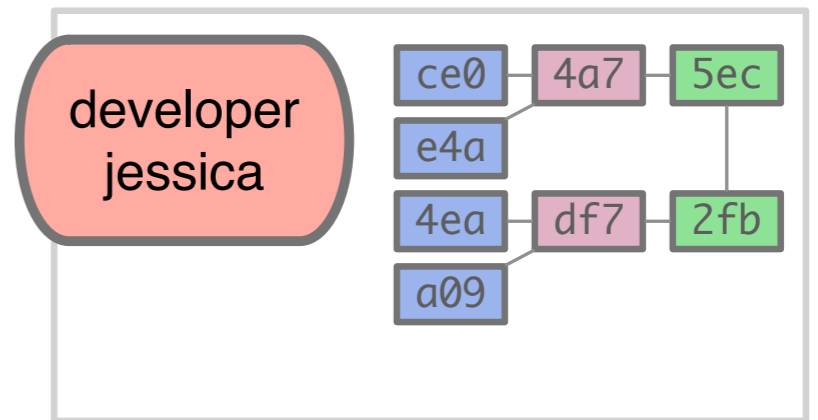
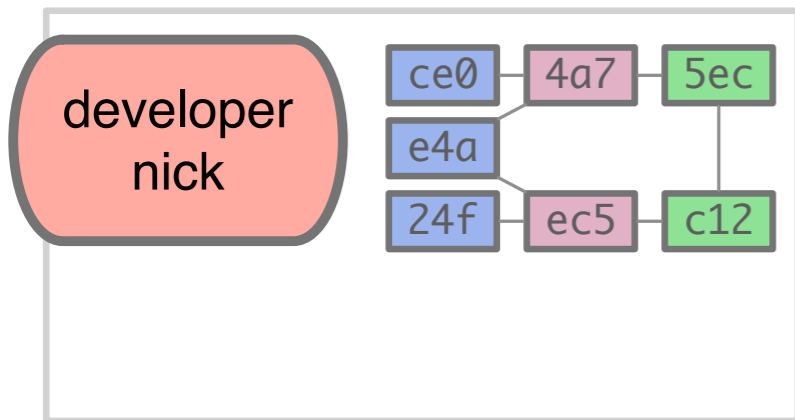


git push public





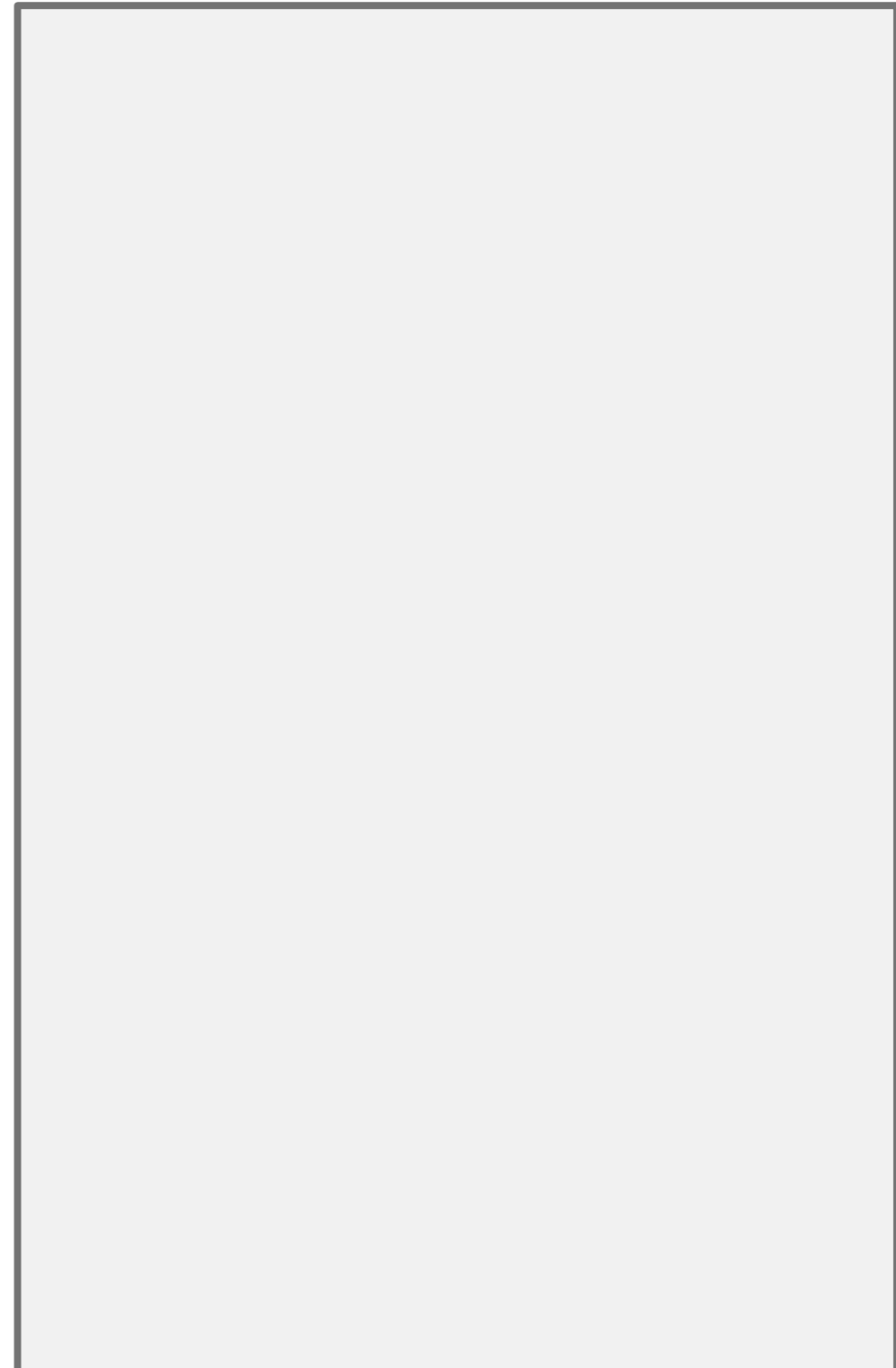
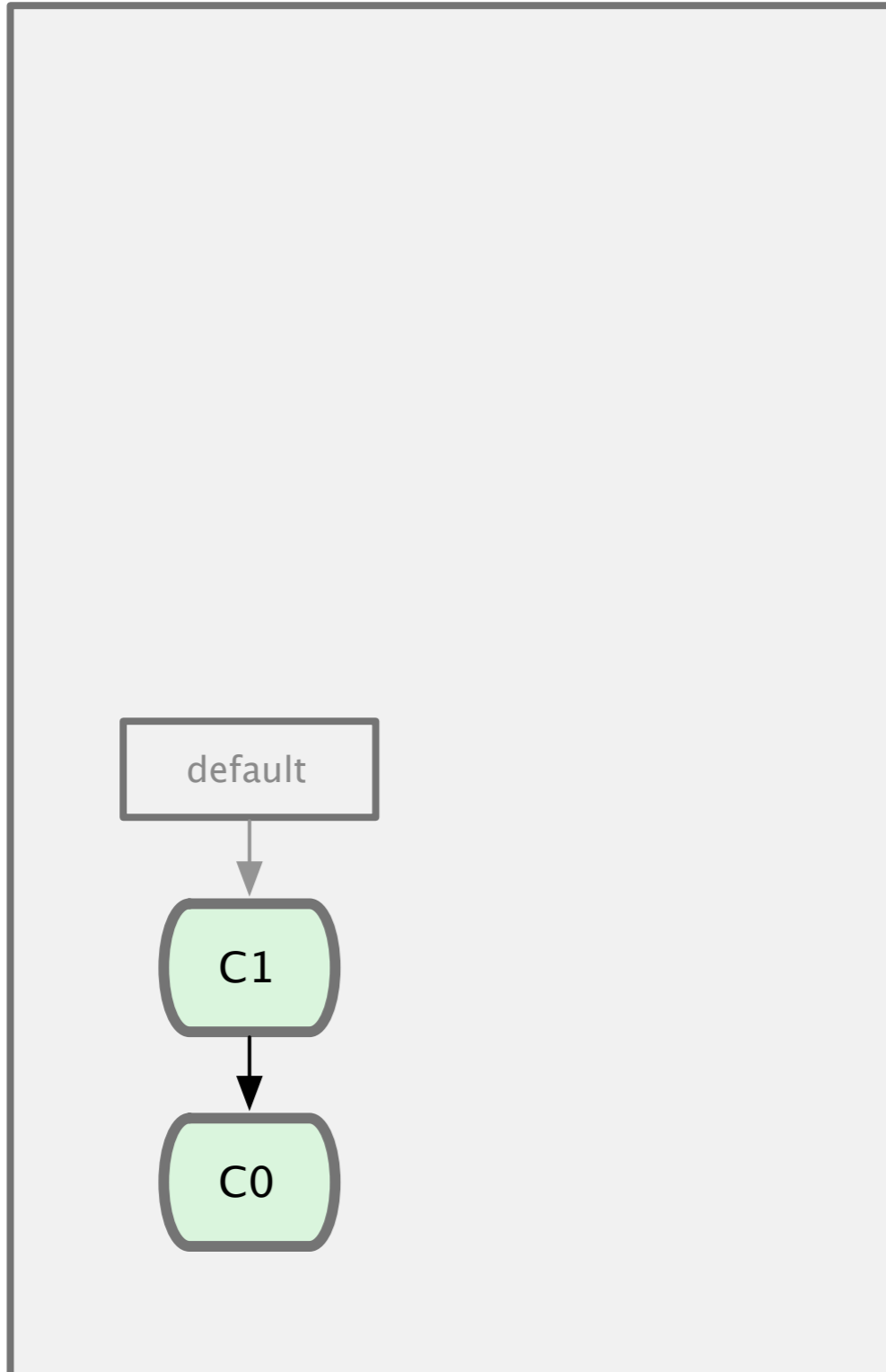




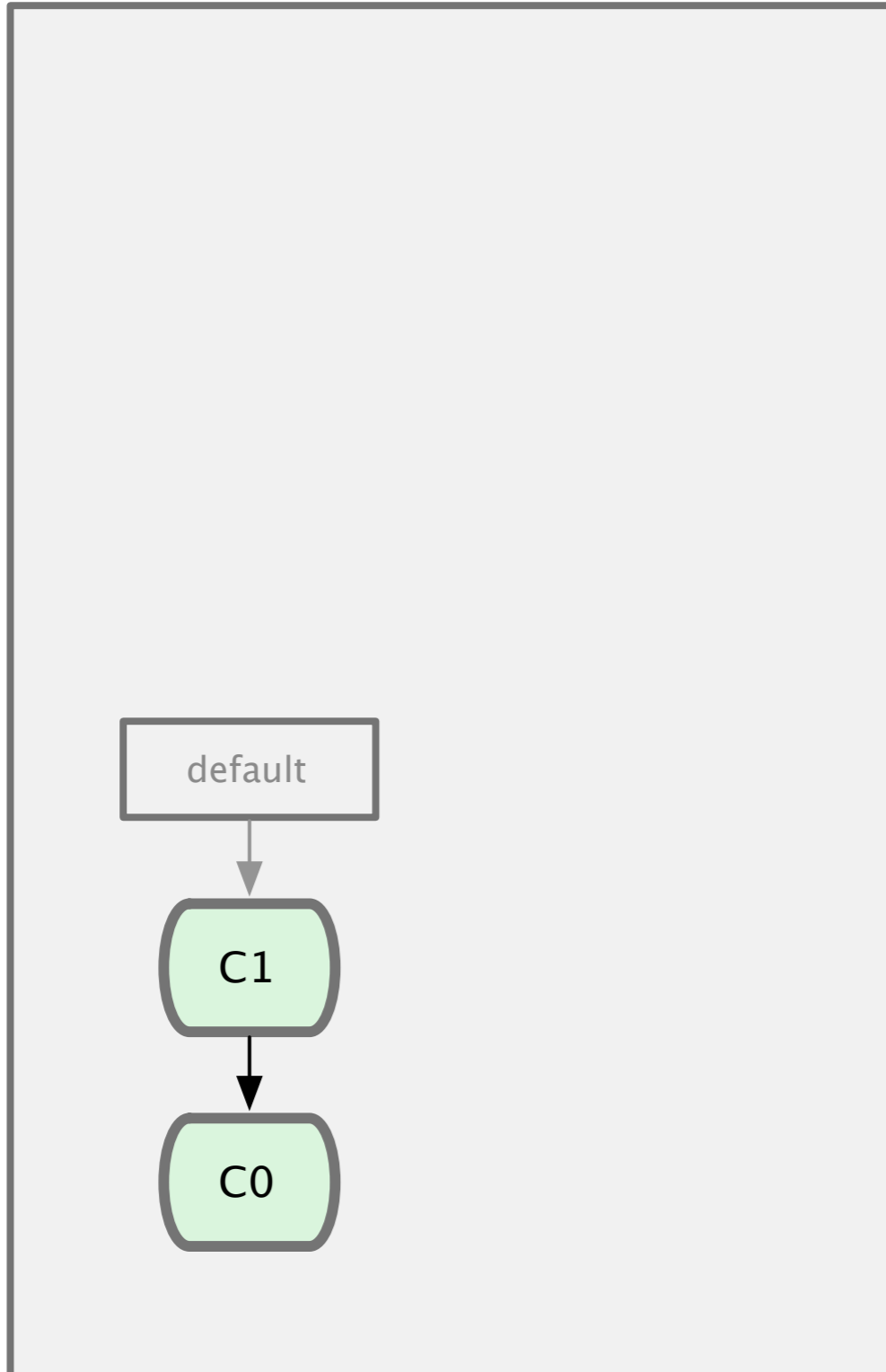
Remotes Are Branches

scott

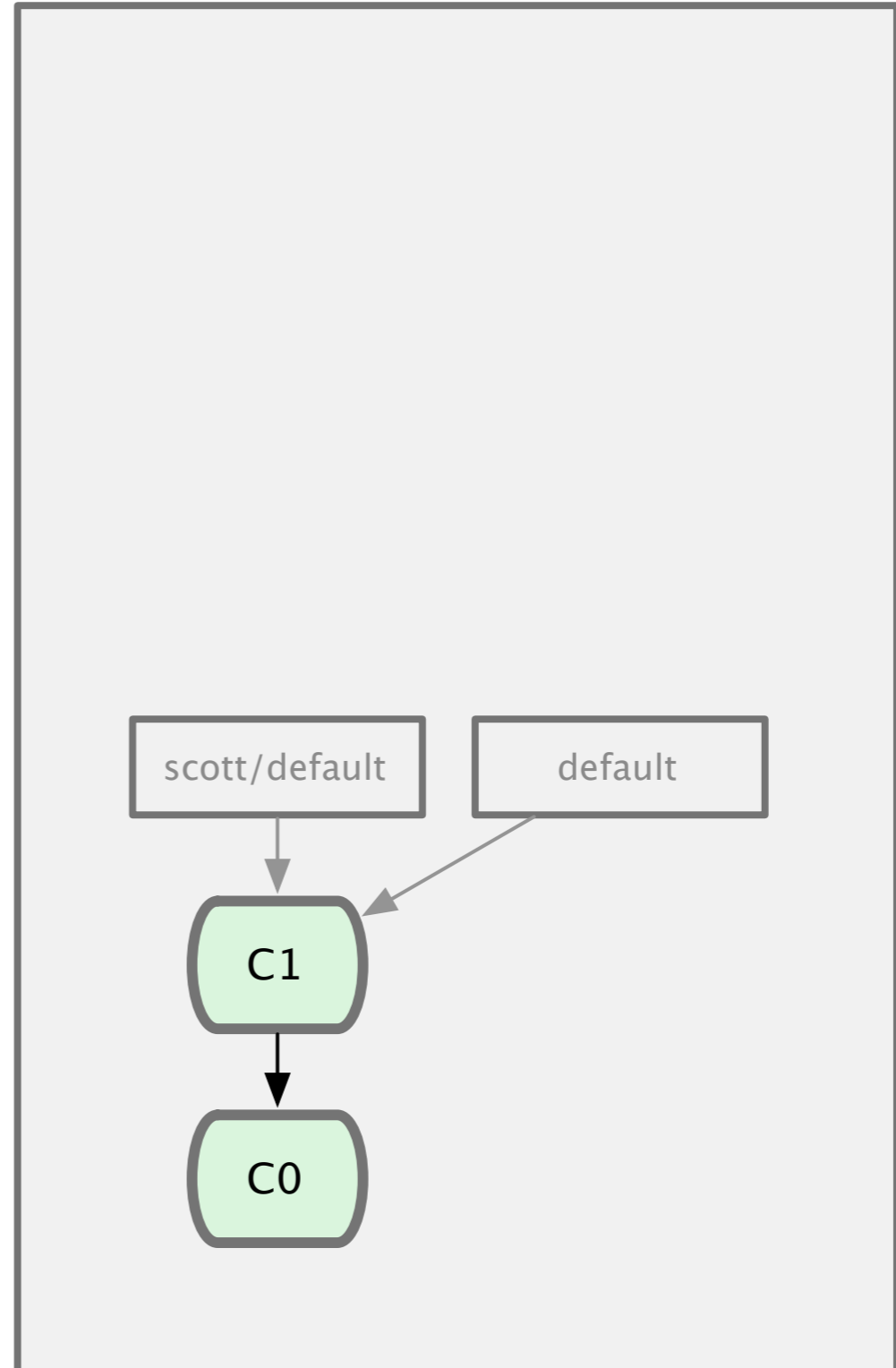
jessica



scott

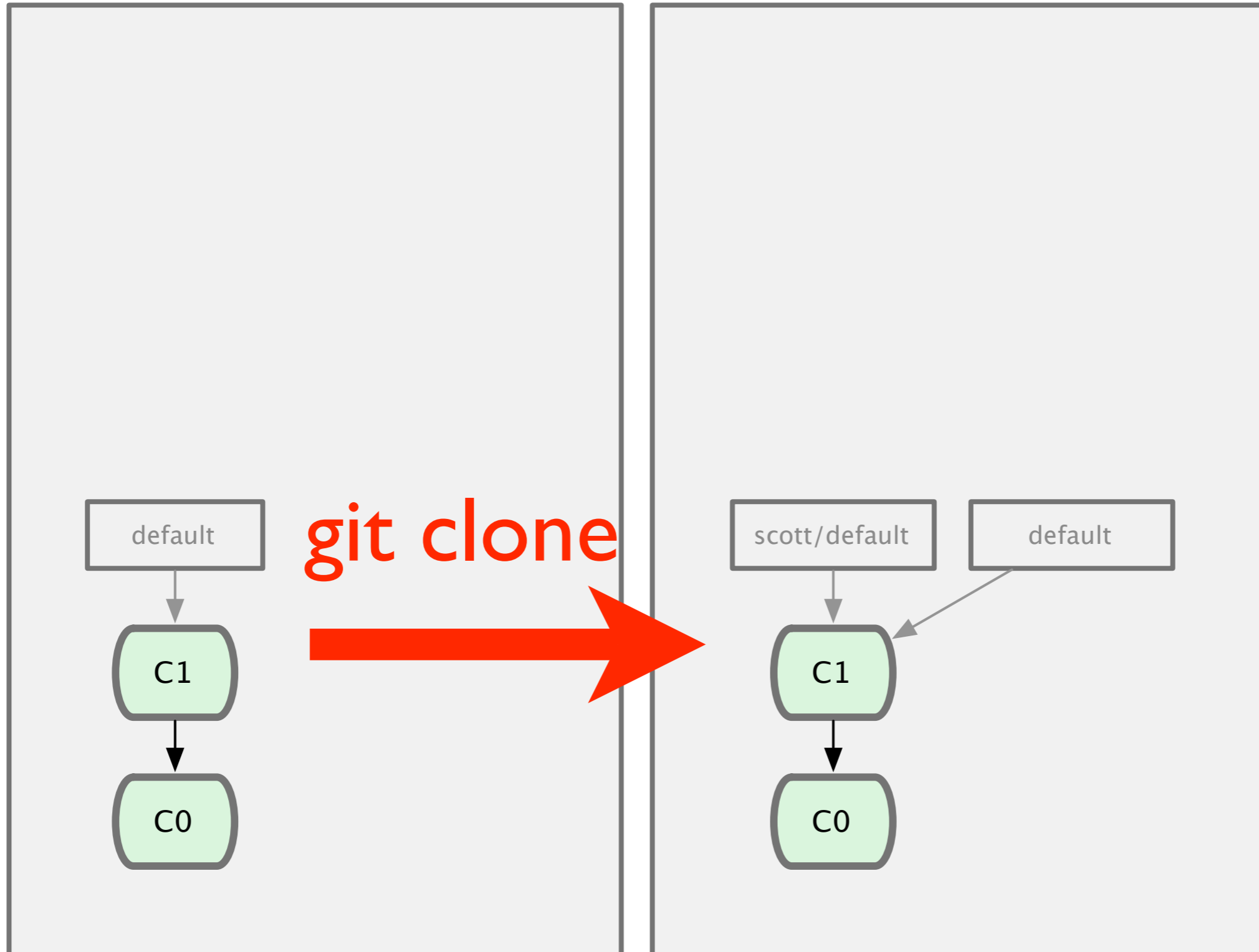


jessica

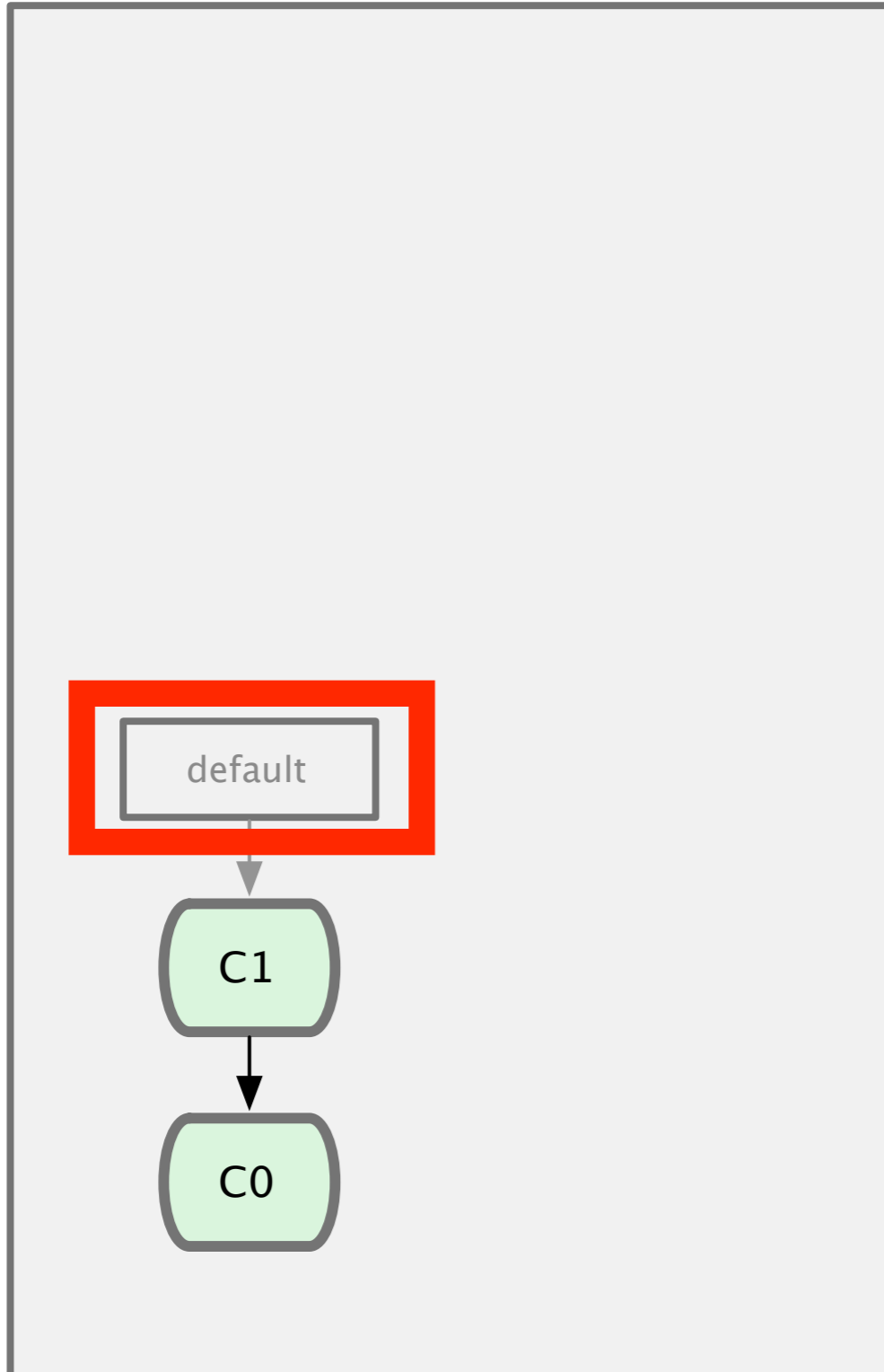


scott

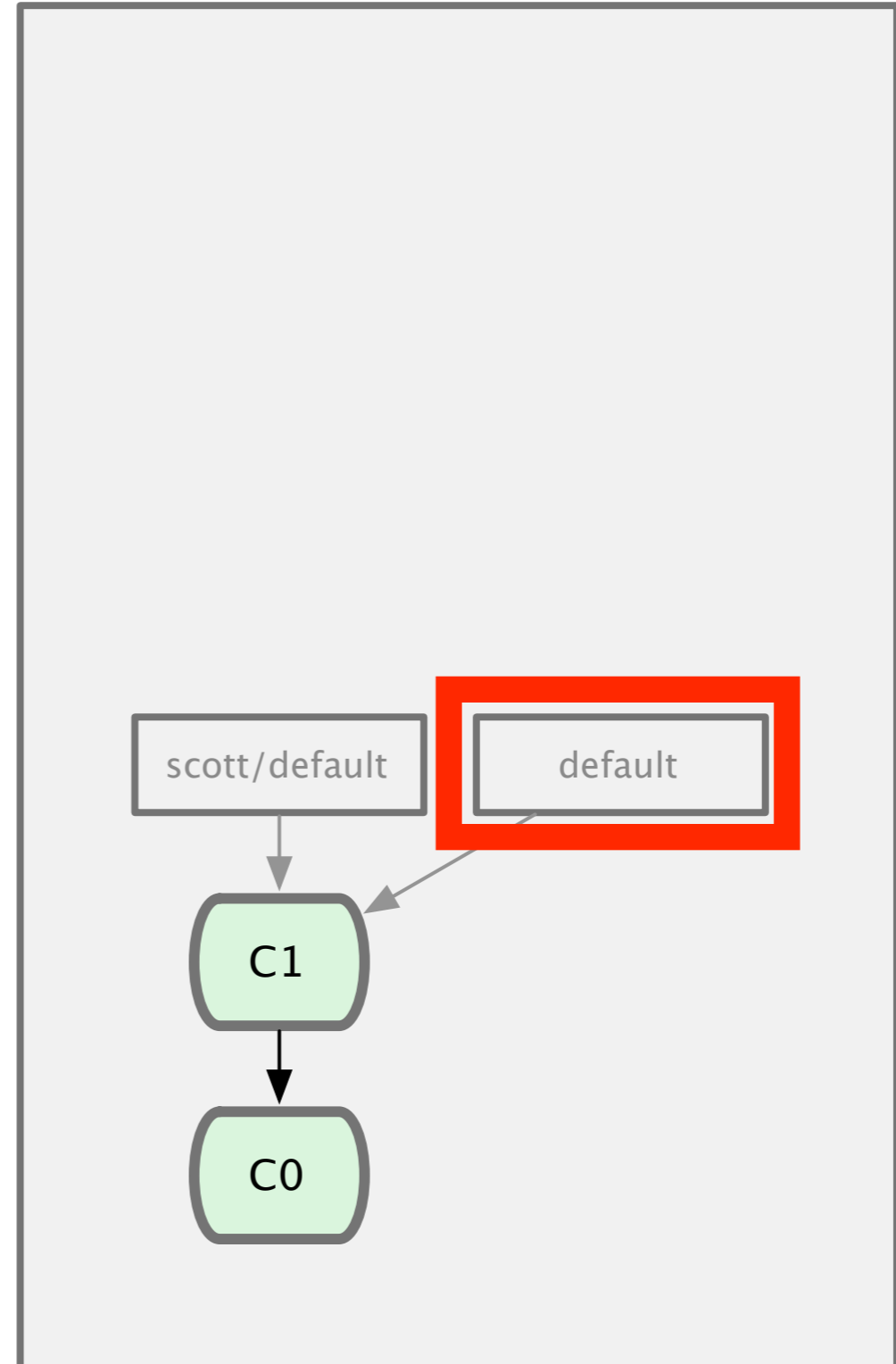
jessica



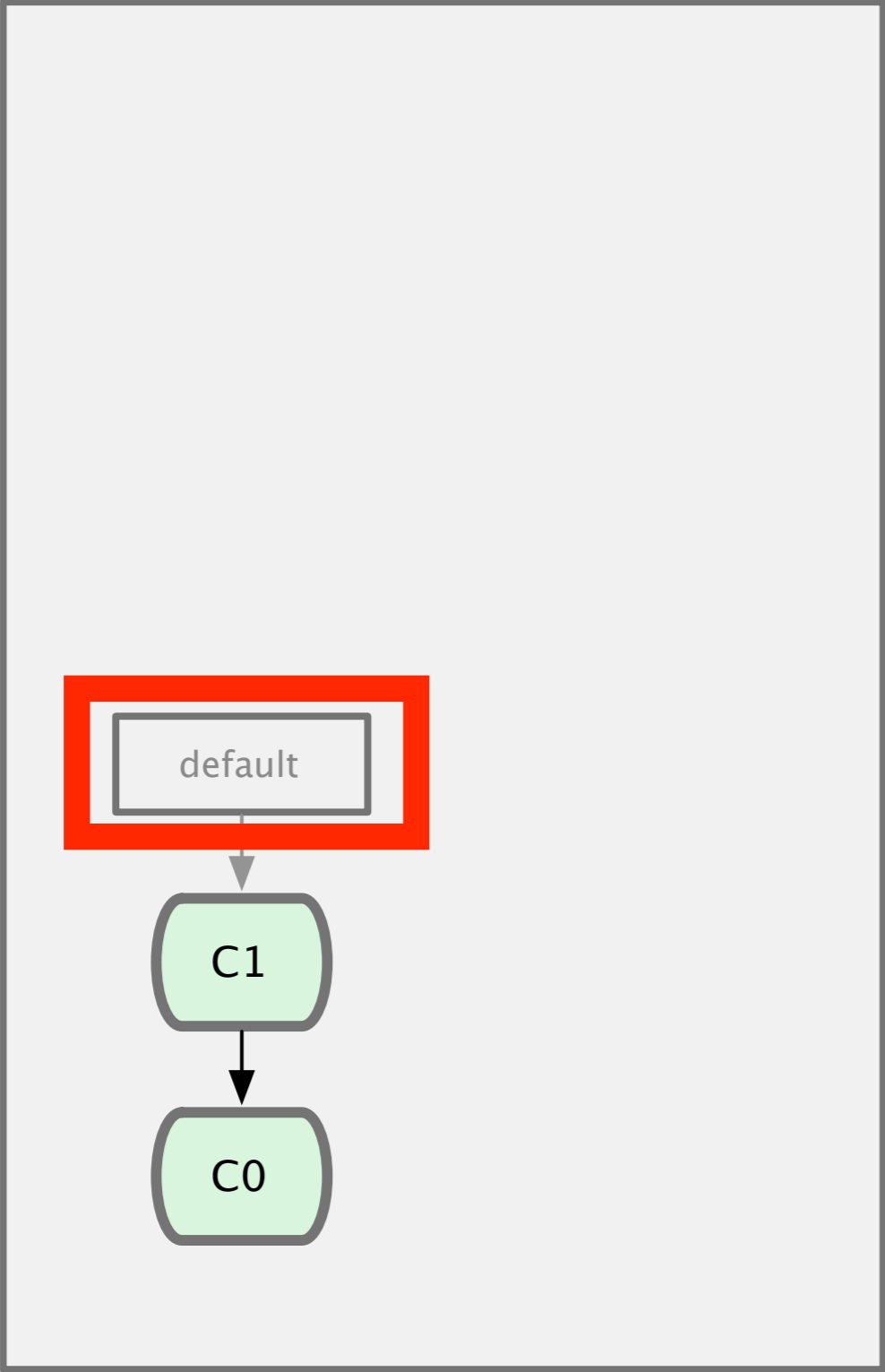
scott



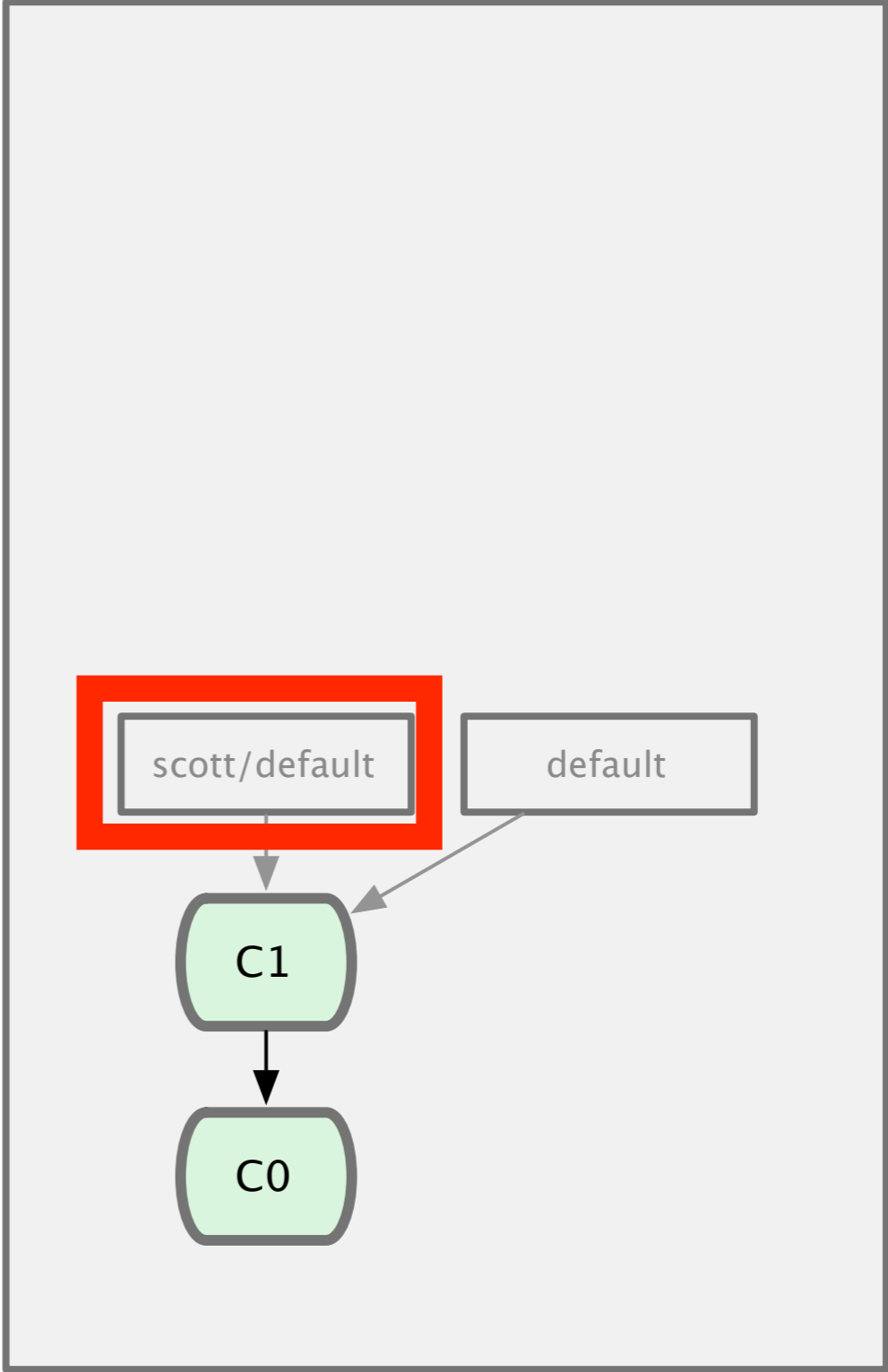
jessica



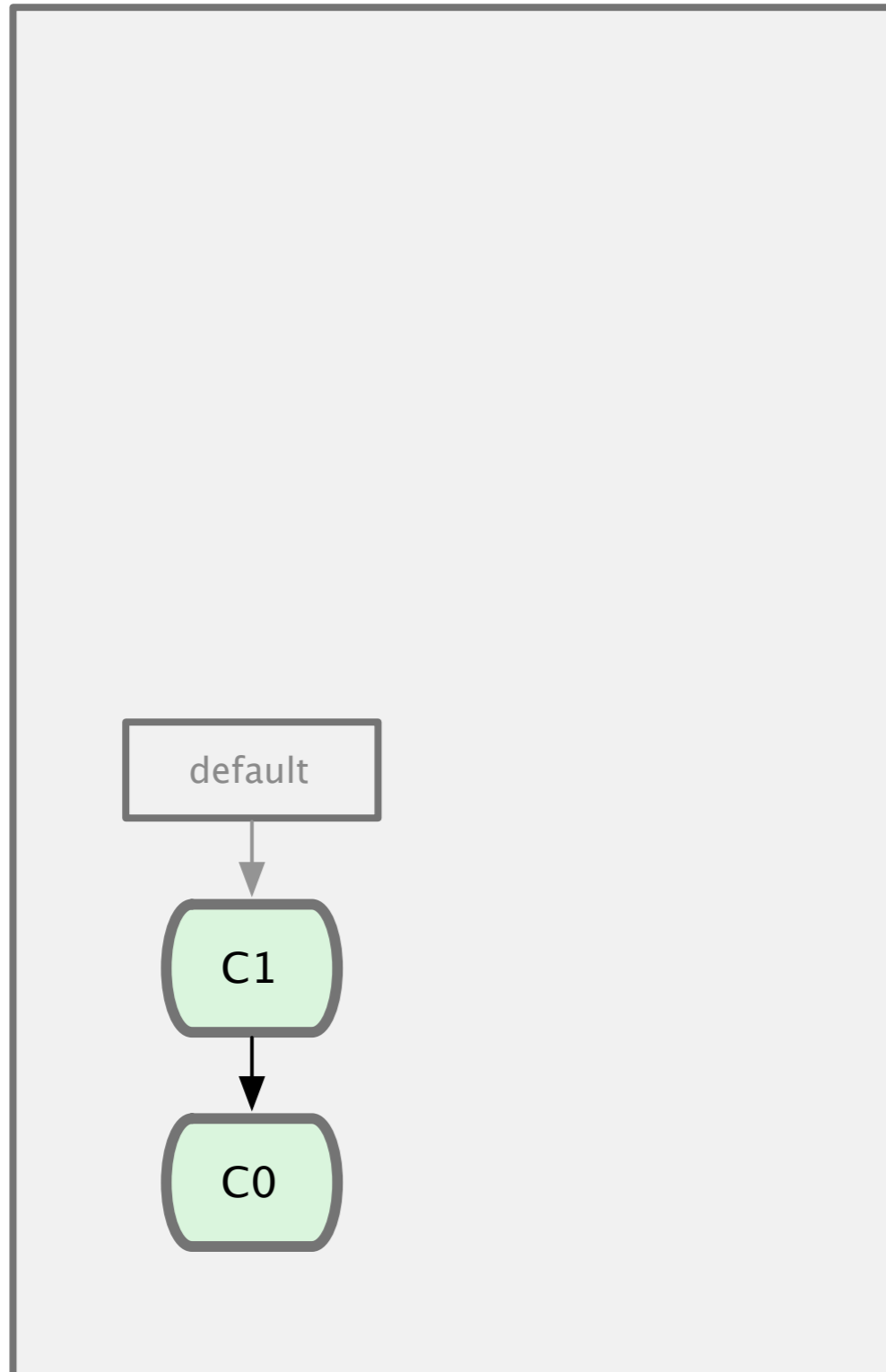
scott



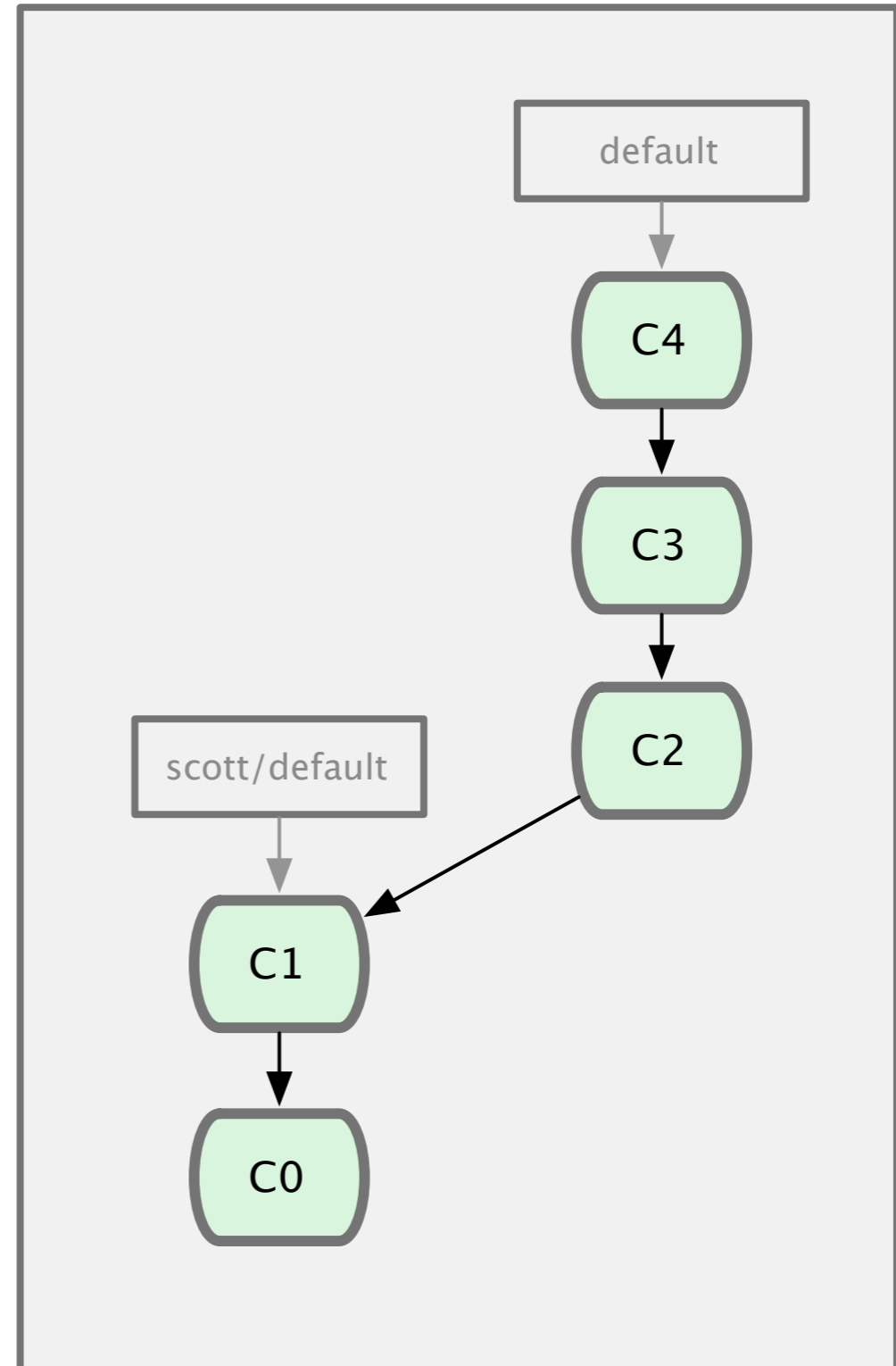
jessica



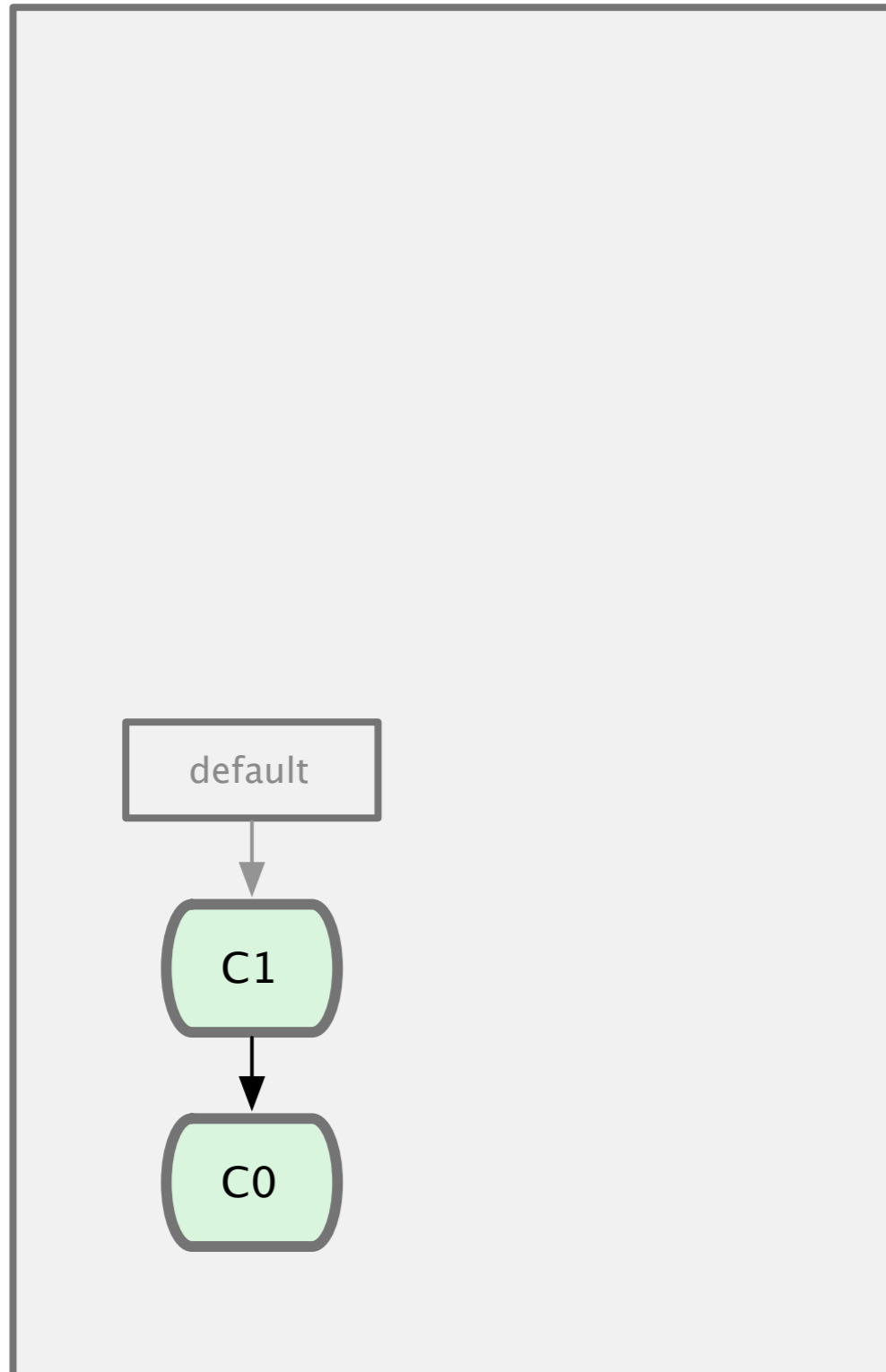
scott



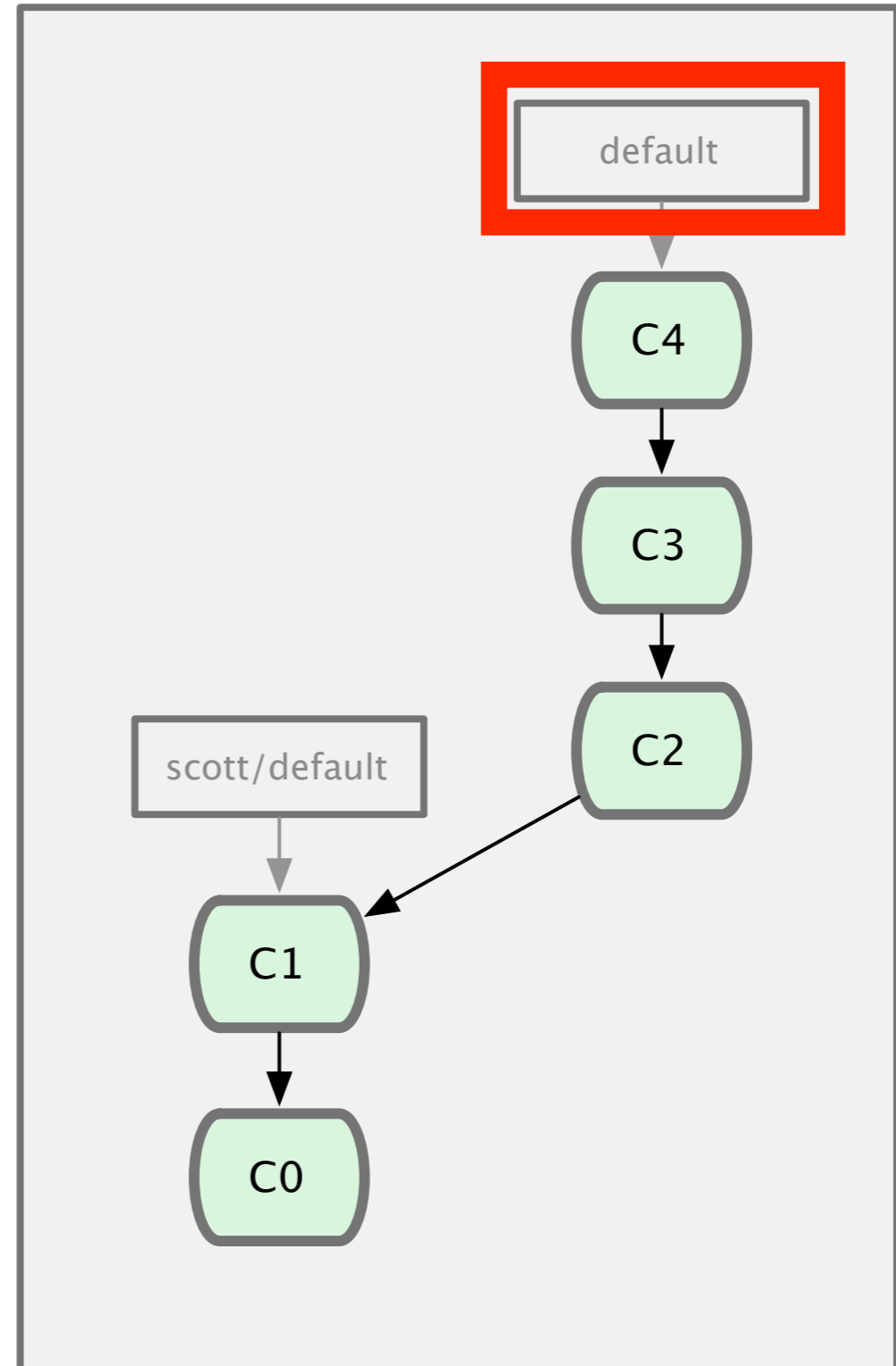
jessica



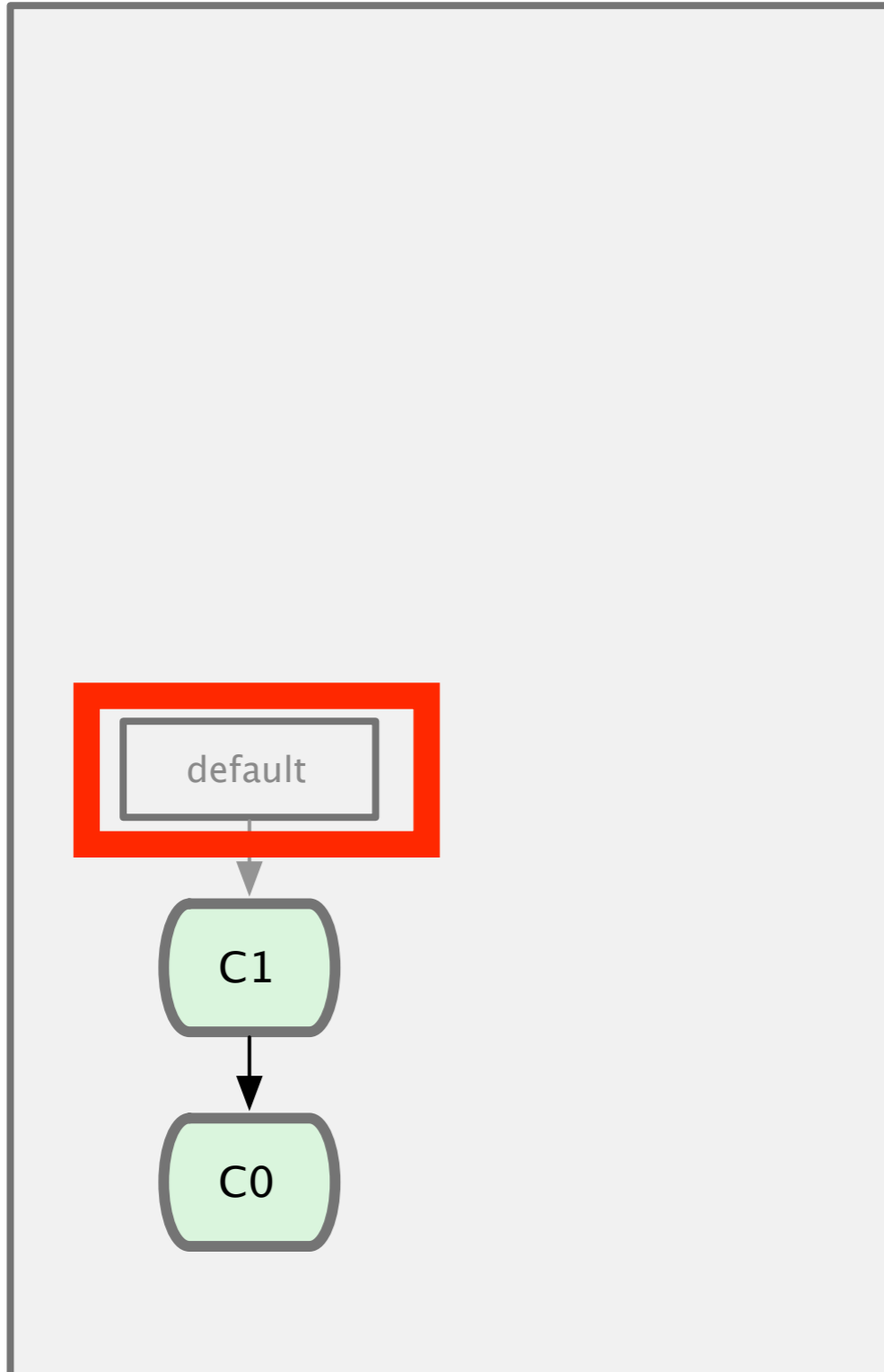
scott



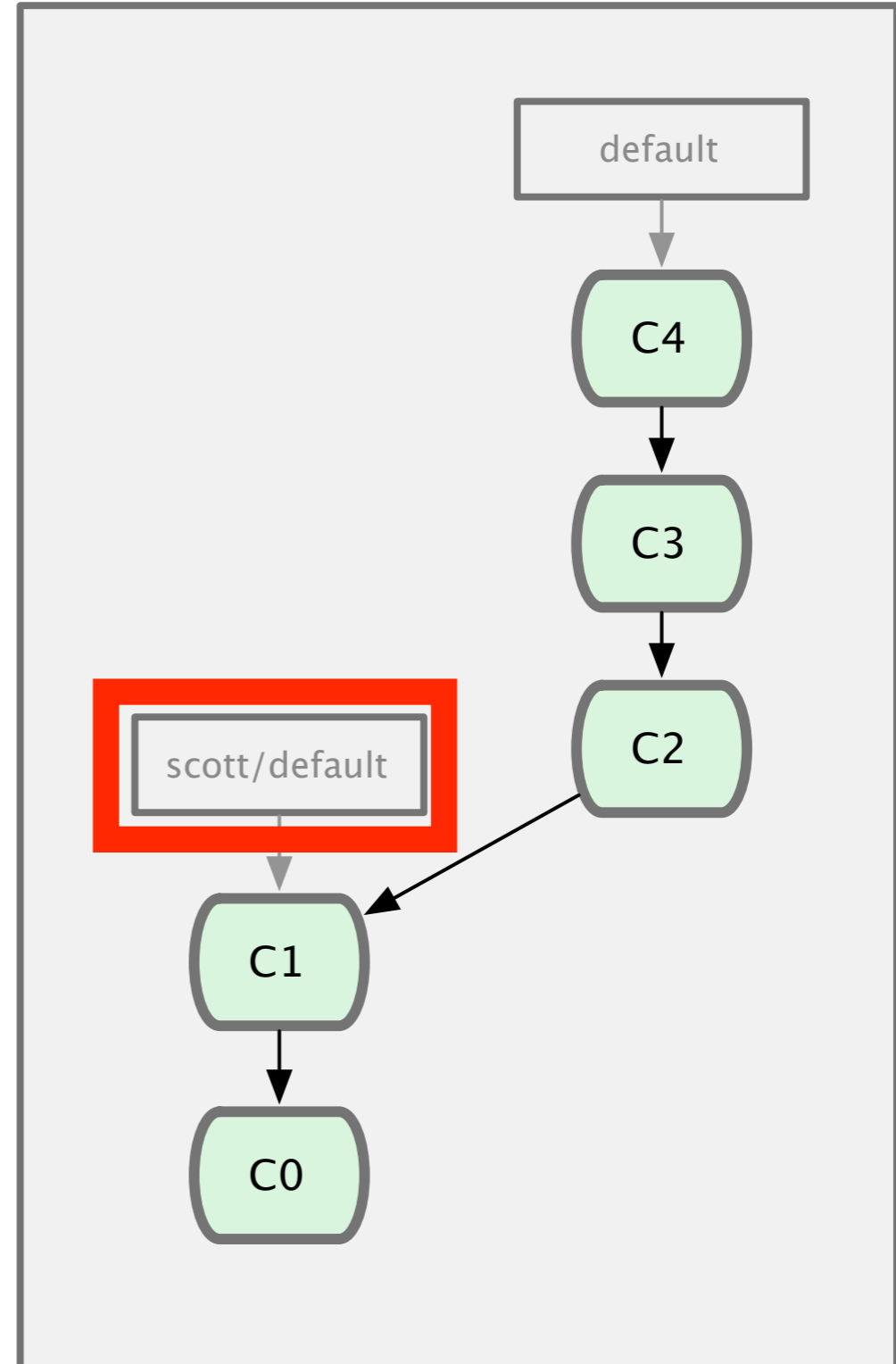
jessica



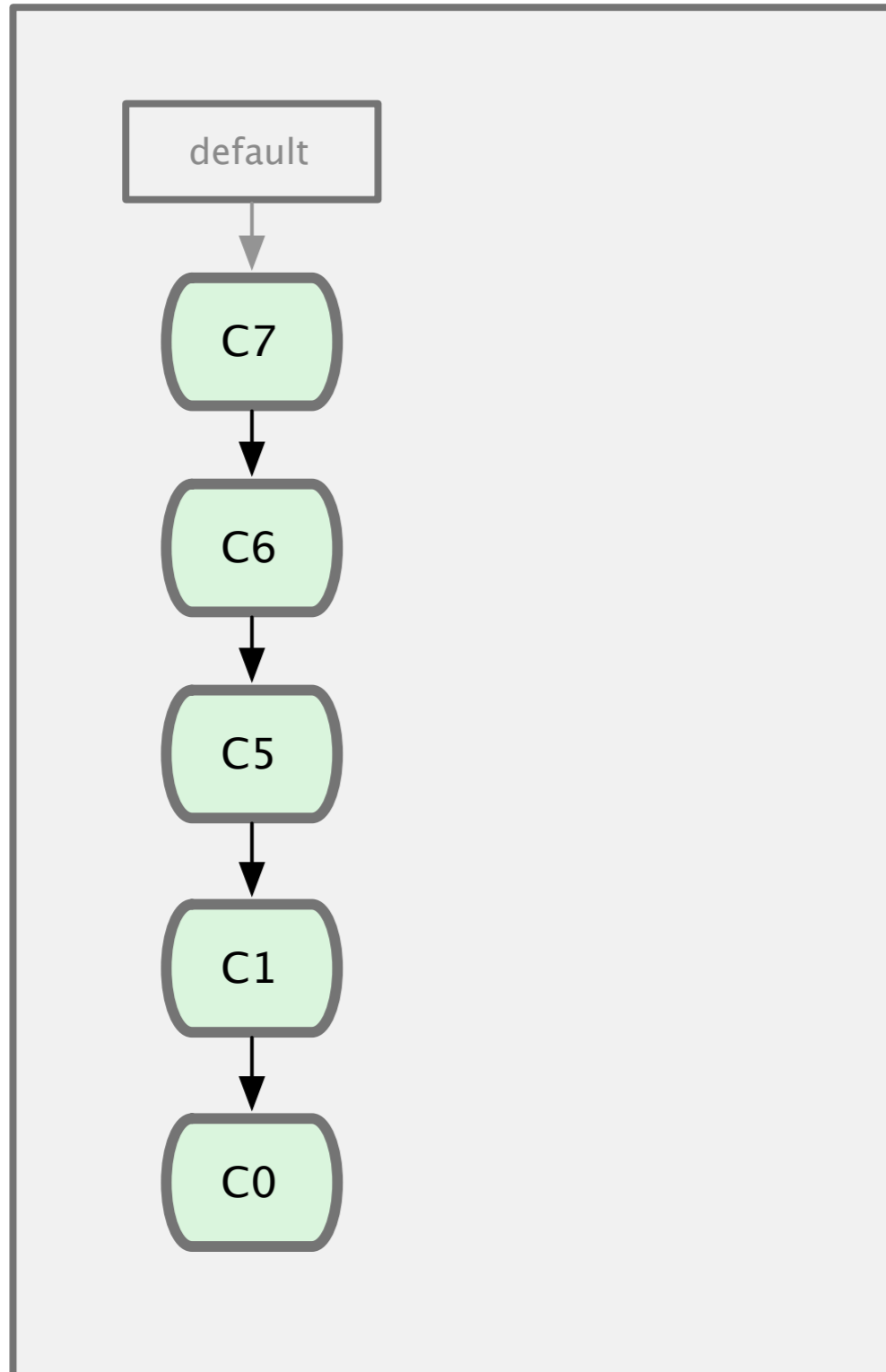
scott



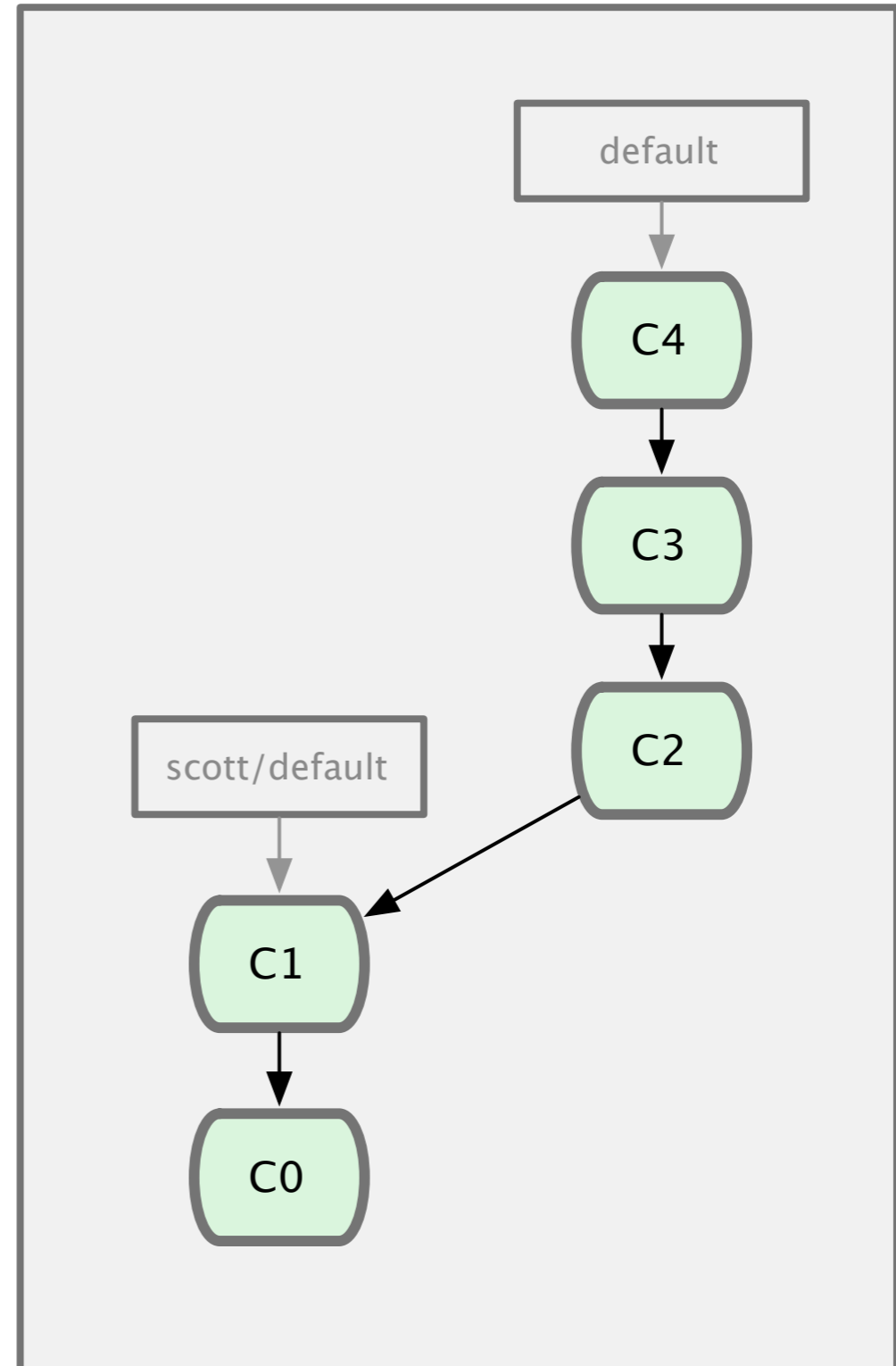
jessica



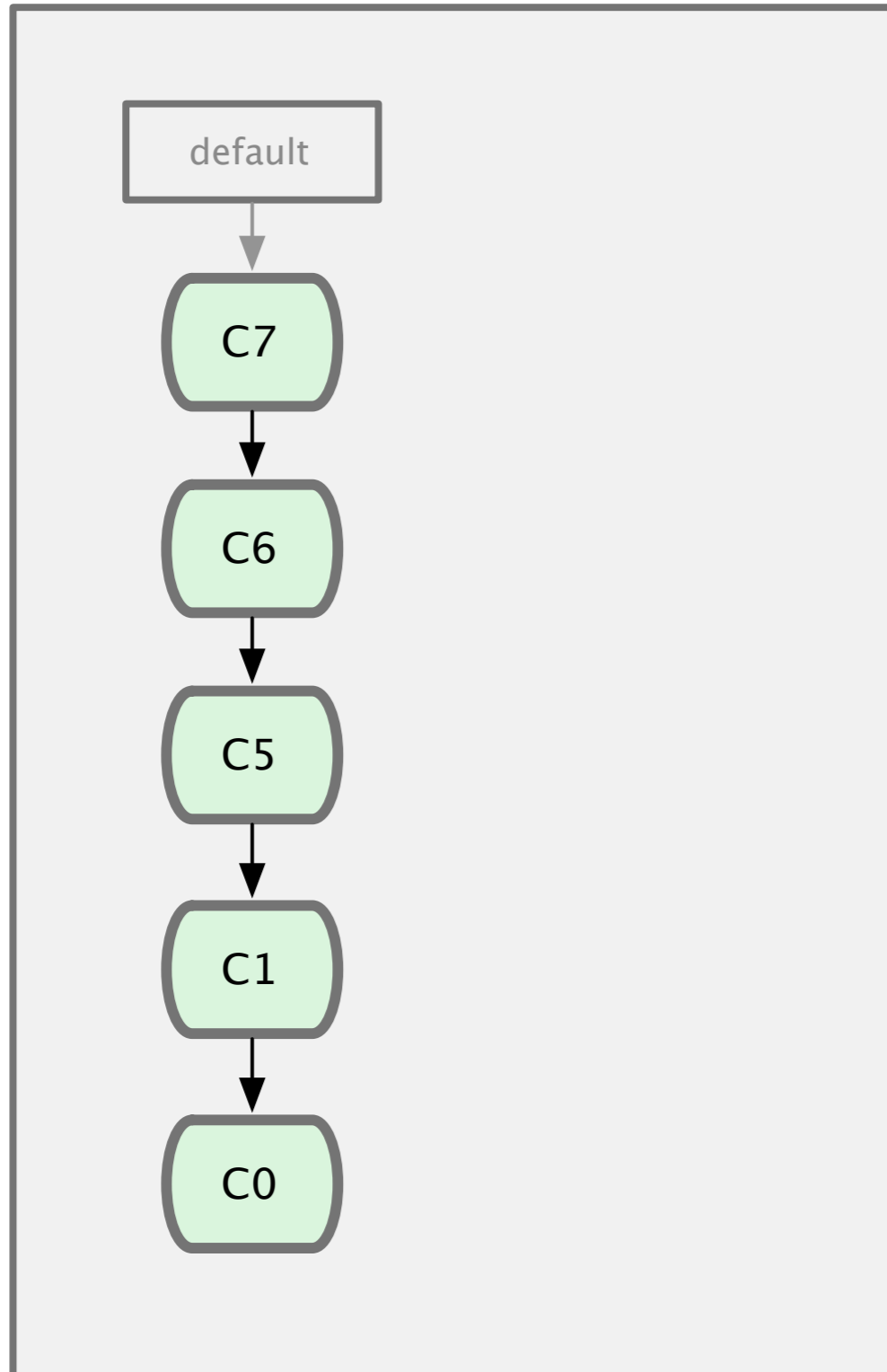
scott



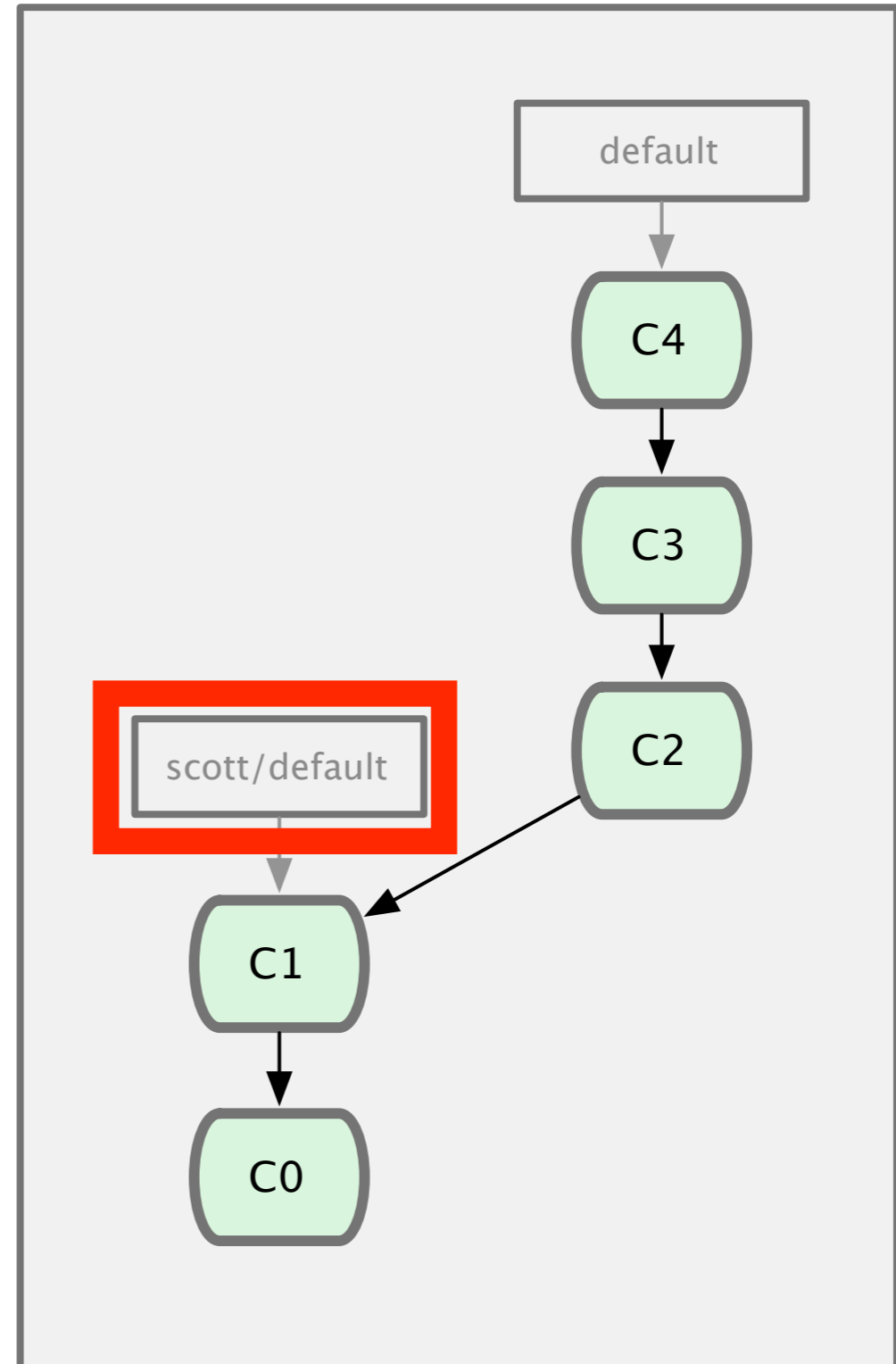
jessica



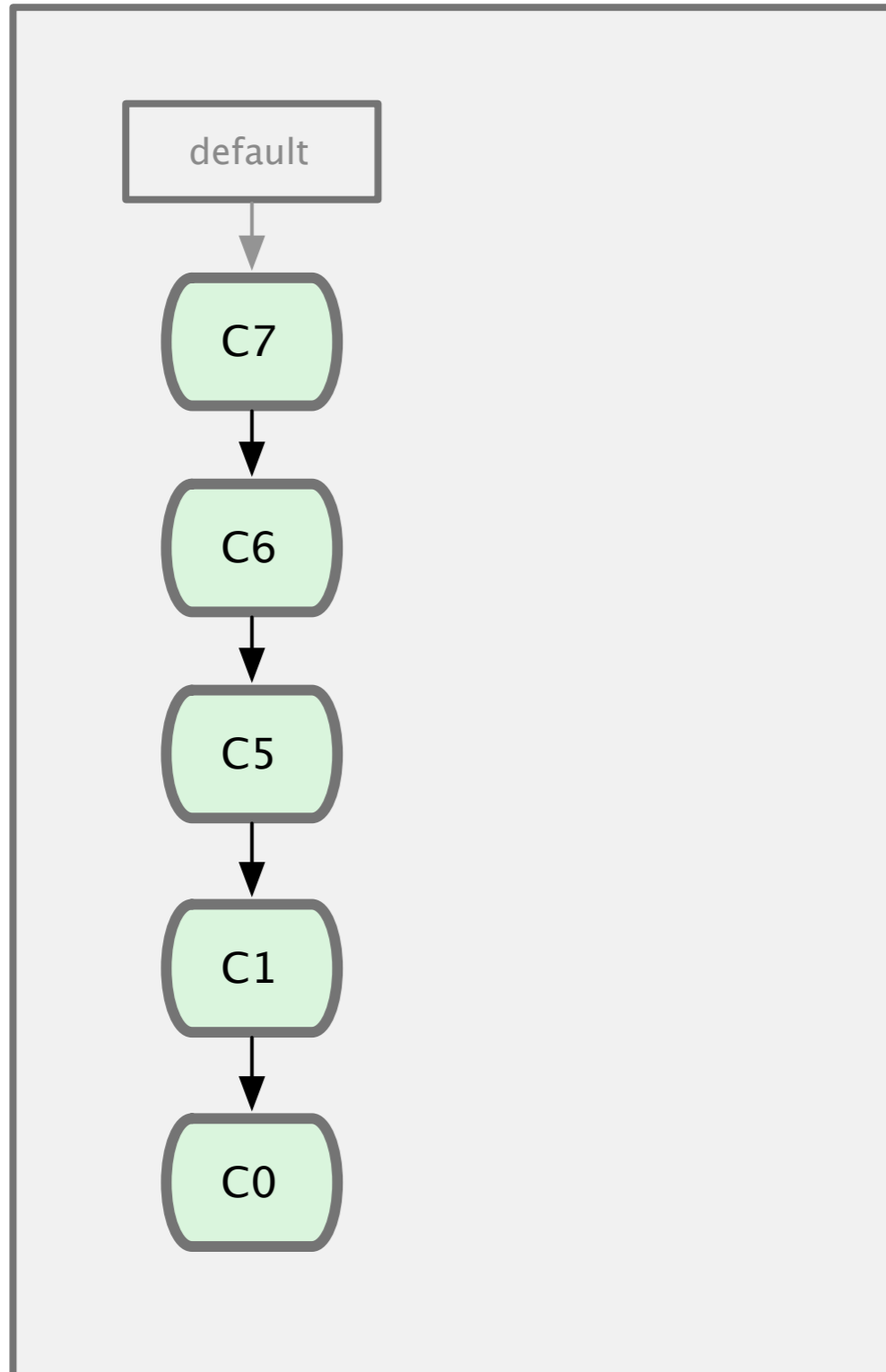
scott



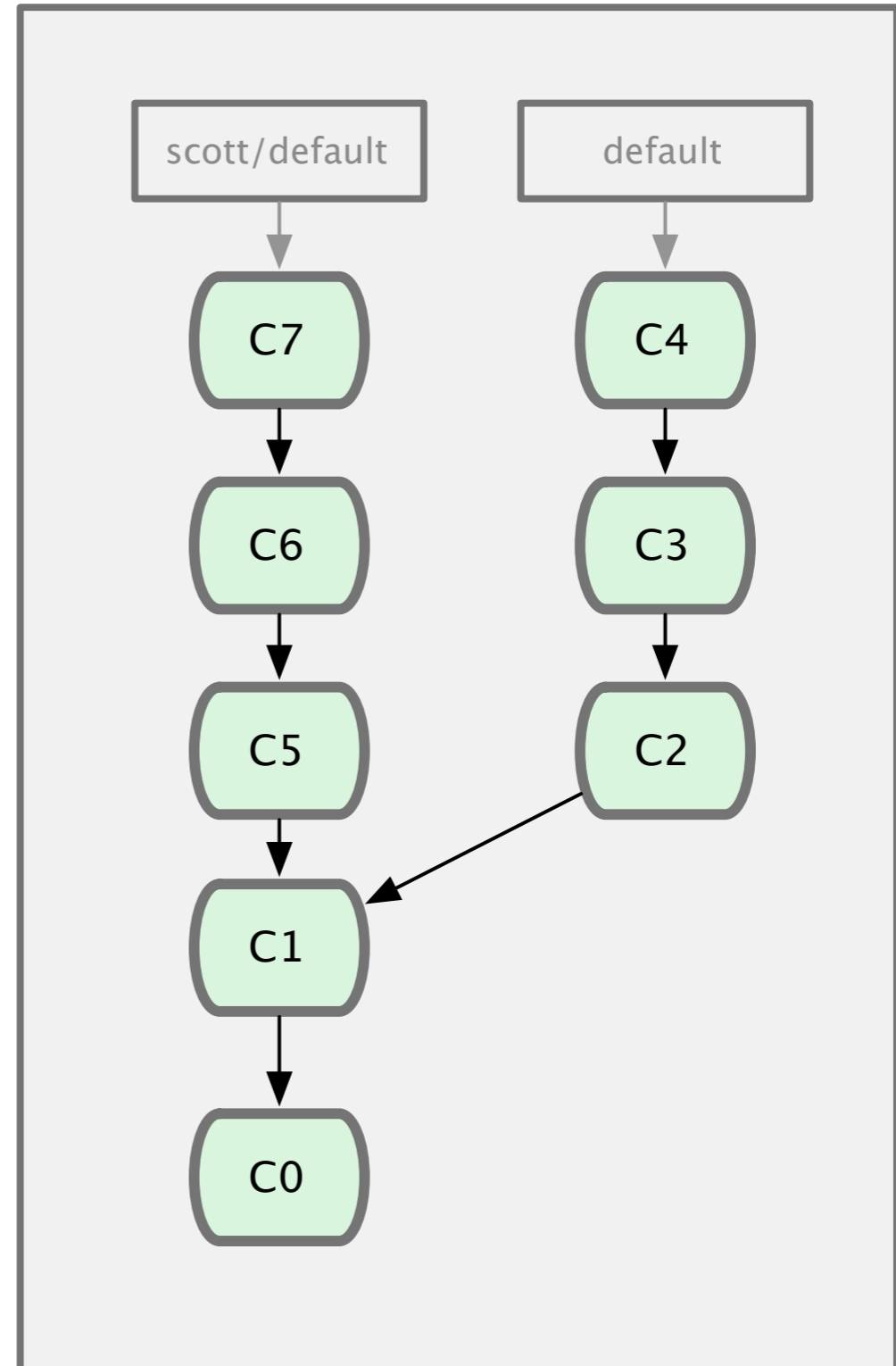
jessica



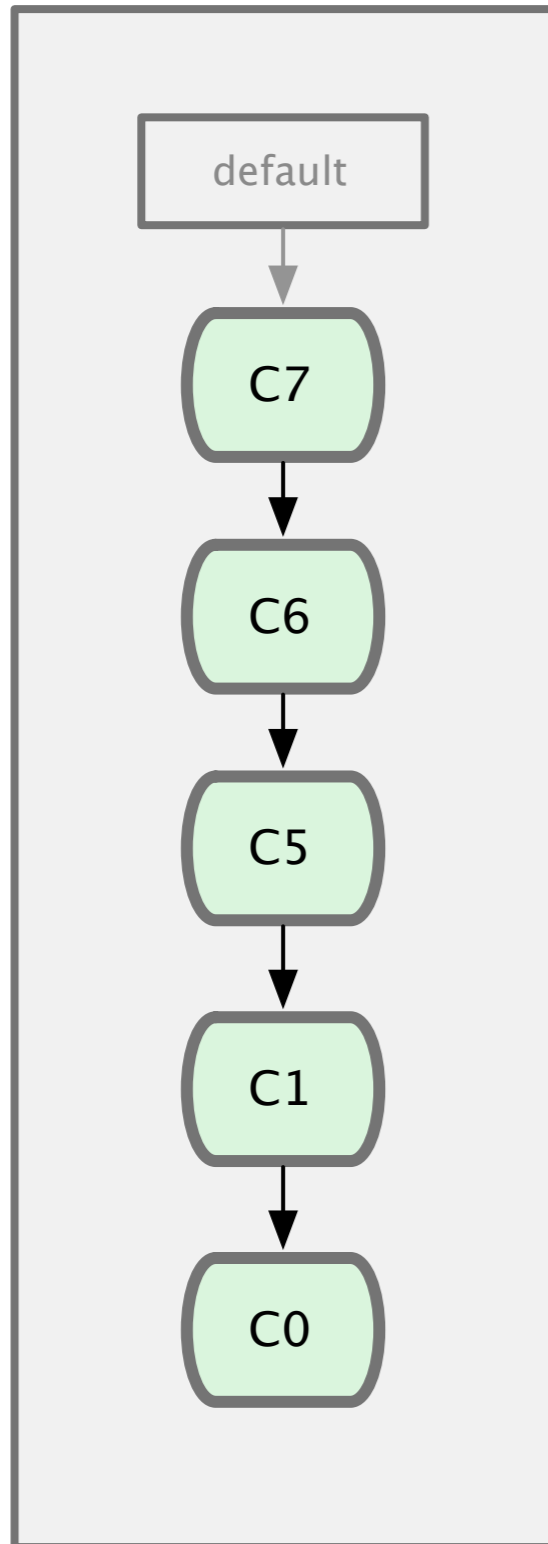
scott



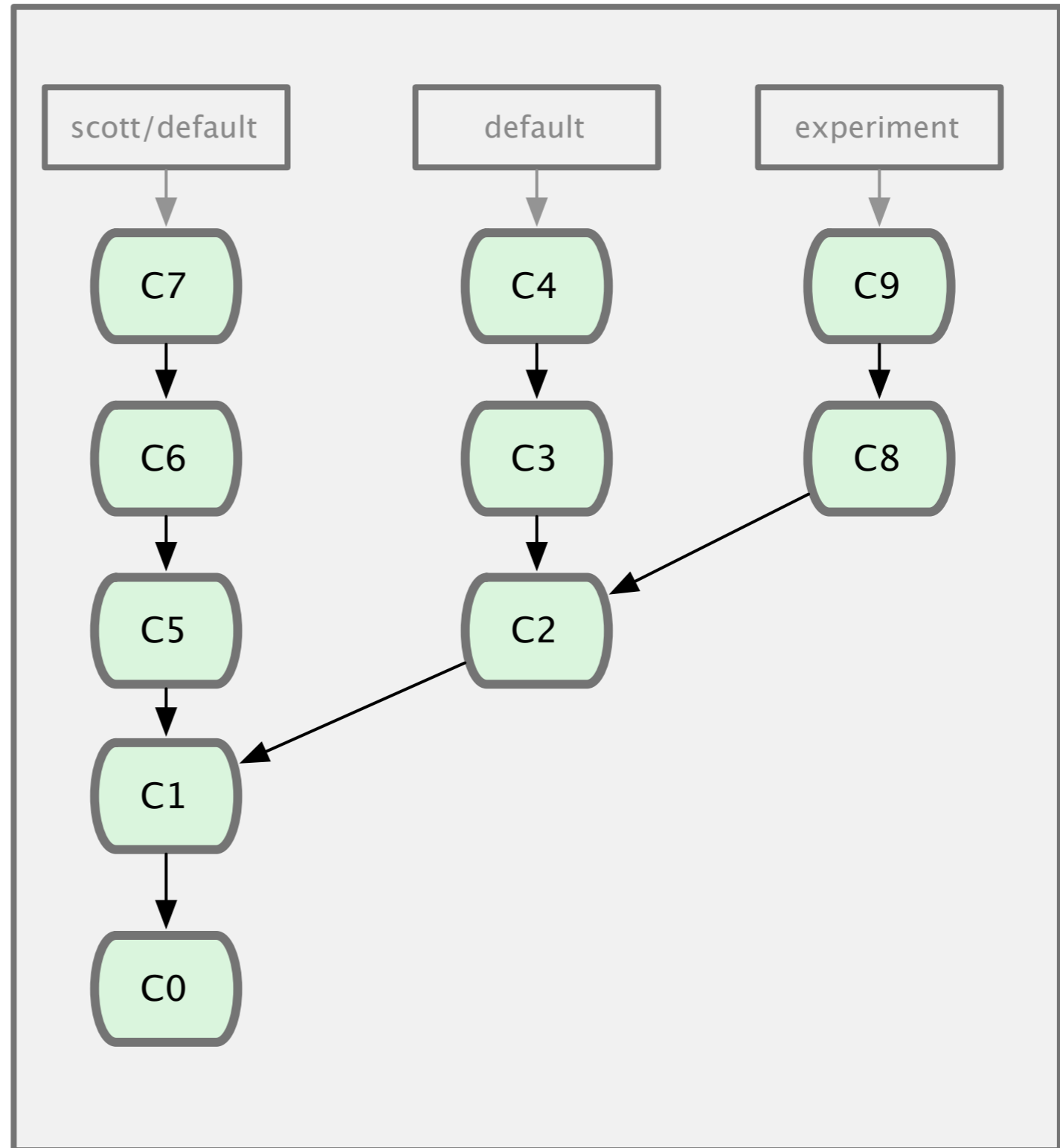
jessica



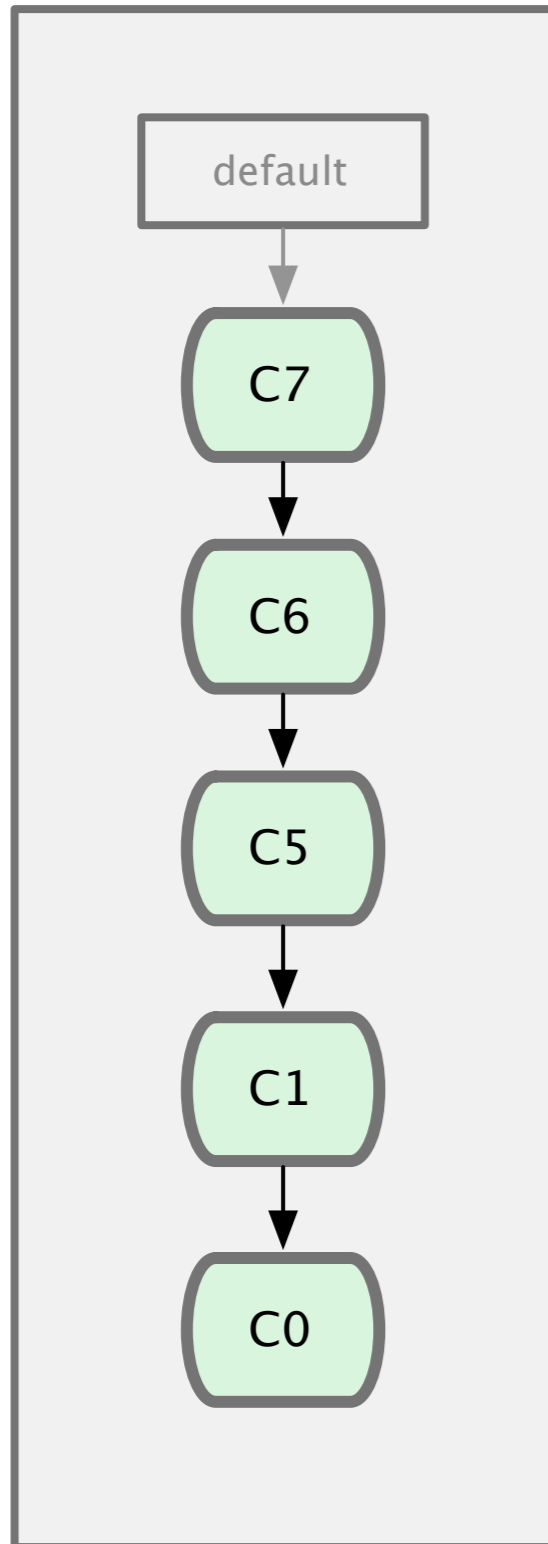
scott



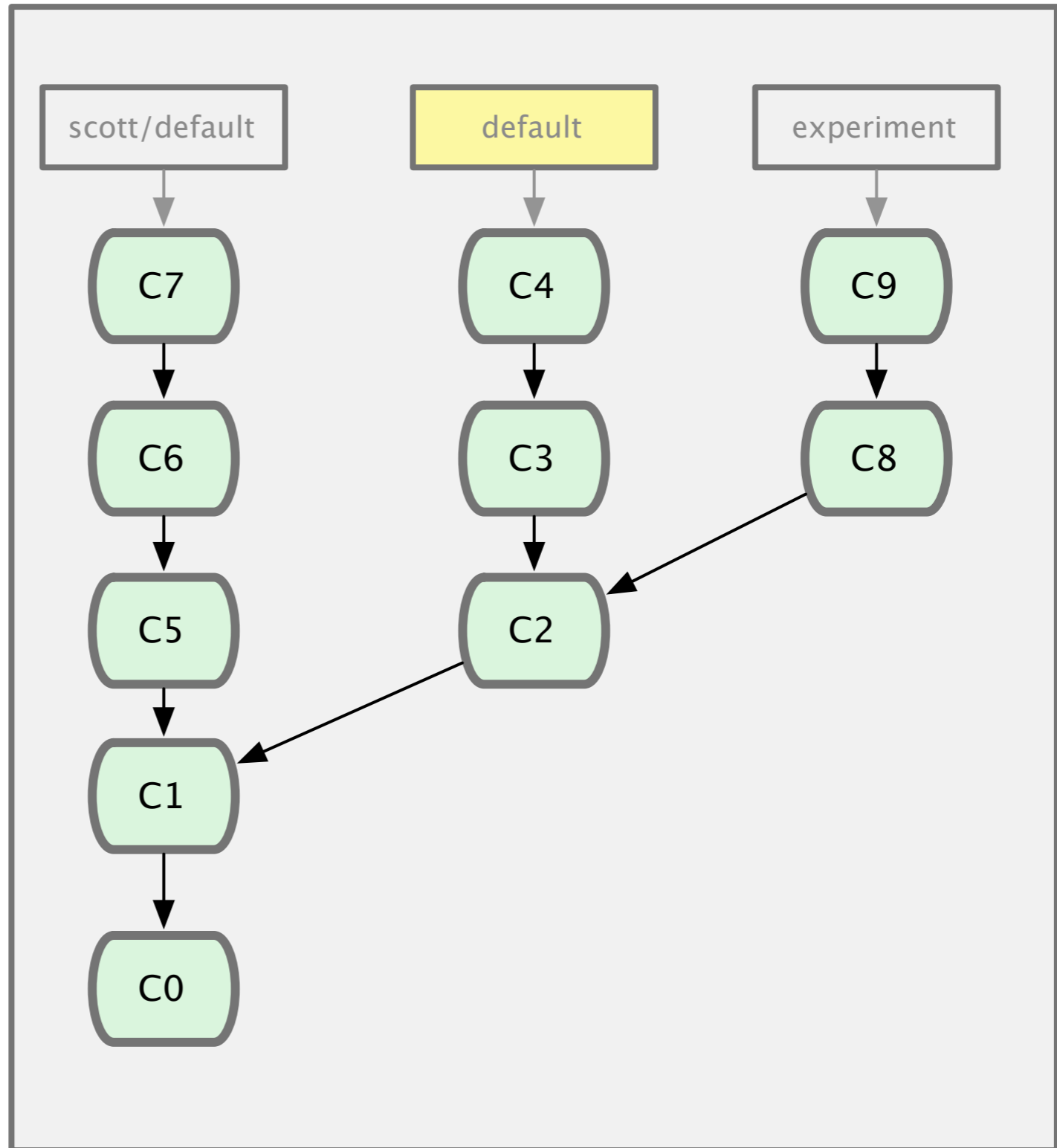
jessica



scott



jessica



scott

jessica

default

scott/default

default

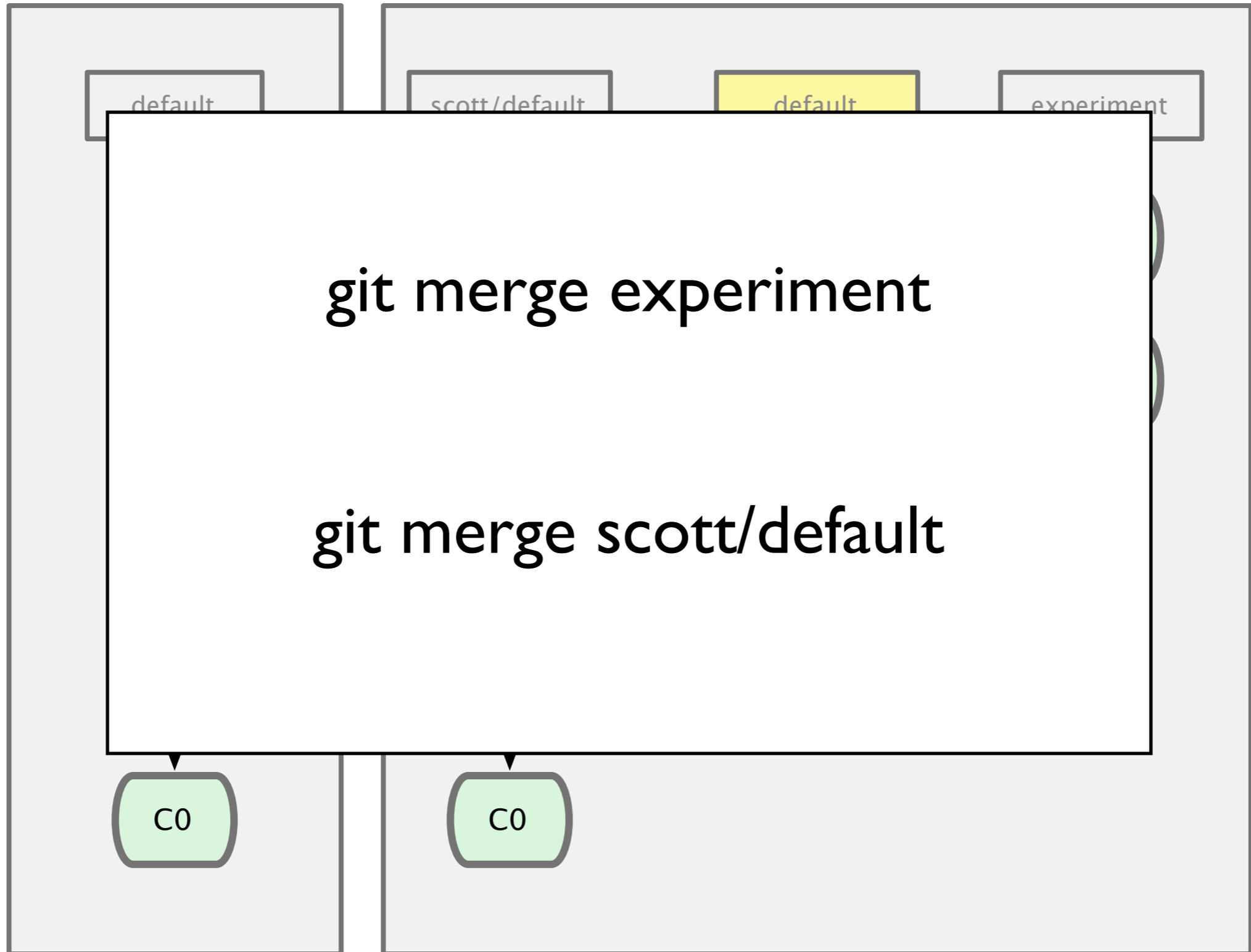
experiment

git merge experiment

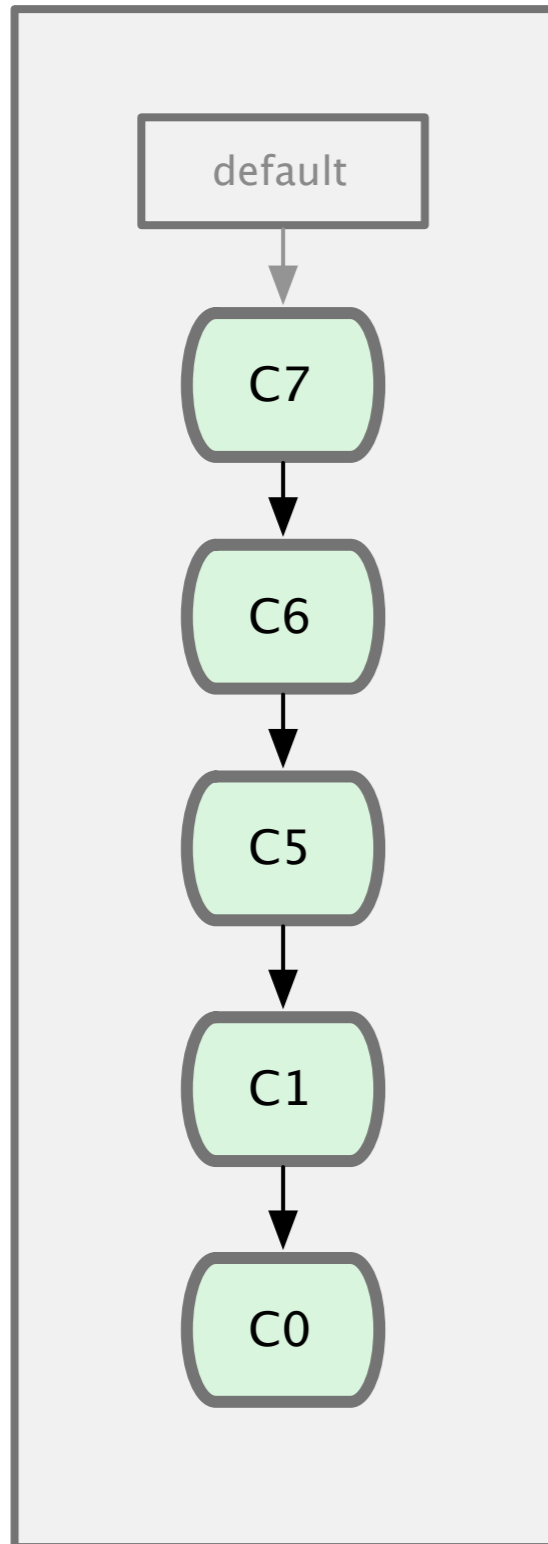
git merge scott/default

C0

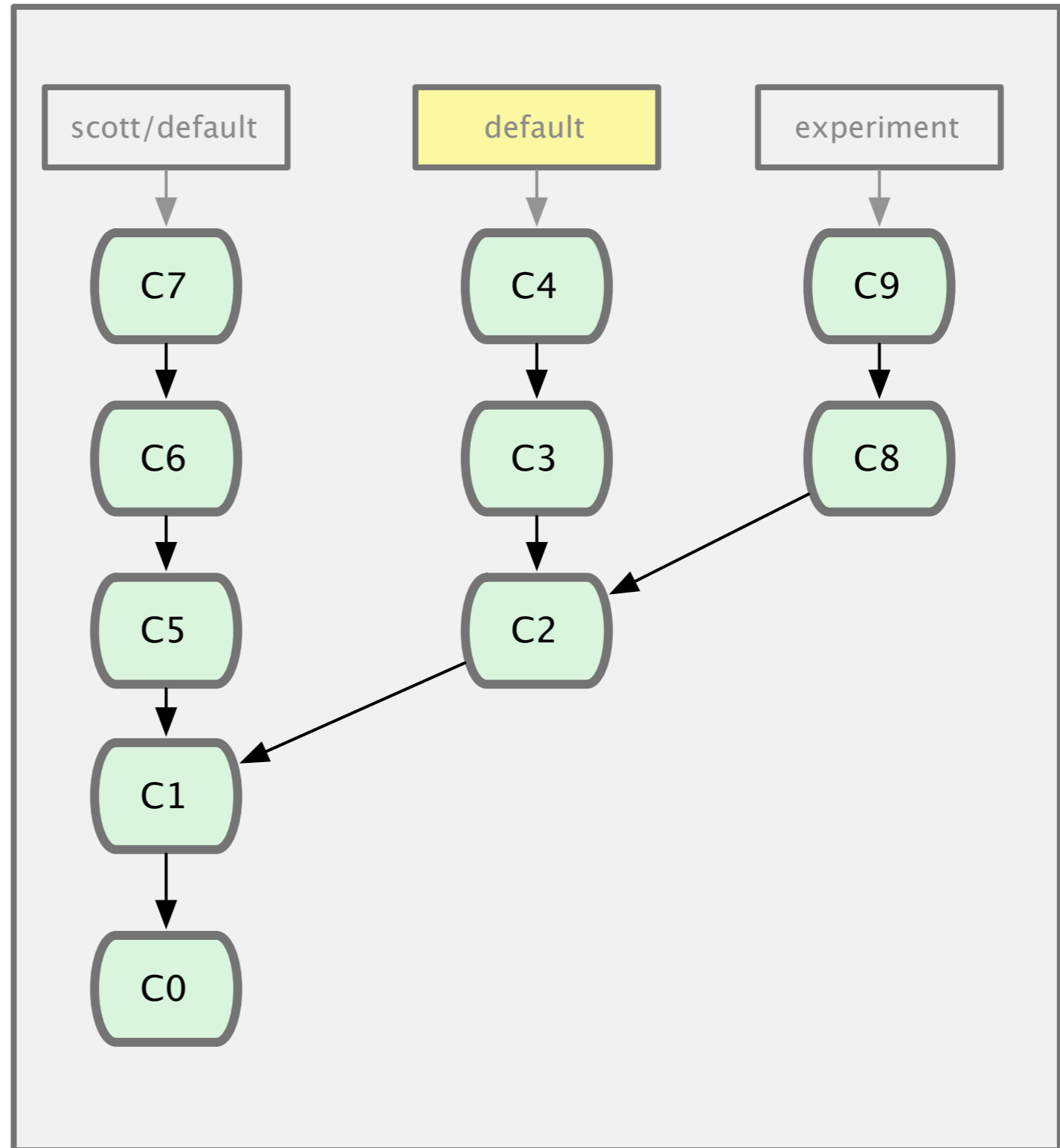
C0



scott

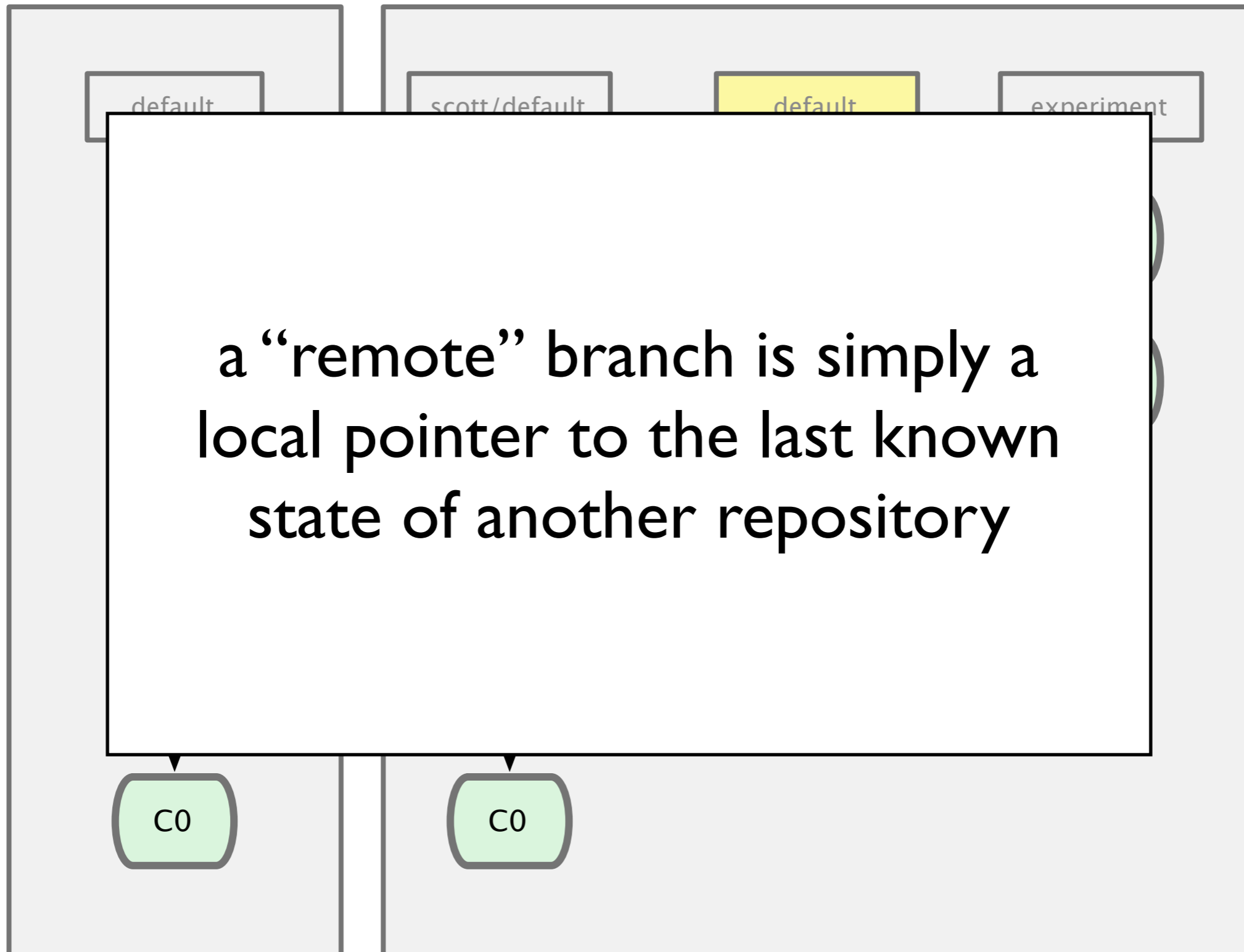


jessica



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'

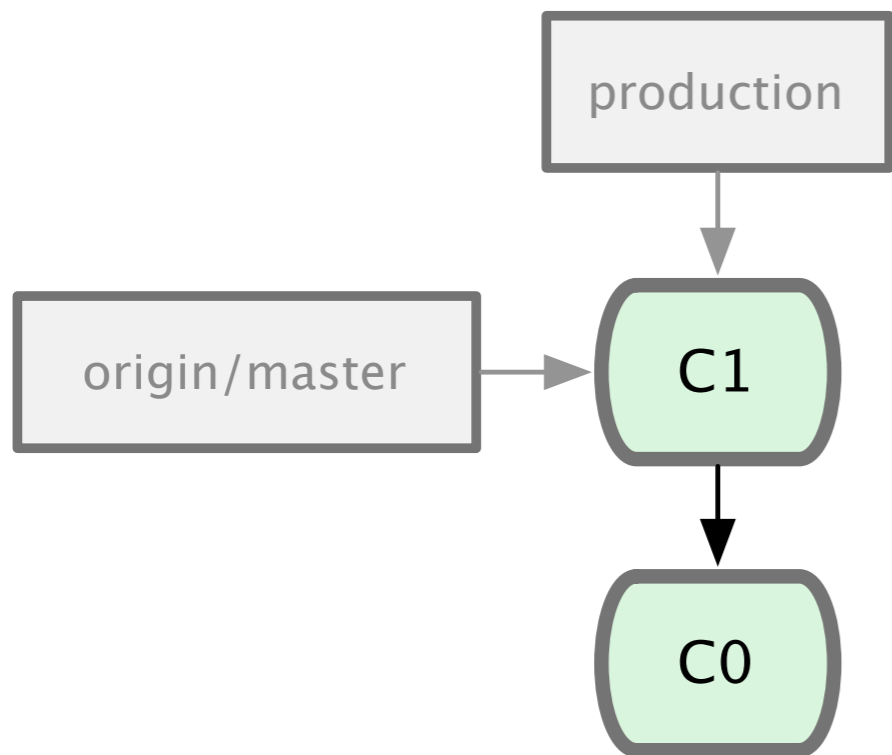
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout 'production'
- create a branch (iss102)

- 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

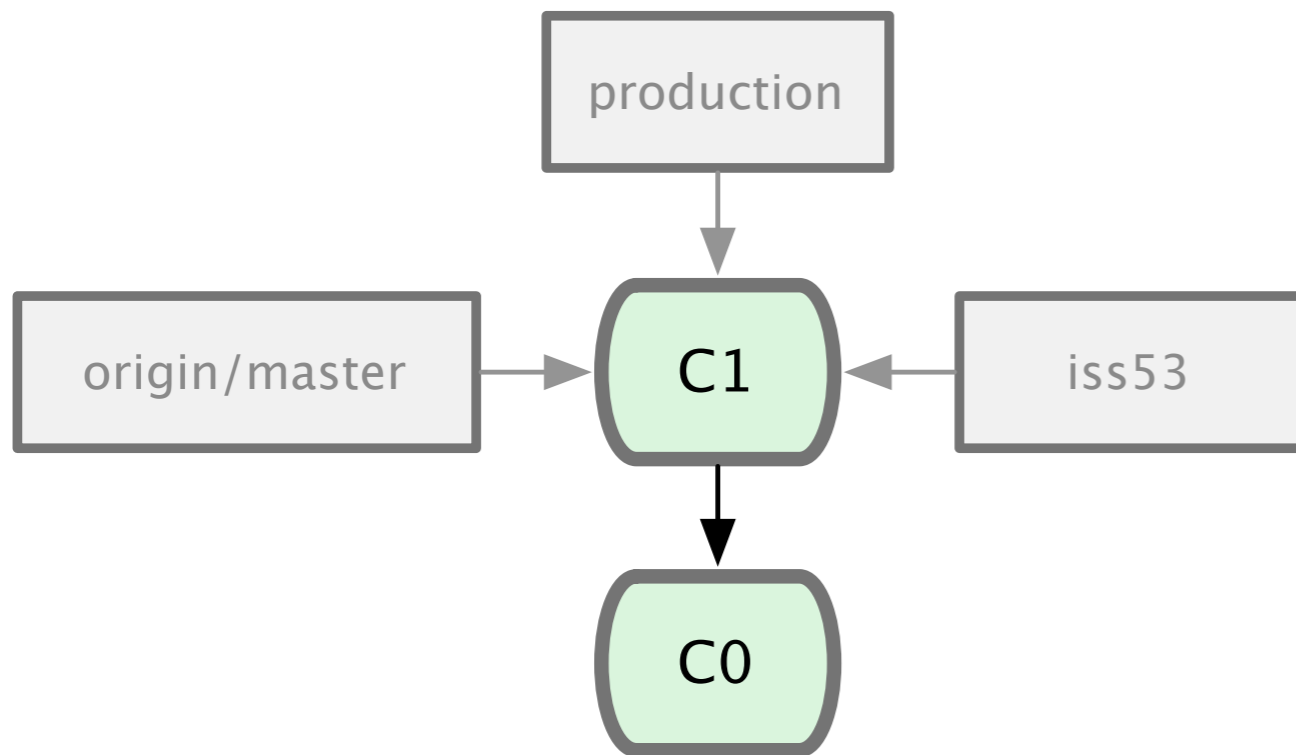
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout 'production'
- create a branch (iss102)
- fix the issue
- checkout 'production', merge 'iss102'

- 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'

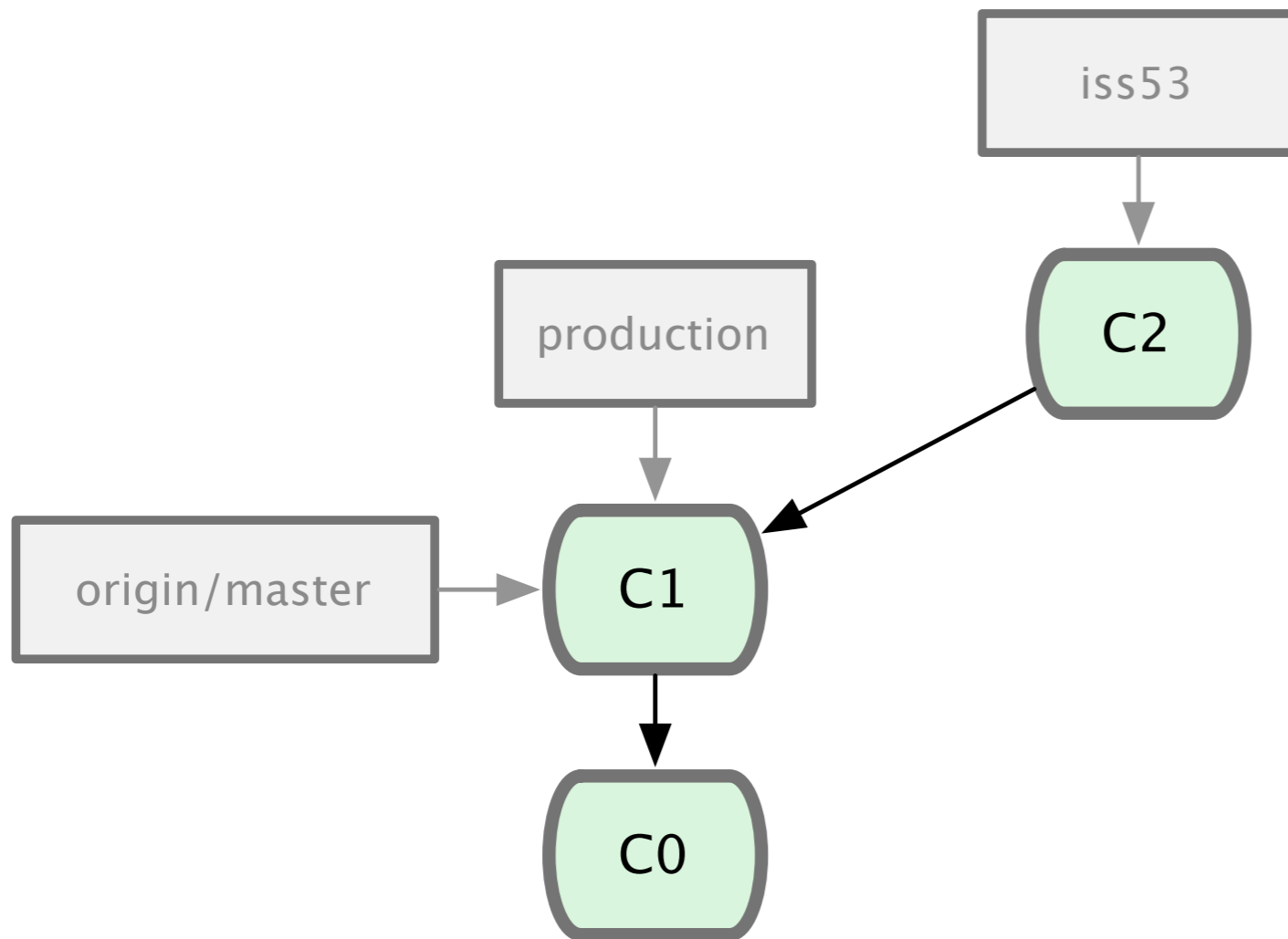
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout 'production'
- create a branch (iss102)
- fix the issue
- checkout 'production', merge 'iss102'
- push 'production'
- etc



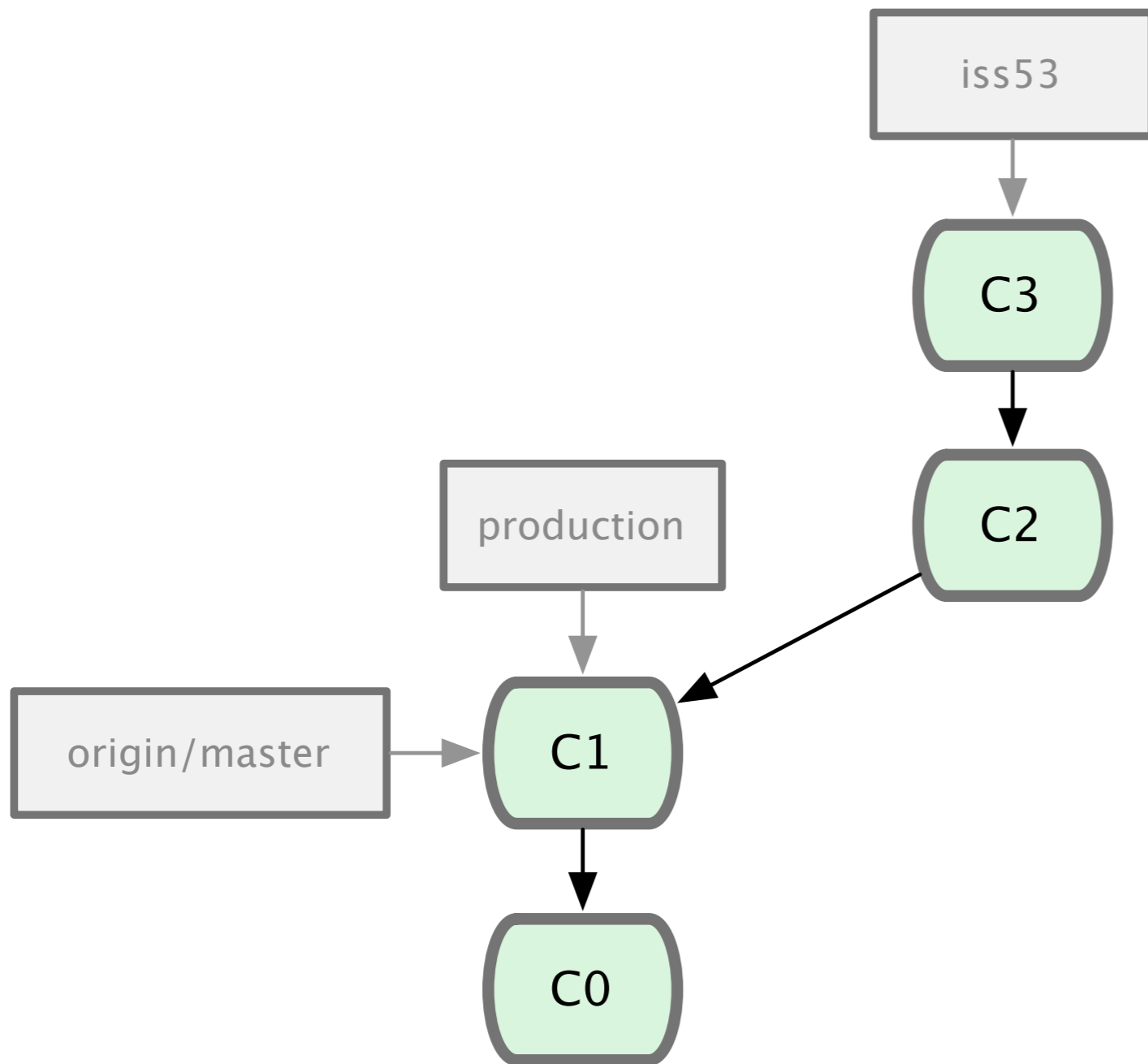
git clone



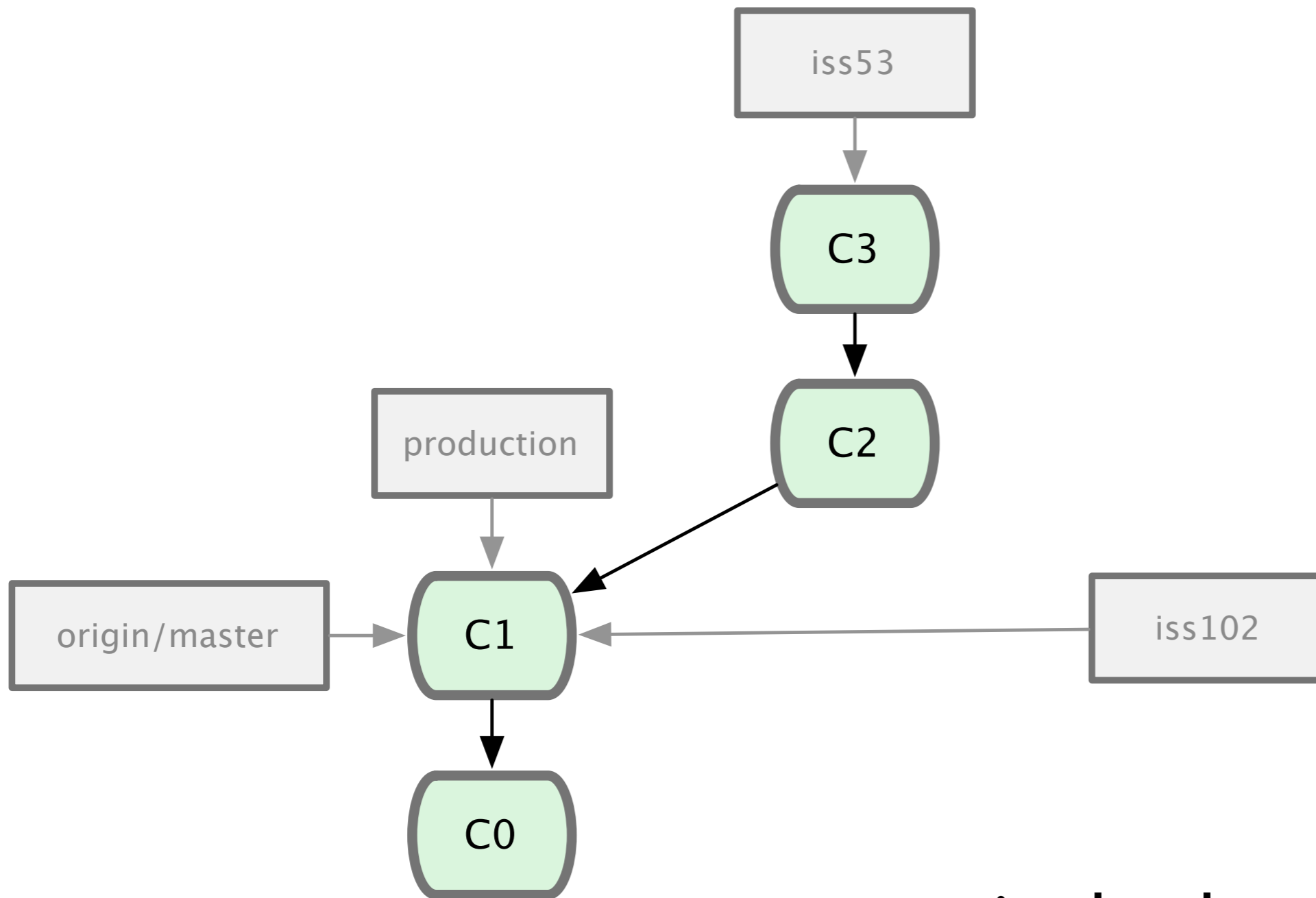
git checkout -b iss53



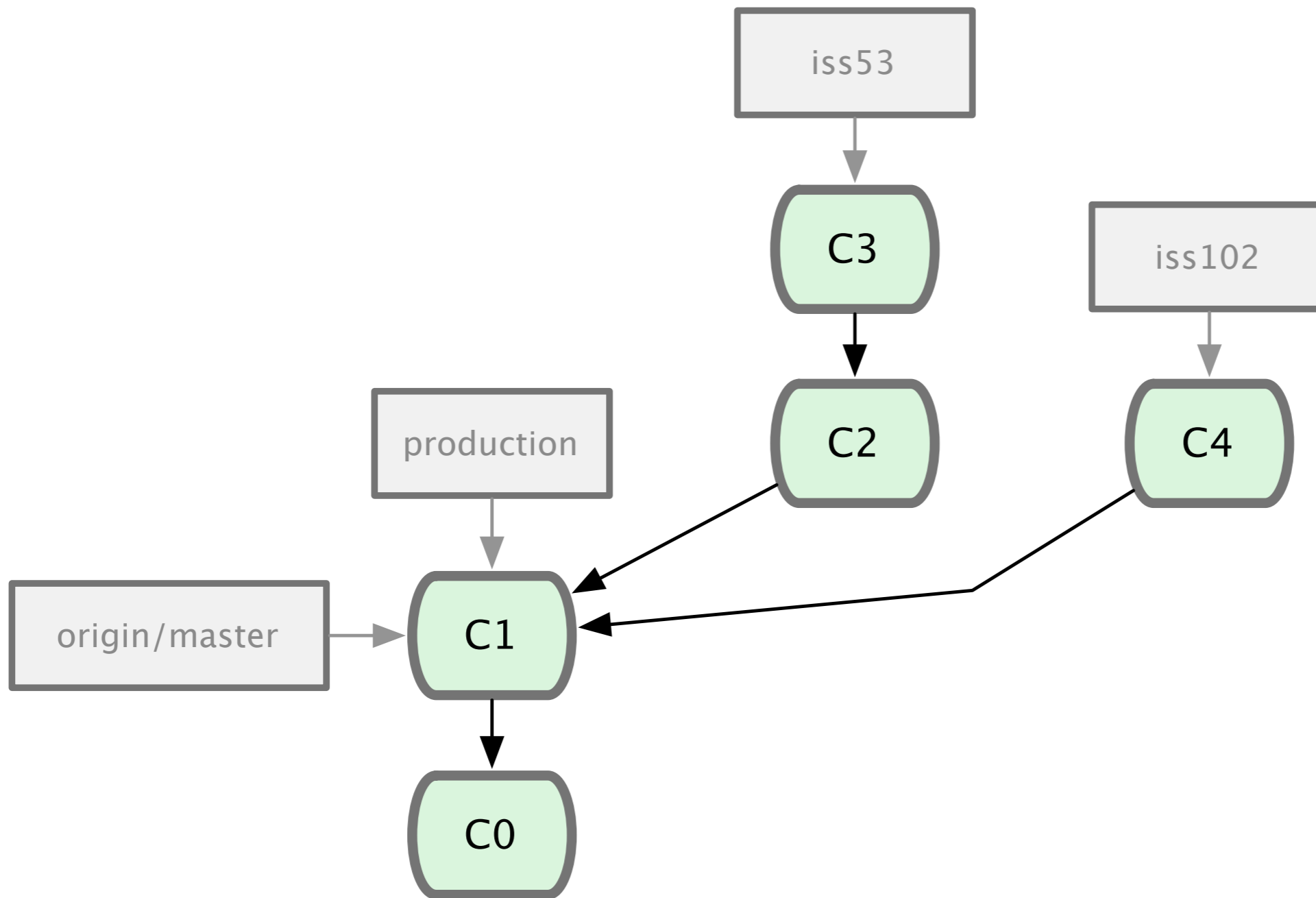
git commit



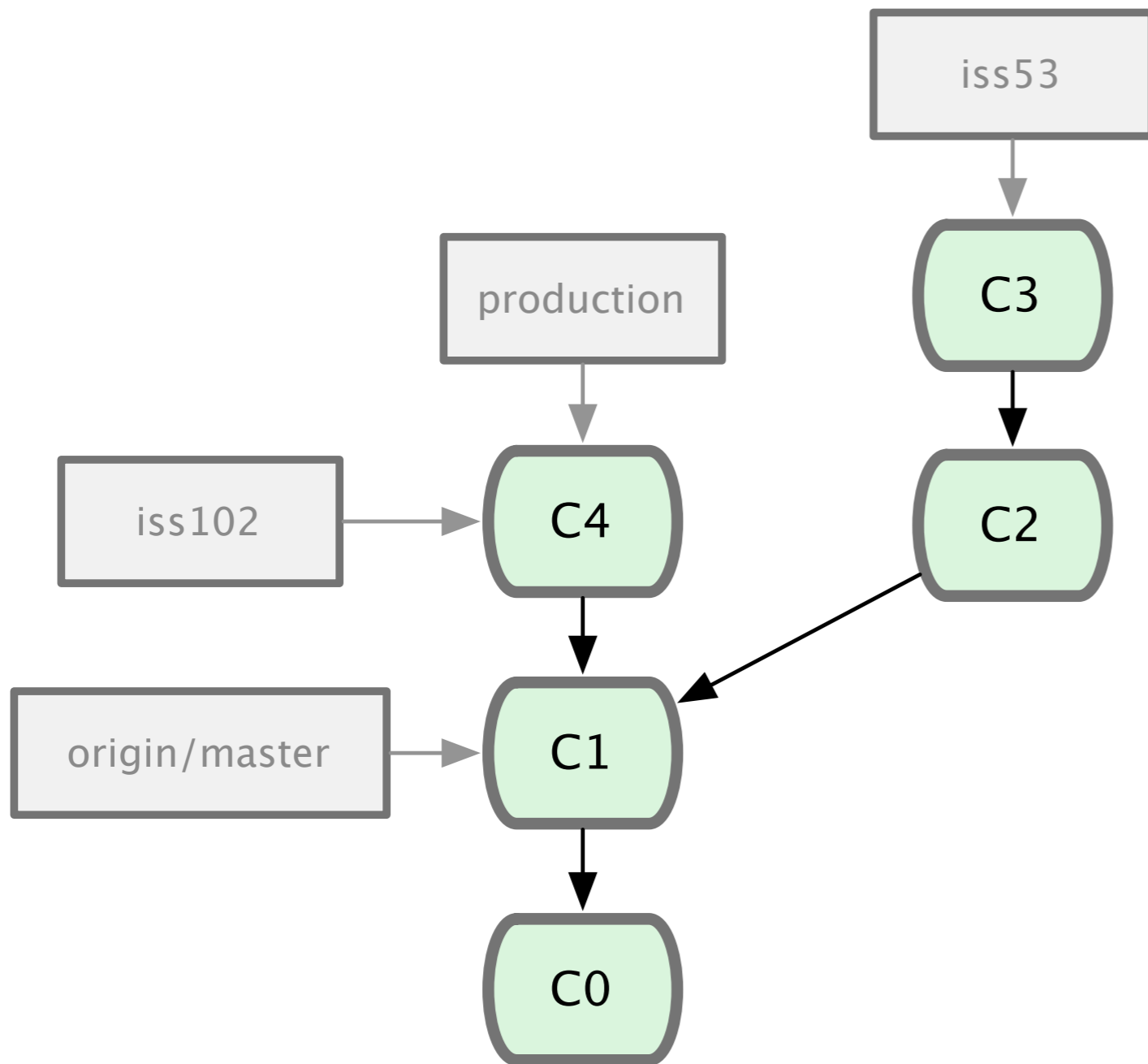
git commit



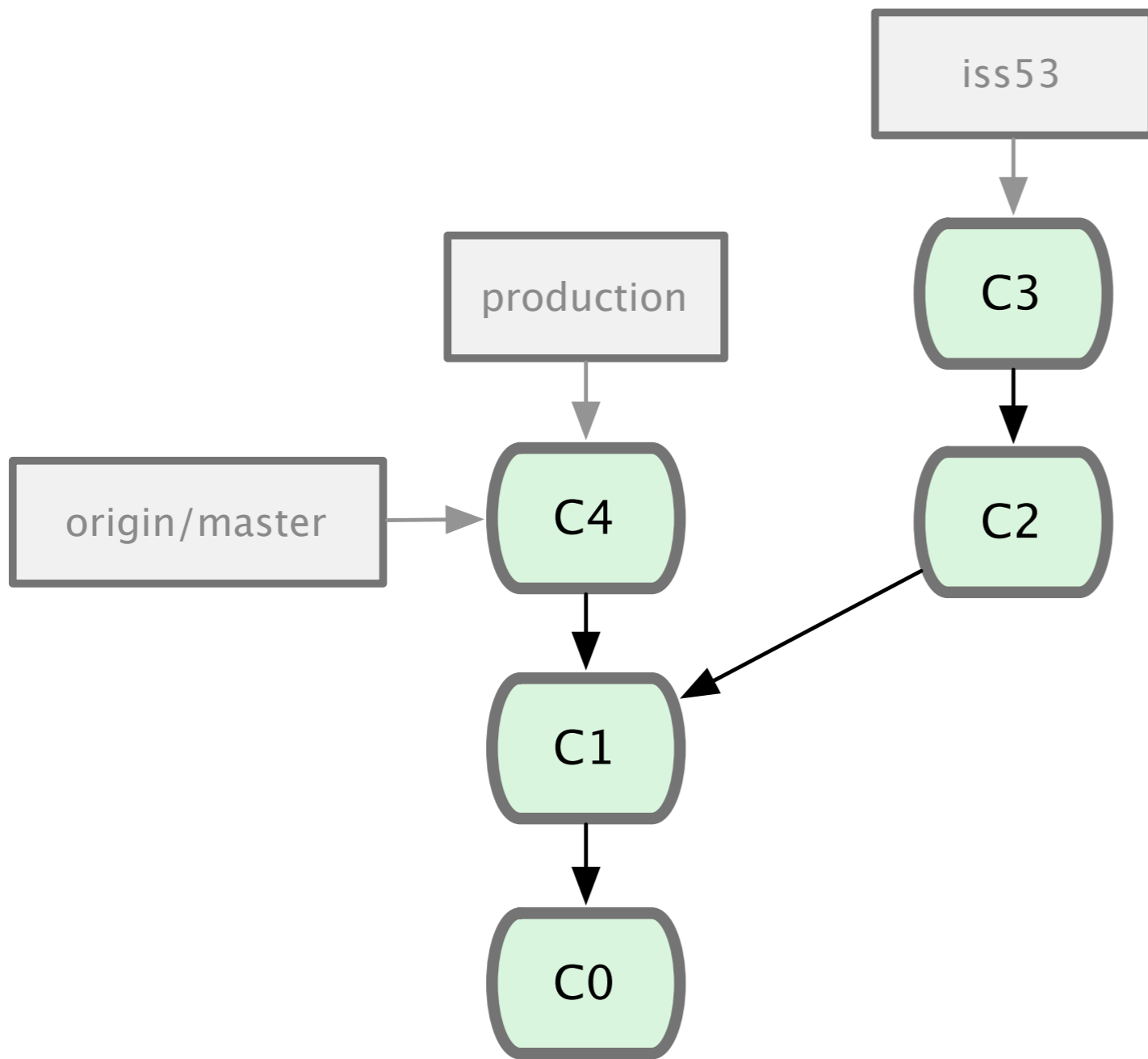
git checkout production
git checkout -b iss102



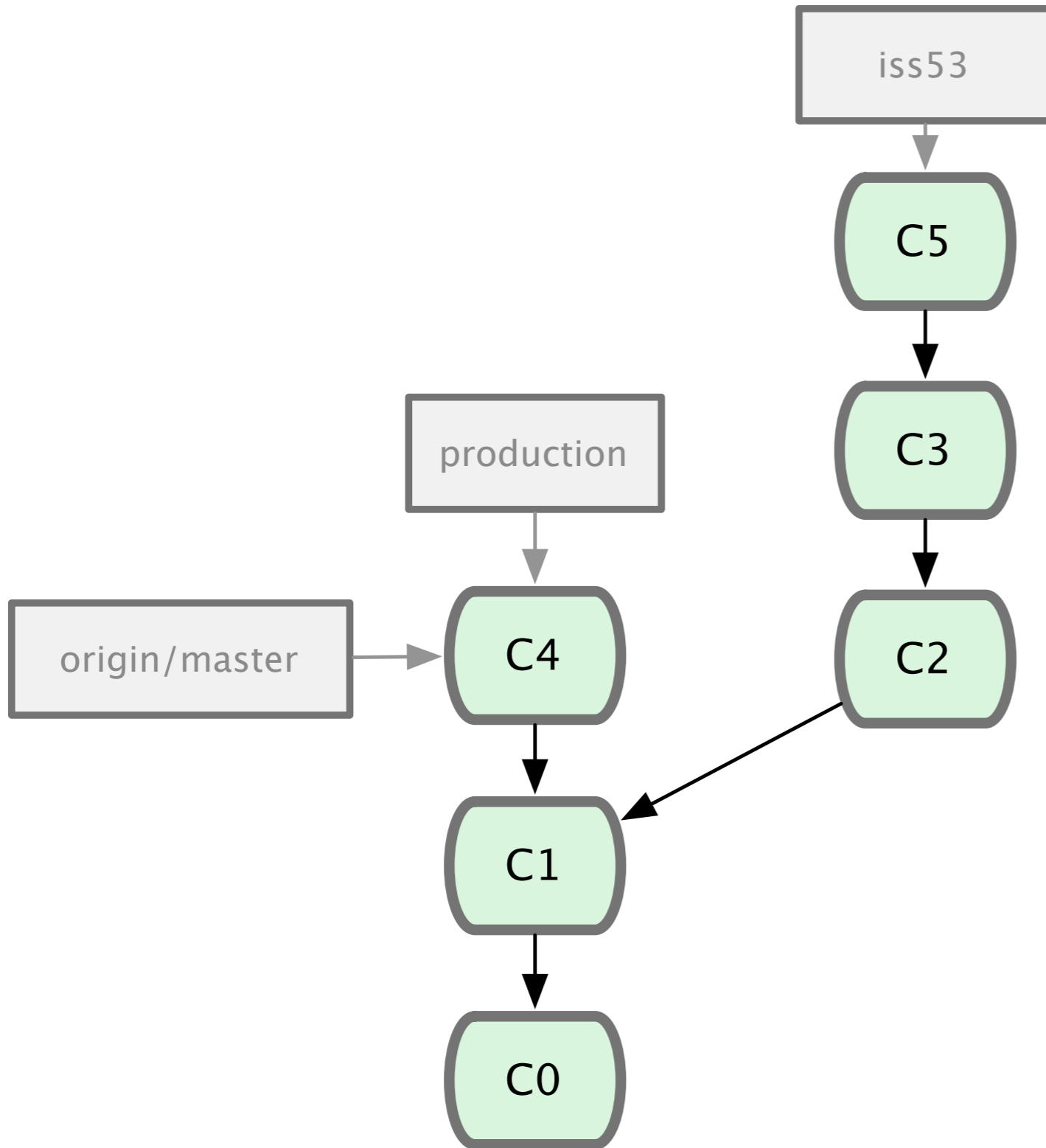
git commit



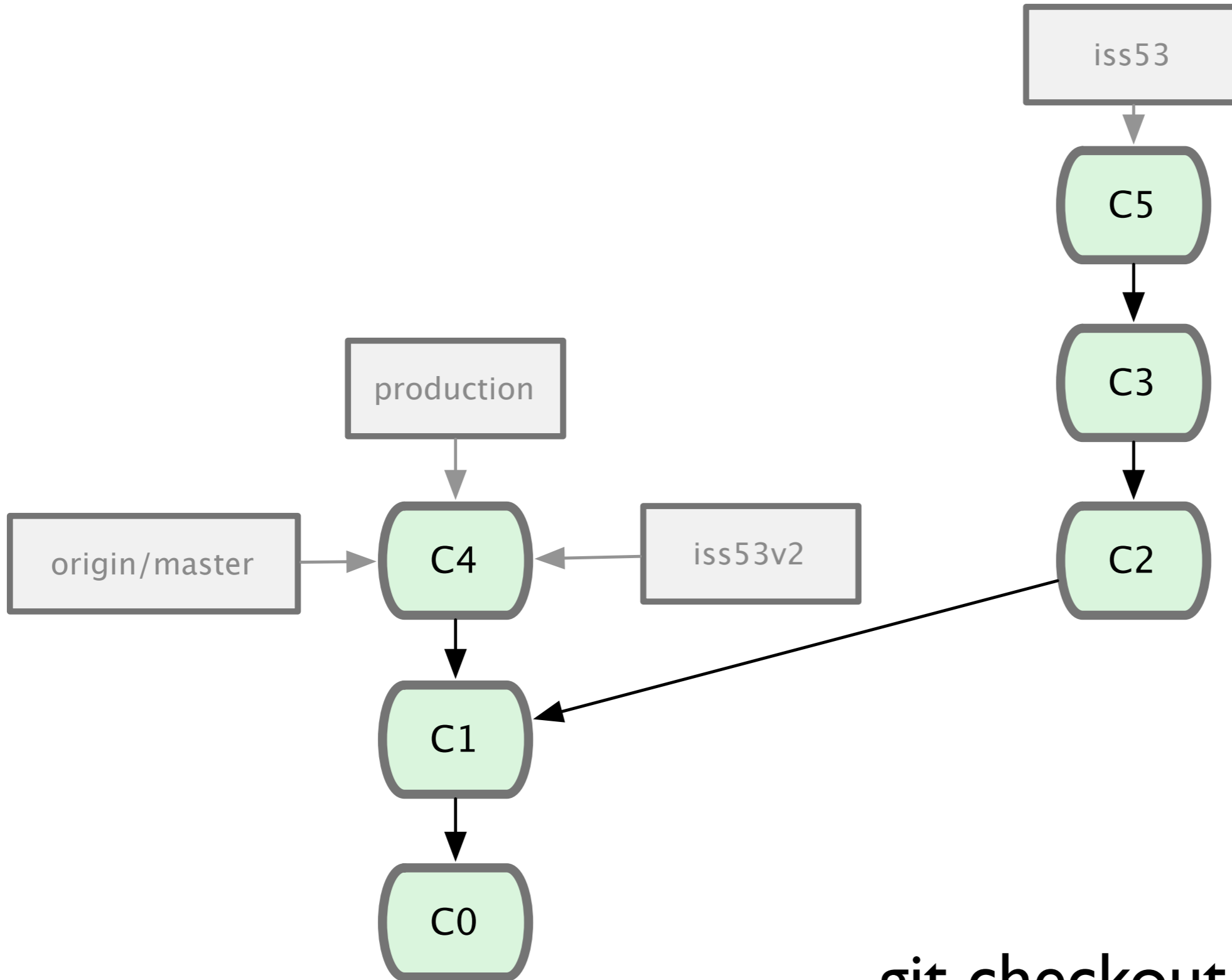
git checkout production
git merge iss102



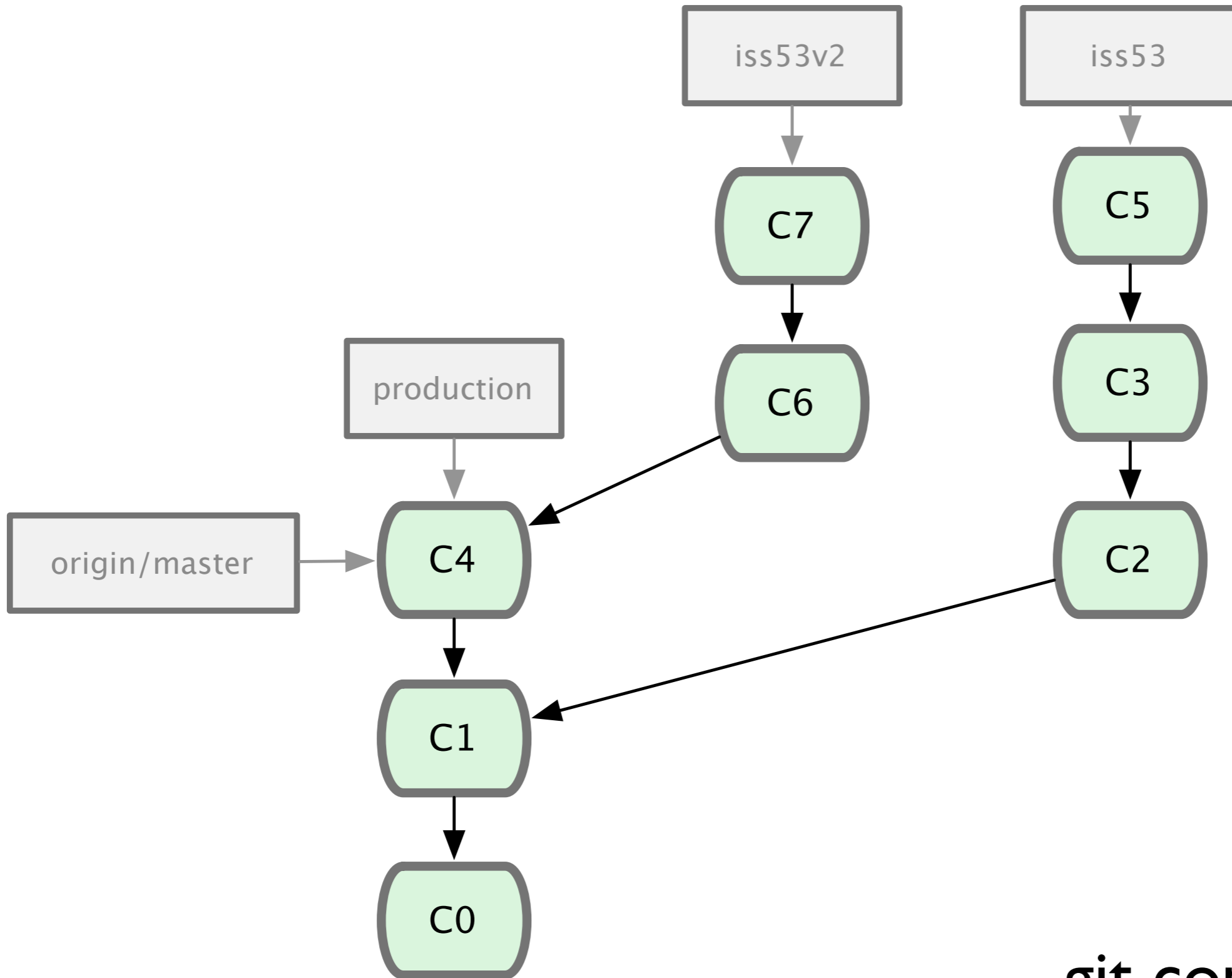
git push



git checkout iss53
git commit



`git checkout production`
`git checkout -b iss53v2`



git commit
git commit

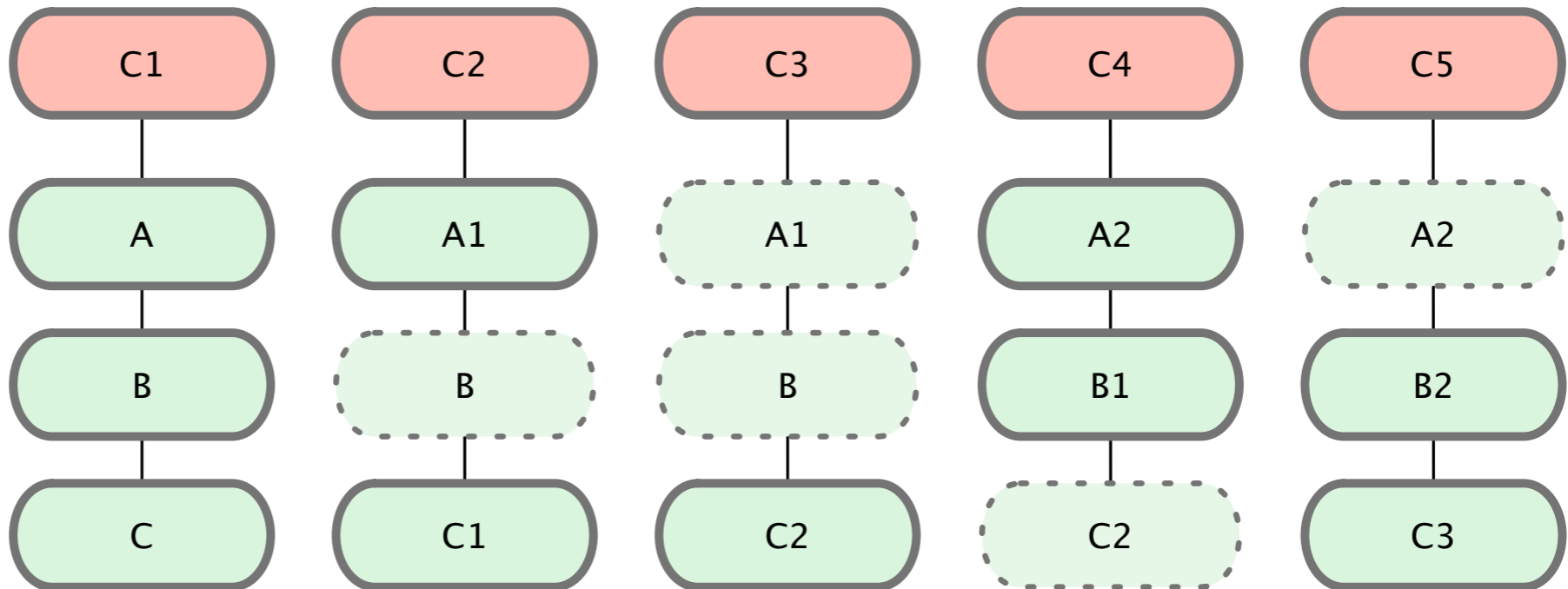
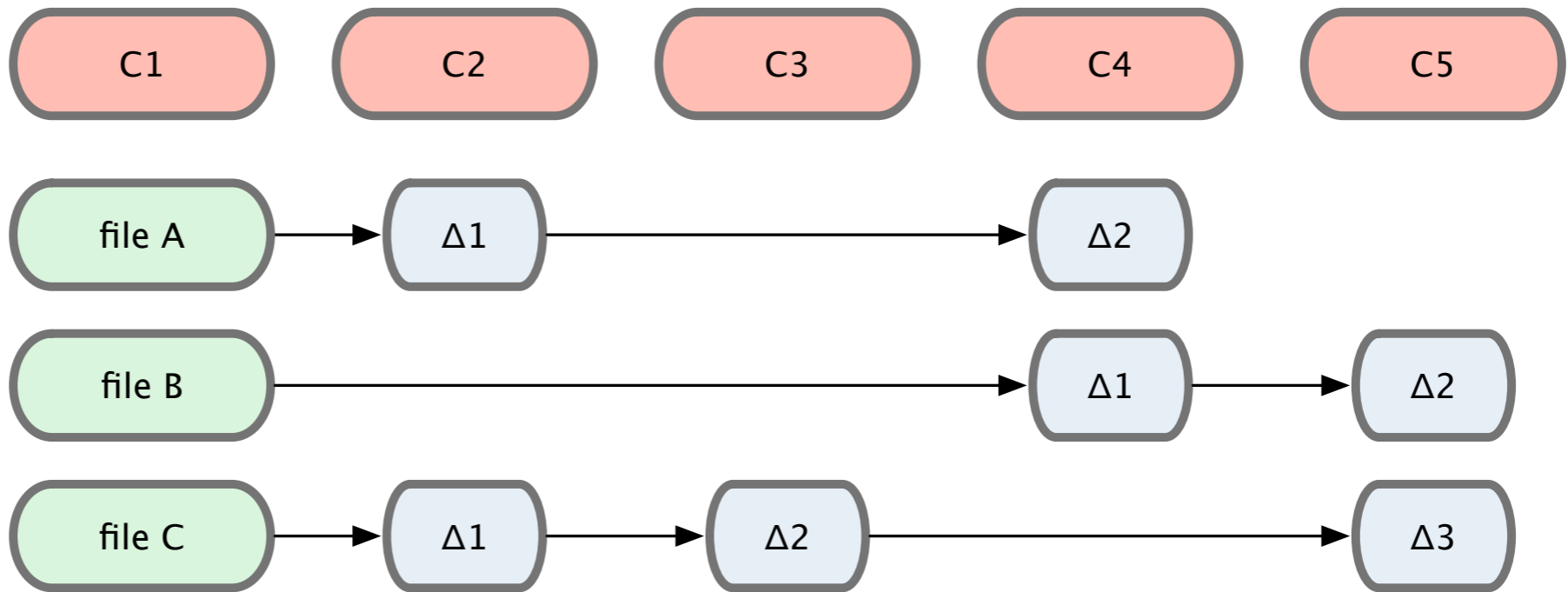
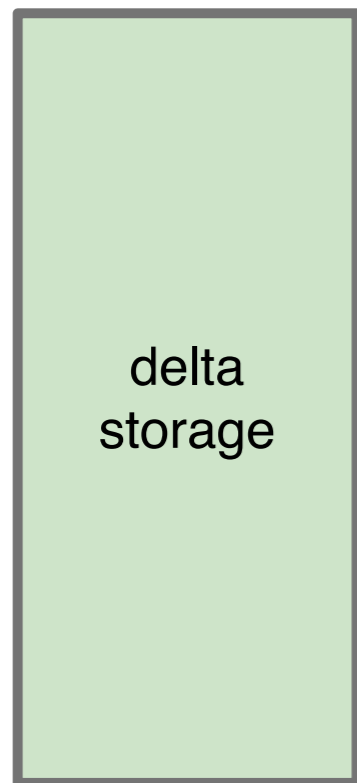
try out an idea

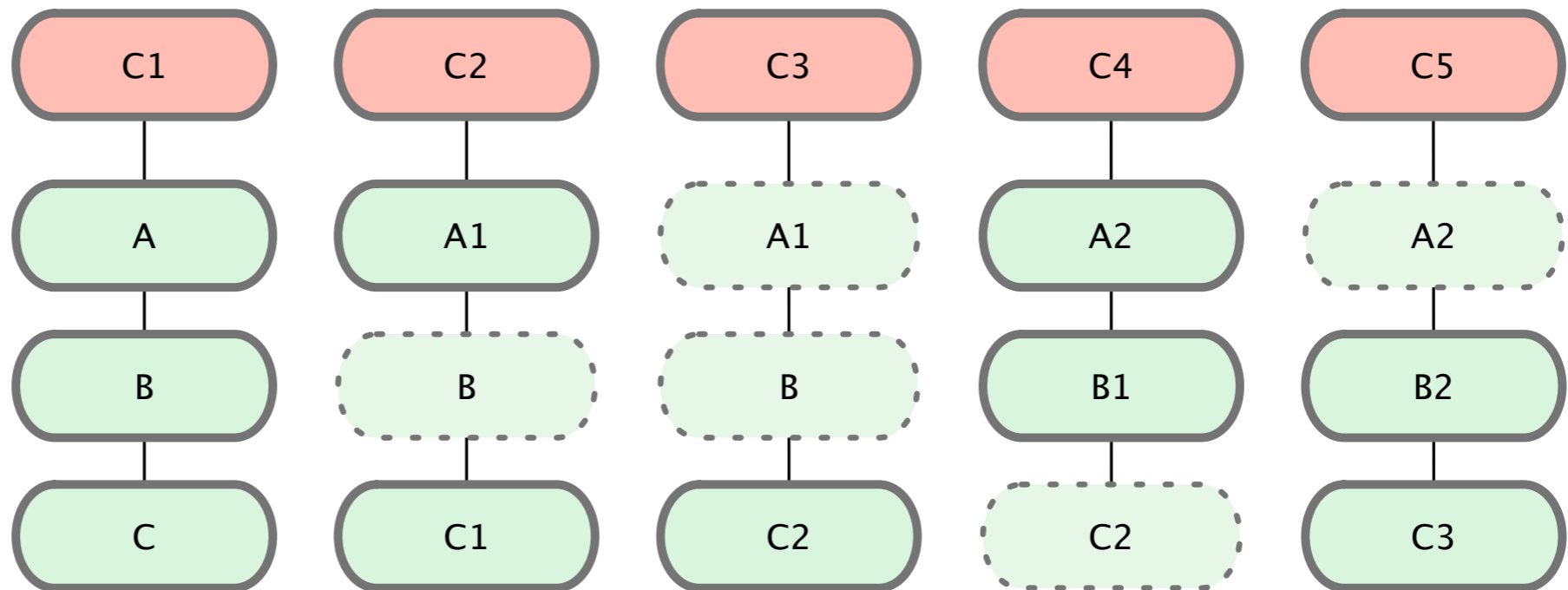
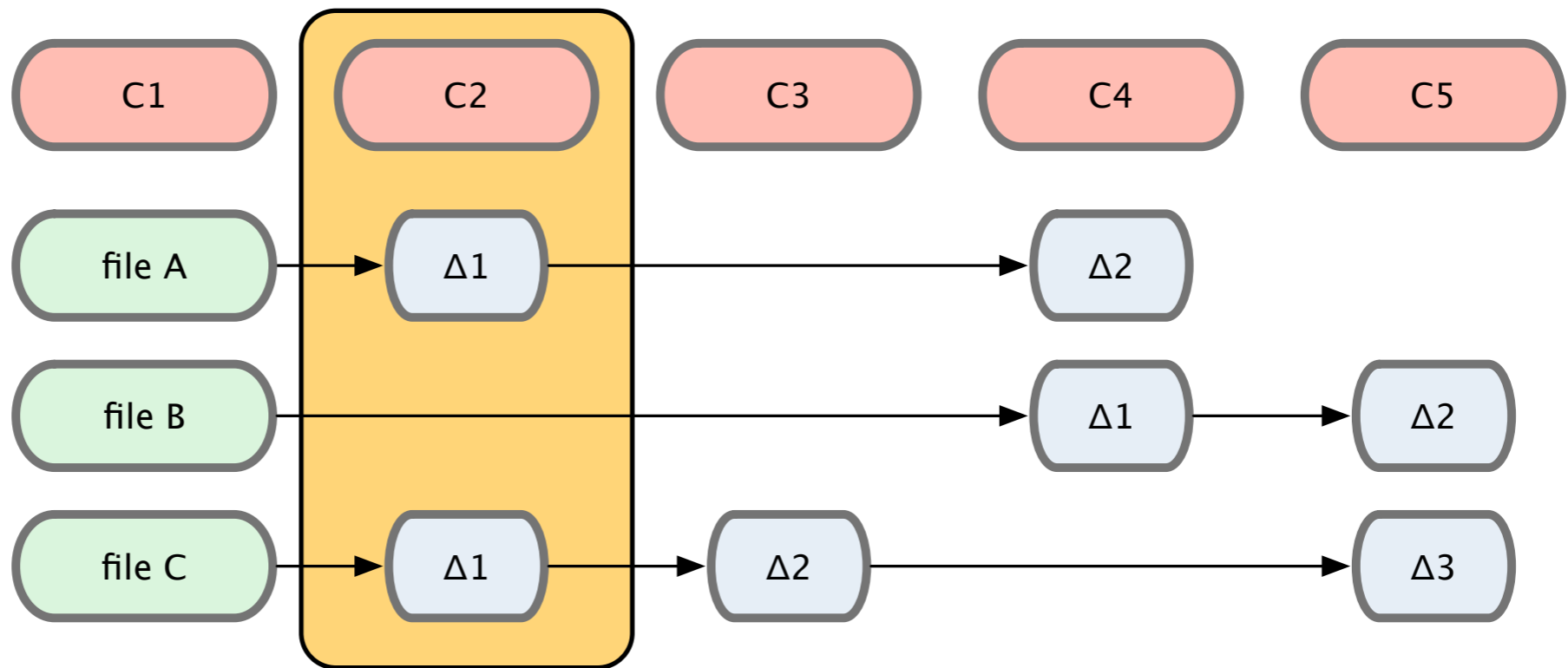
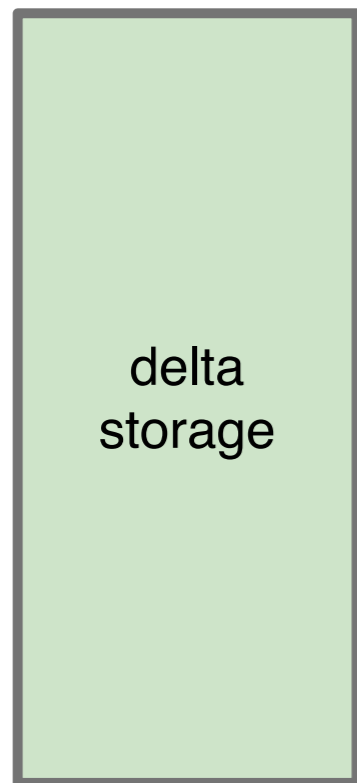
isolate work units

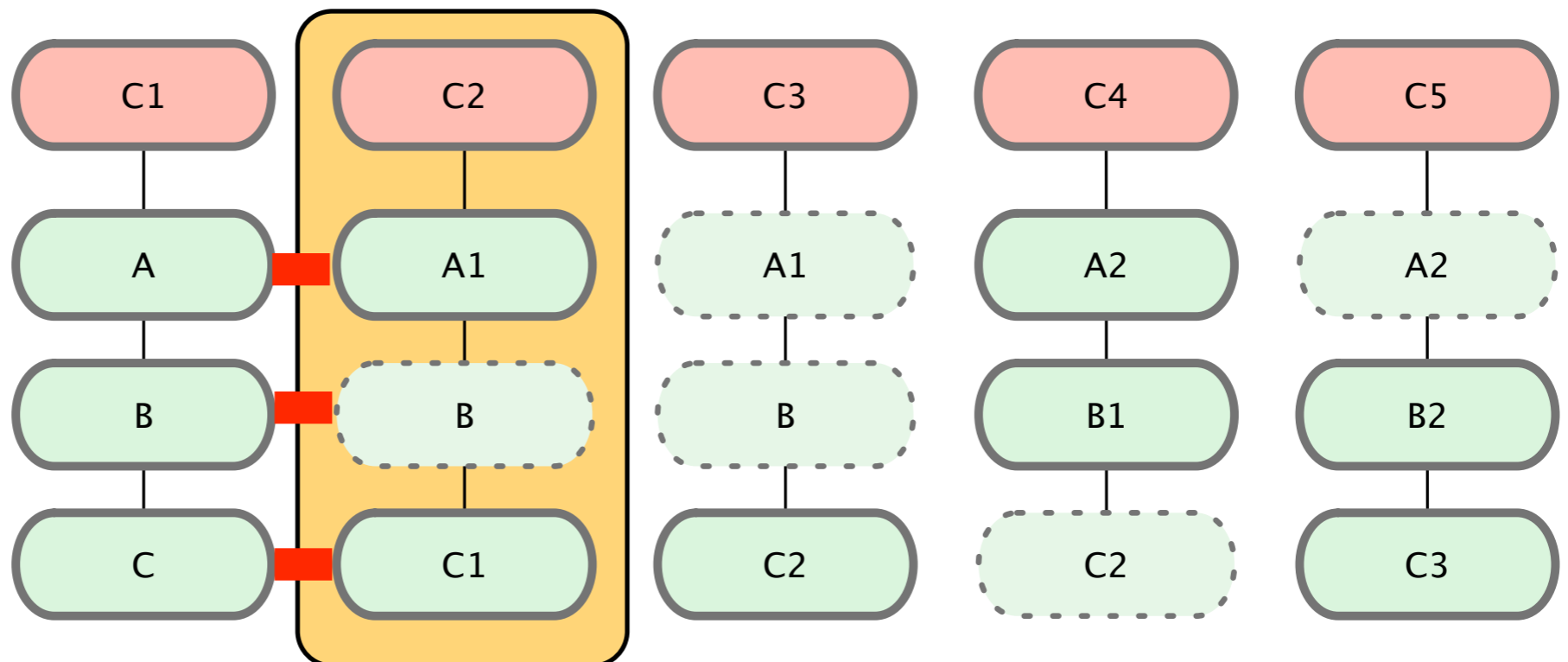
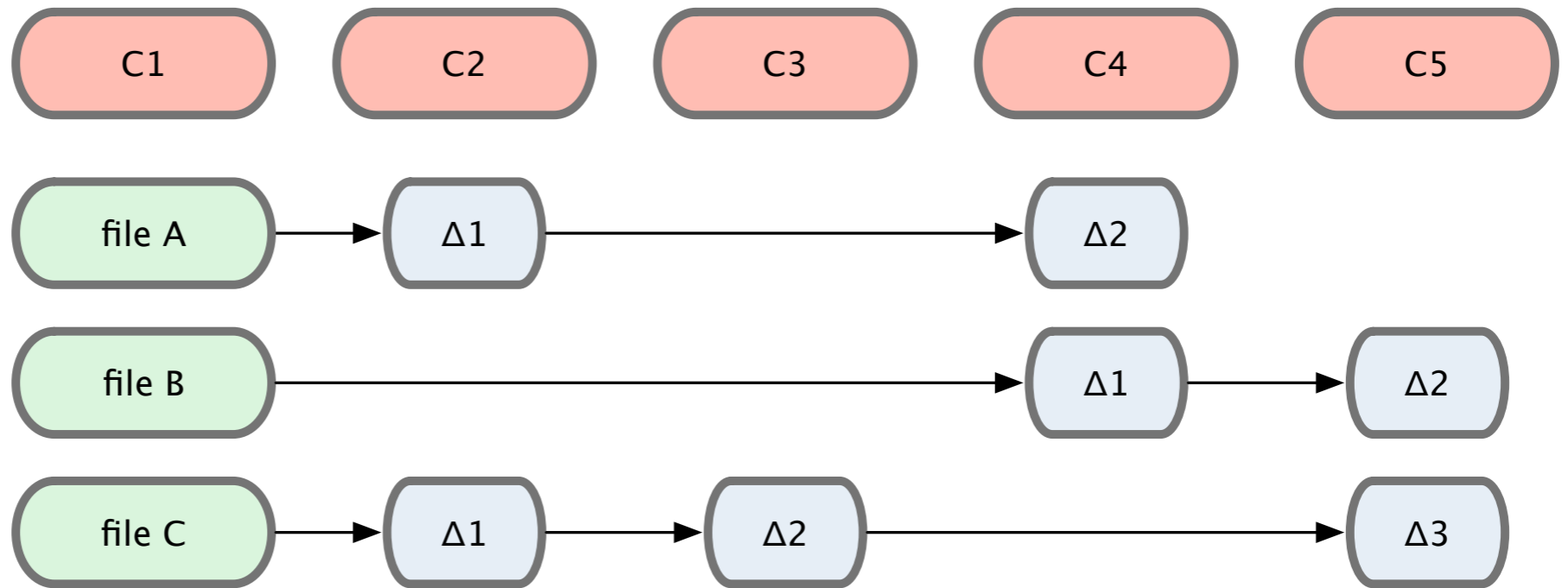
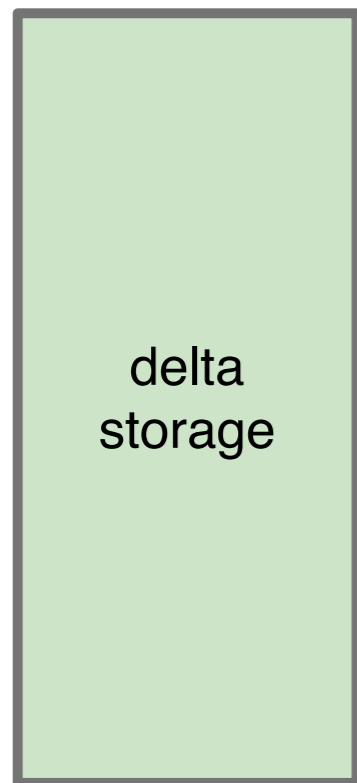
long running topics

**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):

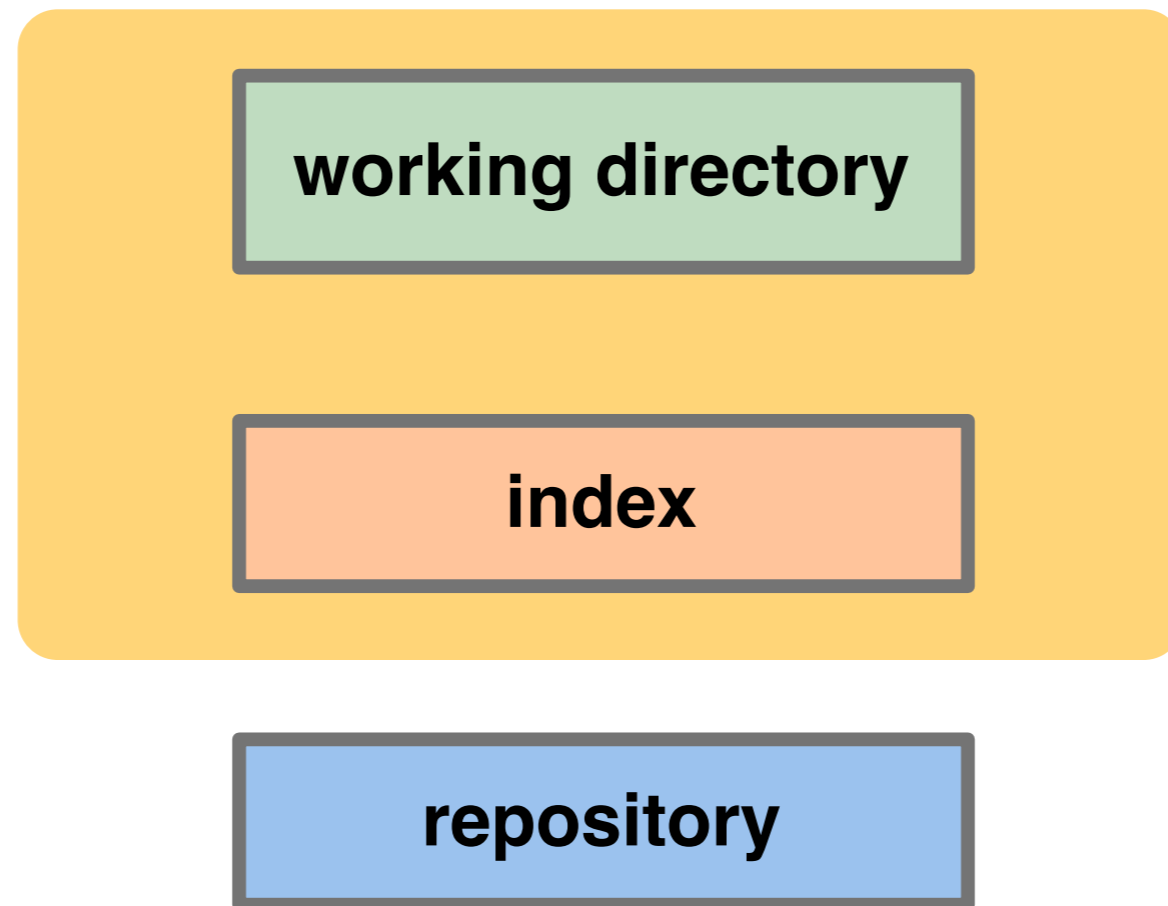
     def get(self):
-        self.response.out.write('Hello world!')
+        self.response.out.write('Hola mundo!')

 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 c110d7ff8cfb86fd5c9a8aee462678dbb4ef9b
```

```
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 c110d7ff8cfb86fd5c9a8aee462678dbb4ef9b
```

```
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 c110d7ff8cfb86fd5c9a8aee462678dbb4ef9b
```

```
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 c110d7ff8cfb86fd5c9e9a8aee462678dbb4ef9b
```

```
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

The image shows the top portion of the git-scm.com website. At the top left is the Git logo, which consists of three red plus signs followed by the word "git" in a green, lowercase, sans-serif font. To the right of the logo, the word "git" is written in a large, bold, black, lowercase font, followed by the tagline "the fast version control system" in a smaller, grey, lowercase font. Below this text is a decorative horizontal band featuring a stylized forest of green and brown trees. On the right side of this band is a cartoon illustration of a brown, bear-like character with a smiling face, holding a green leafy branch. Below the decorative band is a green navigation bar with white text links: "Home", "About Git", "Documentation", "Download", "Tools & Hosting", and "Wiki". The "Documentation" link is circled in black. Below the navigation bar are three main content sections: "Git is..." (orange background), "Projects using Git" (white background), and "Download Git" (red background). The "Git is..." section contains text describing Git as an open source, distributed version control system. The "Projects using Git" section lists "Git", "Linux Kernel", and "Perl" with bullet points. The "Download Git" section states that the latest stable release is "v1.6.1.1" and includes a link to "release notes (2009-01-25)".

+++ 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

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

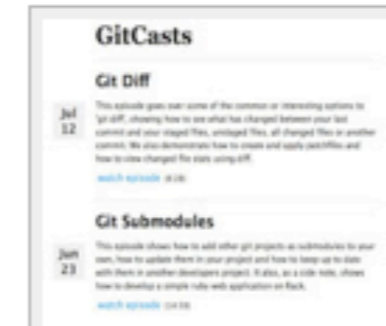



[Git Internals PDF](#)
by Scott Chacon



[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

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

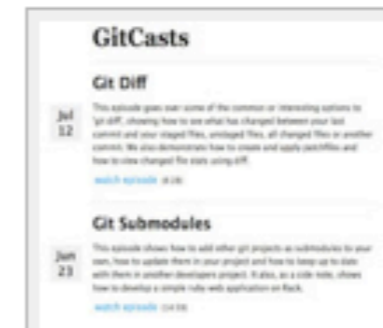



[Git Internals PDF](#)
by Scott Chacon



[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

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

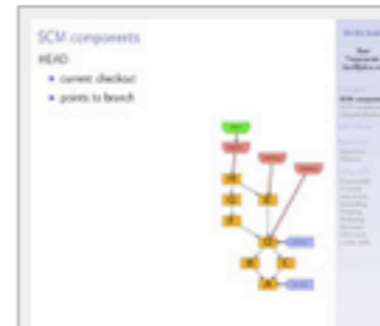



[Git Internals PDF](#)
by Scott Chacon



[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

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

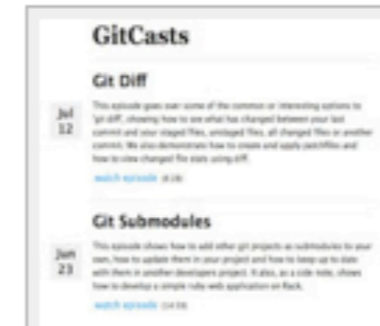



[Git Internals PDF](#)
by Scott Chacon



[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

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

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

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

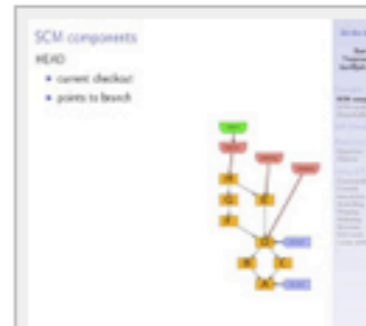


[Git Internals PDF](#)
by Scott Chacon



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

Videos



Basic Commands

Review

12

git init

12

git init

git clone

12

git init

git clone

git add

12

git init

git clone

git add

git commit

12

git init

git clone

git add

git commit

git branch

12

git init

git clone

git add

git commit

git branch

git checkout

12

git init

git merge

git clone

git add

git commit

git branch

git checkout

12

git init

git merge

git clone

git remote

git add

git commit

git branch

git checkout

12

git init

git merge

git clone

git remote

git add

git fetch

git commit

git branch

git checkout

12

git init

git merge

git clone

git remote

git add

git fetch

git commit

git push

git branch

git checkout

12

git init

git merge

git clone

git remote

git add

git fetch

git commit

git push

git branch

git diff

git checkout

12

git init

git merge

git clone

git remote

git add

git fetch

git commit

git push

git branch

git diff

git checkout

git log

12

that's it!

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