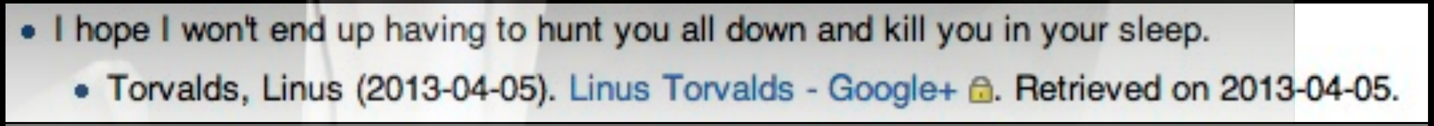
Development with Git

“Digging in your eye-sockets with a fondue fork is strictly considered to be bad for your health, and seven out of nine optometrists are dead set against the practice.”

- Linus Torvalds, on Git mailing list

http://en.wikiquote.org/wiki/Linus_Torvalds

Every time you don't use version control, Linus Torvalds get's a little angrier!



- I hope I won't end up having to hunt you all down and kill you in your sleep.
  - Torvalds, Linus (2013-04-05). Linus Torvalds - Google+ 🔒. Retrieved on 2013-04-05.

Outline

- Version control software
- Git for software development
- Python software packaging

Version Control

Version Control

- 0’th order

Version Control

- 0’th order
  - Development history for your source code

Version Control

- 0’th order
  - Development history for your source code
  - Collaboration with other developers

Version Control

- 0’th order
  - Development history for your source code
  - Collaboration with other developers
  - Allows experimentation without breaking existing code

Version Control

- 0’th order
  - Development history for your source code
  - Collaboration with other developers
  - Allows experimentation without breaking existing code
  - Can be complicated; but it’s not only useful, it is necessary

Version control

- The generics
  - Files and development history stored in repositories
  - Check out files to a working directory
  - Commit changes back to repository
  - Update your working directory with commits from other developers
  - Centralized vs. decentralized

Version control

- Centralized
  - Everyone commits to a server
  - Does not encourage offline development
  - Single point of failure
  - 90's: CVS
  - 00's: Subversion
  - Now

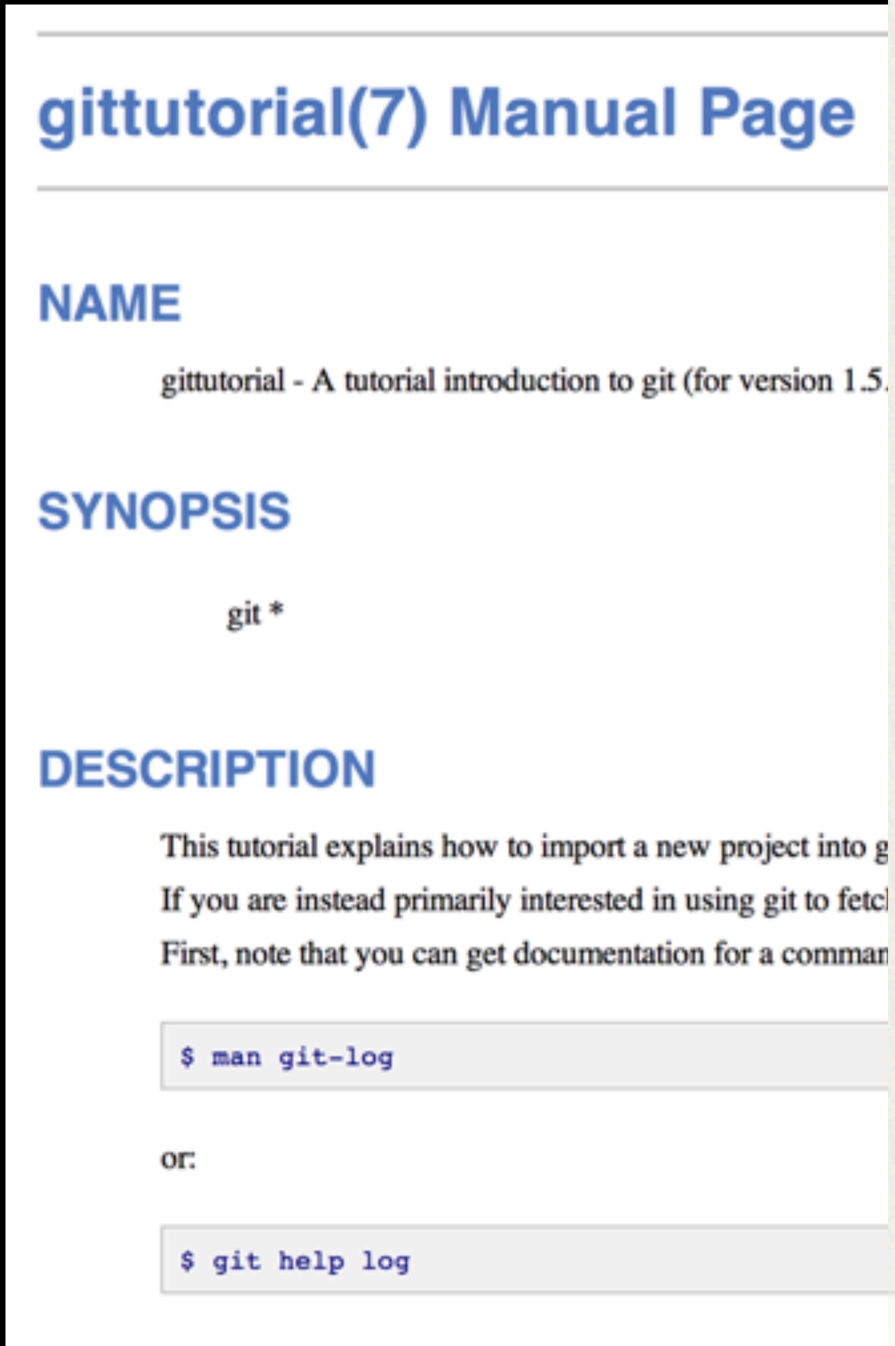
Version control

- Decentralized
  - Everyone has a copy
  - Local commits
  - Push to and pull from a shared copy
  - Encourages experimentation
  - Many Contenders
    - Mercurial, Bazaar, Git, ...

http://git-scm.com

Git

Git basics



gittutorial(7) Manual Page

NAME

gittutorial - A tutorial introduction to git (for version 1.5.

SYNOPSIS

git *

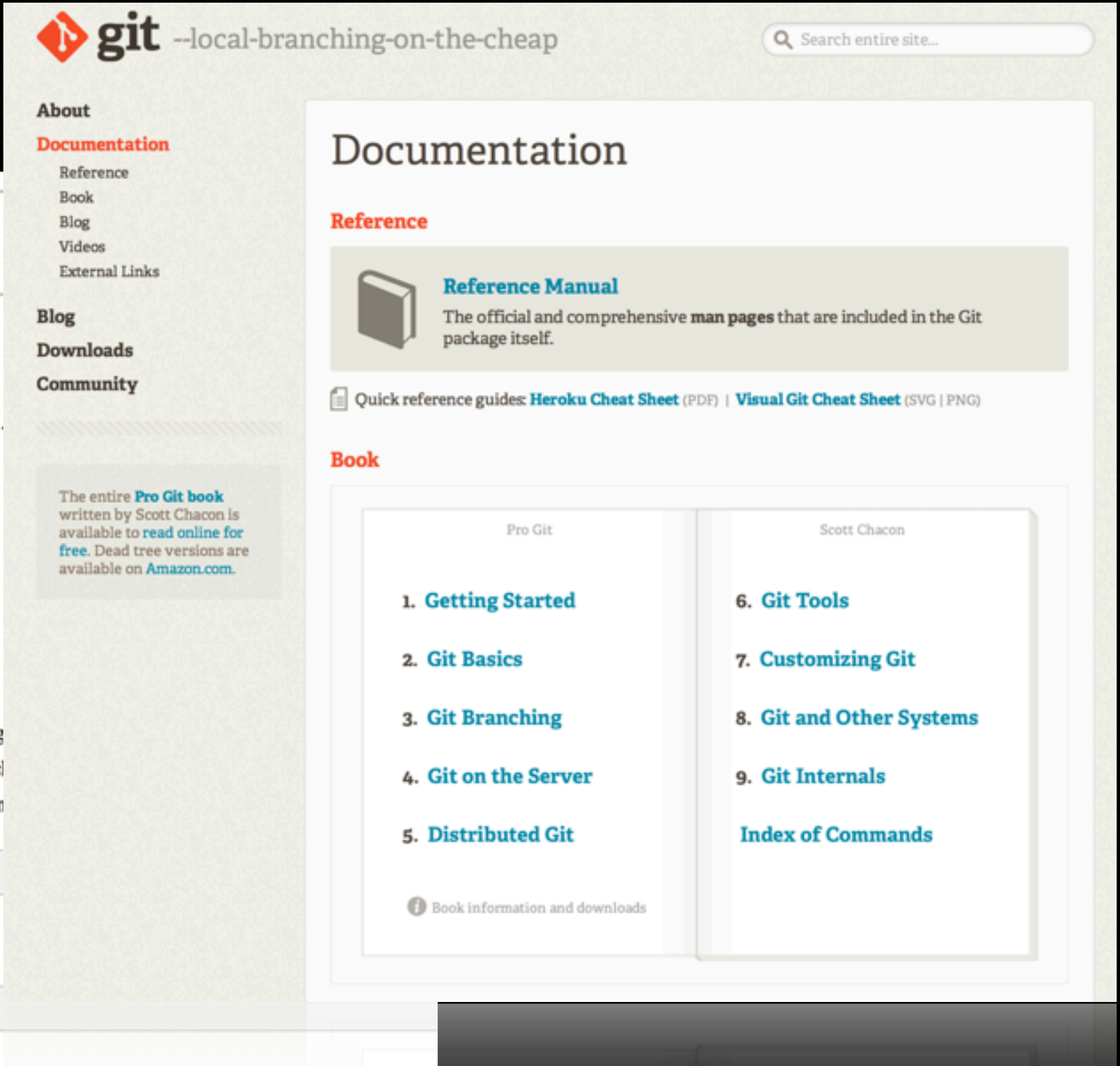
DESCRIPTION

This tutorial explains how to import a new project into g
If you are instead primarily interested in using git to fetcl
First, note that you can get documentation for a comman

```
$ man git-log
```

or:

```
$ git help log
```



git --local-branching-on-the-cheap

Search entire site...

About
Documentation
Reference
Book
Blog
Videos
External Links
Blog
Downloads
Community

The entire Pro Git book written by Scott Chacon is available to read online for free. Dead tree versions are available on Amazon.com.

Documentation

Reference

Reference Manual
The official and comprehensive **man pages** that are included in the Git package itself.

Quick reference guides: Heroku Cheat Sheet (PDF) | Visual Git Cheat Sheet (SVG | PNG)

Book

Pro Git

Scott Chacon

1. Getting Started
2. Git Basics
3. Git Branching
4. Git on the Server
5. Distributed Git
6. Git Tools
7. Customizing Git
8. Git and Other Systems
9. Git Internals

Index of Commands

Book information and downloads

Git

- Getting started

```
[]$ mkdir myawesomesoftware
[]$ cd myawesomesoftware/
[]$ git init
Initialized empty Git repository in myawesomesoftware/.git/
[]$ ls -a
.    ..   .git
```

Git

- Getting started

```
[]$ mkdir myawesomesoftware
[]$ cd myawesomesoftware/
[]$ git init
Initialized empty Git repository in myawesomesoftware/.git/
[]$ ls -a
.    ..   .git
```

```
[]$ git status
# On branch master
#
# Initial commit
#
nothing to commit (create/copy
files and use "git add" to
track)
```

Git

- Getting started

```
[]$ mkdir myawesomesoftware
[]$ cd myawesomesoftware/
[]$ git init
Initialized empty Git repository in myawesomesoftware/.git/
[]$ ls -a
.    ..   .git
[]$ echo "My awesesome software" > README
```

```
[]$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to
include in what will be
committed)
#
#    README
nothing added to commit but
untracked files present (use
"git add" to track)
```

Git

- Getting started

```
[]$ mkdir myawesomesoftware
[]$ cd myawesomesoftware/
[]$ git init
Initialized empty Git repository in myawesomesoftware/.git/
[]$ ls -a
.    ..   .git
[]$ echo "My awesesome software" > README
[]$ git add README
```

```
[]$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached
<file>..." to unstage)
#
#    new file:   README
#
```

Git

- Getting started

```
[]$ mkdir myawesomesoftware
[]$ cd myawesomesoftware/
[]$ git init
Initialized empty Git repository in
myawesomesoftware/.git/
[]$ ls -a
.    ..   .git
[]$ echo "My awesesome software" > README
[]$ git add README
[]$ git commit -m "Initial commit with README file."
[master (root-commit) b2e92ae] Initial commit with
README file.
 1 file changed, 1 insertion(+)
 create mode 100644 README
```

```
[]$ git status
# On branch master
nothing to commit (working
directory clean)
```

Git

- Getting started

```
[]$ mkdir myawesomesoftware
[]$ cd myawesomesoftware/
[]$ git init
Initialized empty Git repository in
myawesomesoftware/.git/
[]$ ls -a
.     ..    .git
[]$ echo "My awesesome software" > README
[]$ git add README
[]$ git commit -m "Initial commit with README file."
[master (root-commit) b2e92ae] Initial commit with
README file.
 1 file changed, 1 insertion(+)
 create mode 100644 README


[]$ git help
```

Git

- Under the hood
  - Git keeps track of a database of commits
  - Every directory under version control has a *.git* folder
  - A commit consists of [tree, author, timestamp, log message, parent commit(s)]
  - Commits are named with hashes
  - Formally a *directed acyclic graph*

Git

- Working and staging

```
[]$ touch another.txt
[]$ touch athird.txt
[]$ echo “Hope you like it” >> README
[]$ git add another.txt
[]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#    new file:   another.txt
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working 
directory)
#
#    modified:   README
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#    athird.txt
```

Git

- Local workflow

```
[]$ git diff
diff --git a/README b/README
index 4c295d0..a577986 100644
--- a/README
+++ b/README
@@ -1 +1,2 @@
 My awesome software
+Hope you like it
```

Git

- Local workflow

```
[]$ git diff
diff --git a/README b/README
index 4c295d0..a577986 100644
--- a/README
+++ b/README
@@ -1 +1,2 @@
 My awesome software
+Hope you like it
[]$ git commit -a -m “Added more info to the README.”
[master 3aeaa04] Added more info to the README.
 1 file changed, 1 insertion(+)
 create mode 100644 another.txt
[]$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#    athird.txt
nothing added to commit but untracked files present (use "git add" to
track)
```

Git

- Local workflow

```
[]$ git log
commit 3aeaa04154e55dd9cb9e027869dbe71dbfb02537
Author: Jeremy S. Perkins <>
Date:   Tue Jun 11 14:56:24 2013 -0400

    Added more info to the README.

commit acfcf539f4e3d3ce052acfd81153d6e6cc8c2d96
Author: Jeremy S. Perkins <>
Date:   Tue Jun 11 14:32:17 2013 -0400

    Initial commit with README file.
```

Git

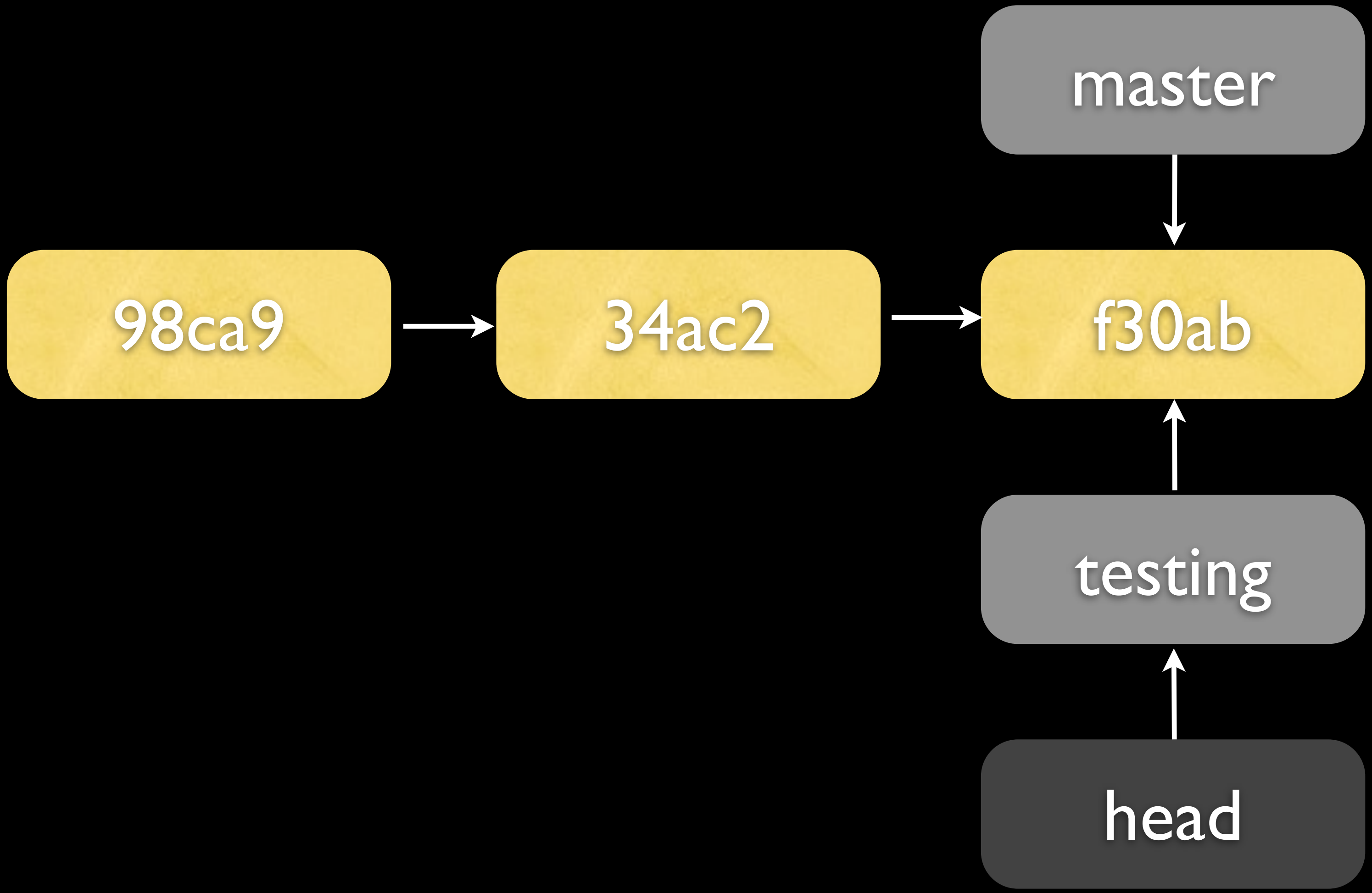
- I have made a huge mistake...
  - Different levels of undo
  - If committed, you can (almost) never lose it:

```
Amend the previous commit
[]$ git commit --amend

Discard changes to files
[]$ echo "This is a huge mistake" > README
[]$ git checkout README

Unstage a change
[]$ git add athird.txt
[]$ git status
...
[]$ git reset HEAD athird.txt
[]$ git status

Create a new commit that removes some old commits
[]$ git revert

Rewind commits. Only if they have not been pushed.
[]$ git reset --hard
```

Git

- Branches

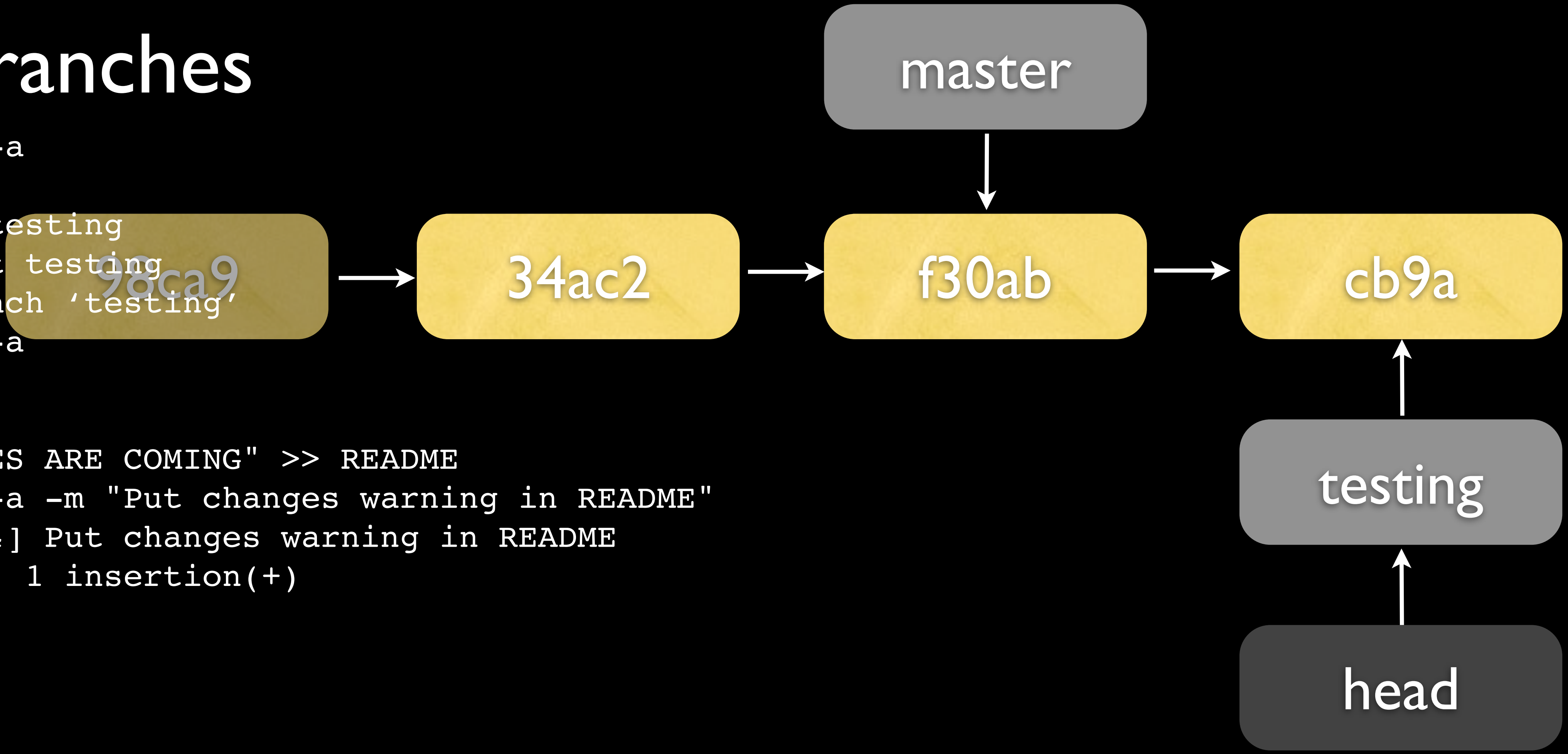
```
[]$ git branch -a
* master
[]$ git branch testing
[]$ git checkout testing
Switched to branch ‘testing’
[]$ git branch -a
  master
* testing
```



Git

- Branches

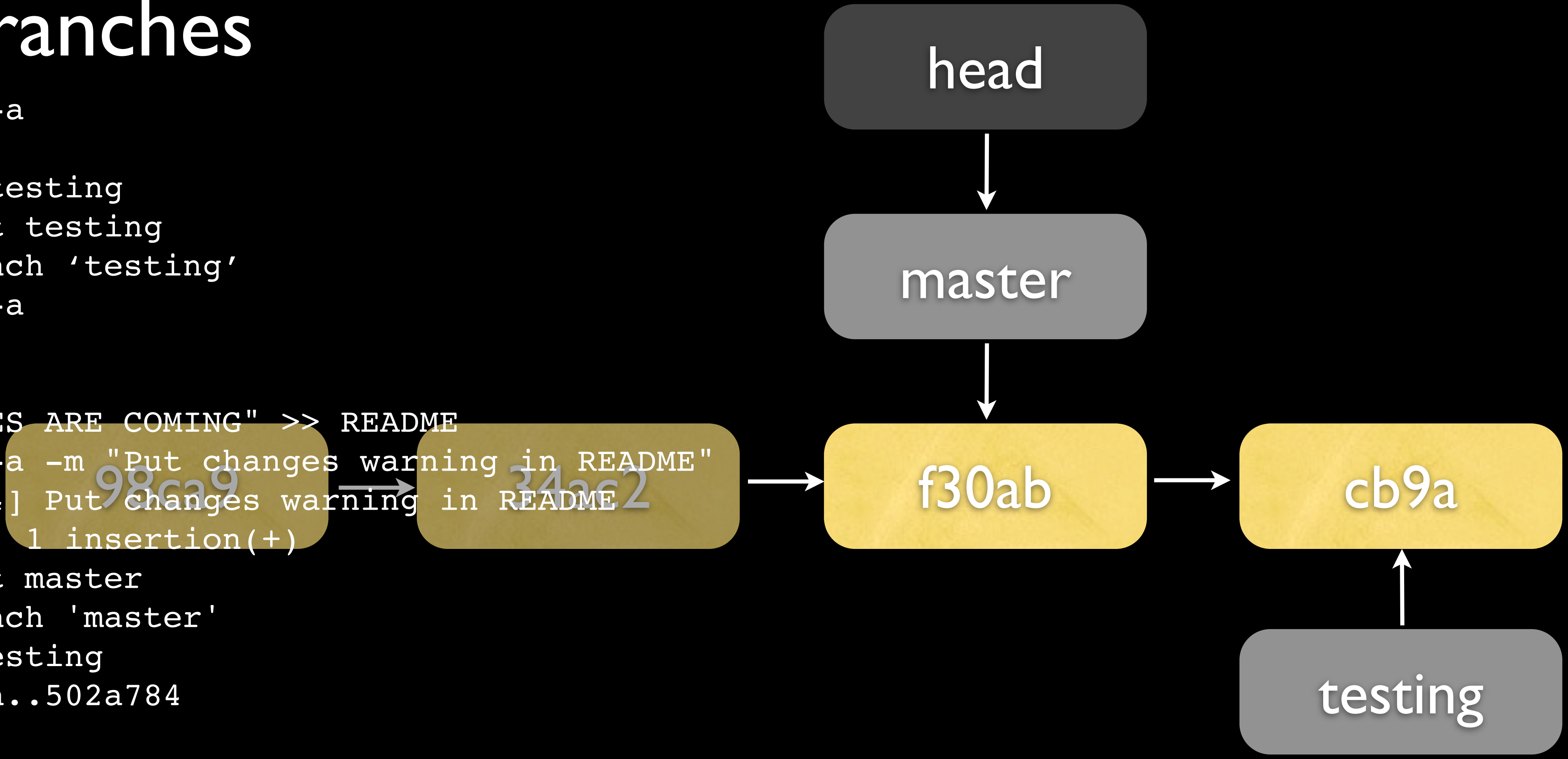
```
[]$ git branch -a
* master
[]$ git branch testing
[]$ git checkout testing
Switched to branch ‘testing’
[]$ git branch -a
  master
* testing
[]$ echo "CHANGES ARE COMING" >> README
[]$ git commit -a -m "Put changes warning in README"
[testing 502a784] Put changes warning in README
 1 file changed, 1 insertion(+)
```



Git

- Branches

```
[]$ git branch -a
* master
[]$ git branch testing
[]$ git checkout testing
Switched to branch ‘testing’
[]$ git branch -a
  master
* testing
[]$ echo "CHANGES ARE COMING" >> README
[]$ git commit -a -m "Put changes warning in README"
[testing 502a784] Put changes warning in README
 1 file changed, 1 insertion(+)
[]$ git checkout master
Switched to branch 'master'
[]$ git merge testing
Updating ac77bba..502a784
Fast-forward
 README | 1 +
 1 file changed, 1 insertion(+)
```



Git

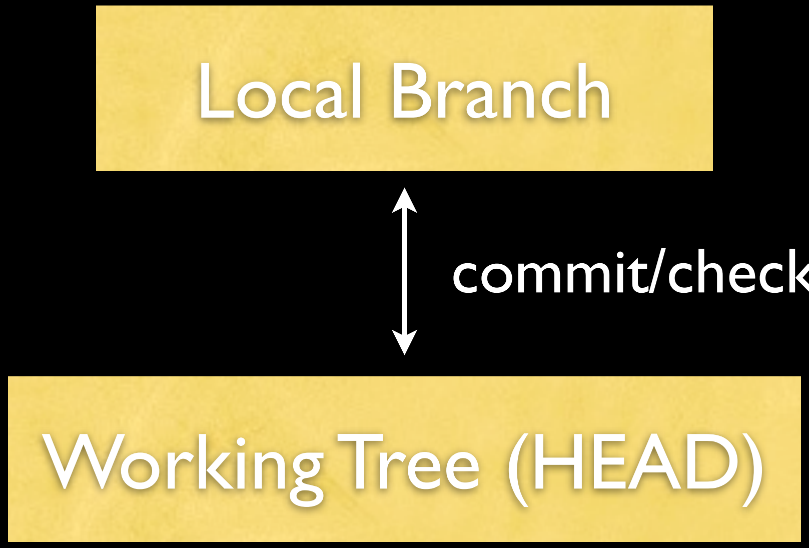
- Collaboration

```
Via a local shared repository
[]$ git clone /path/on/shared/disk

Via a git server
[]$ git clone git://git-server.com/...

Via ssh
[]$ git clone user@host:/path/to/repo.git

Via http(s)
[]$ git clone https://host/repo.git
```

Git

- Collaboration

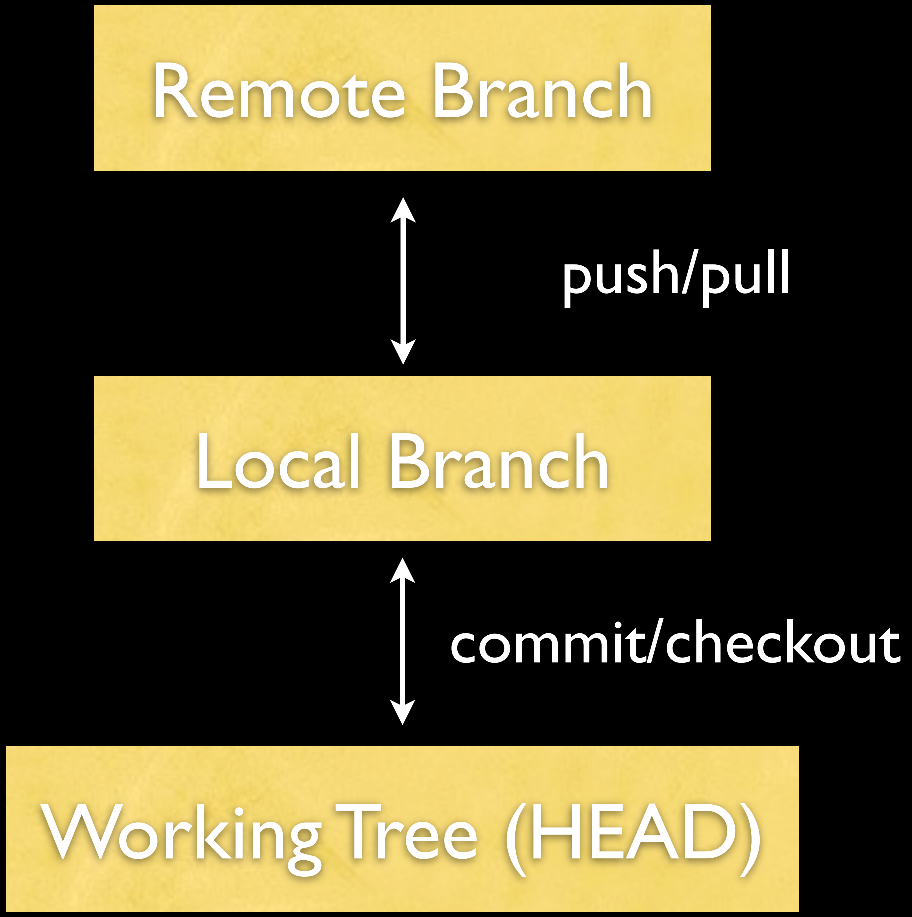
```
Via a local shared repository
[]$ git clone /path/on/shared/disk

Via a git server
[]$ git clone git://git-server.com/...

Via ssh
[]$ git clone user@host:/path/to/repo.git

Via http(s)
[]$ git clone https://host/repo.git

[]$ echo "My 5 cents" >> README
[]$ git diff
...
[]$ git commit -a -m "Changed README to include my 5 cents."
[master 30c6bbf] Changed README to include my 5 cents.
 1 file changed, 1 insertion(+)
```

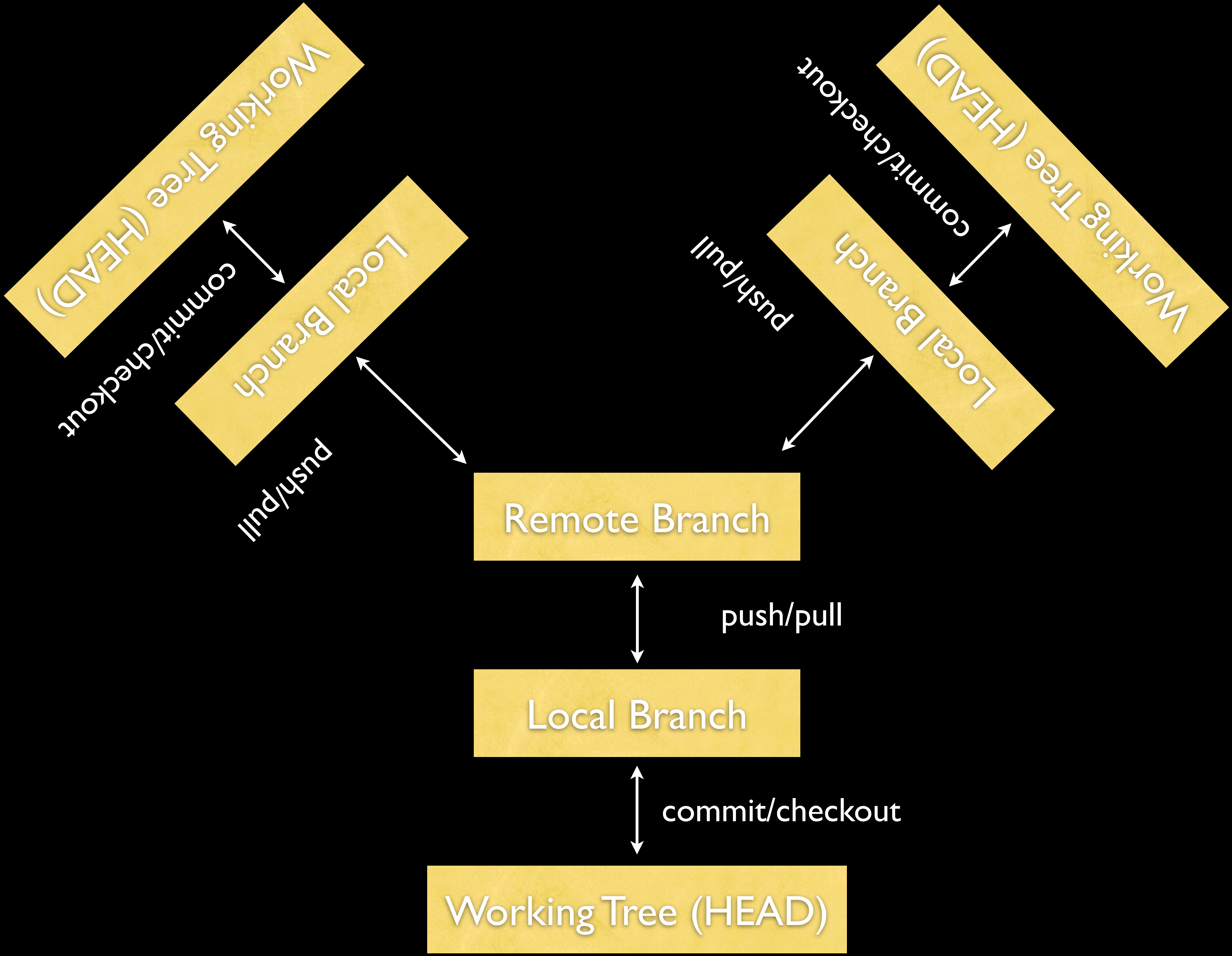


Git

- Collaboration

```
Via a local shared repository
[]$ git clone /path/on/shared/disk

Via a git server
[]$ git clone git://git-server.com/...

Via ssh
[]$ git clone user@host:/path/to/repo.git

Via http(s)
[]$ git clone https://host/repo.git

[]$ echo "My 5 cents" >> README
[]$ git diff
...
[]$ git commit -a -m "Changed README to include my 5 cents."
[master 30c6bbf] Changed README to include my 5 cents.
 1 file changed, 1 insertion(+)
[]$ git pull
[]$ git push
```





Working Tree (HEAD)
commit/checkout
Local Branch
push/pull
Working Tree (HEAD)
commit/checkout
Local Branch
push/pull
Remote Branch
push/pull
Local Branch
commit/checkout
Working Tree (HEAD)

Git

- Setting up a shared repo

```
[]$ ssh myserver
[]$ cd /path/to/repos
[]$ mkdir myrepo.git
[]$ cd myrepo.git
[]$ git init --bare --shared
[]$ exit
```

Git

- Setting up a shared repo

```
[]$ ssh myserver
[]$ cd /path/to/repos
[]$ mkdir myrepo.git
[]$ cd myrepo.git
[]$ git init --bare --shared
[]$ exit

[]$ cd /path/to/local/code
[]$ git remote add origin ssh://myserver/path/to/repos/myrepo.git
[]$ git push -u origin master
```

Git

- Resolving conflicts

```
[]$ git pull
CONFLICT (content): Merge conflict in file.txt
```

Git

- Resolving conflicts

```
[]$ git pull
CONFLICT (content): Merge conflict in file.txt
[]$ cat file.txt
<<<<<<<< HEAD:file.txt
Hello world
==========
Goodbye
>>>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt
```

Git

- Resolving conflicts

```
[]$ git pull
CONFLICT (content): Merge conflict in file.txt
[]$ cat file.txt
<<<<<<<< HEAD:file.txt
Hello world
==========
Goodbye
>>>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt

[]$ emacs file.txt
```

Git

- Resolving conflicts

```
[]$ git pull
CONFLICT (content): Merge conflict in file.txt
[]$ cat file.txt
<<<<<<<< HEAD:file.txt
Hello world
==========
Goodbye
>>>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt

[]$ emacs file.txt
[]$ git add file.txt
[]$ git commit -m “Merged conflicts in file.txt”
```

Git

- Github Flow

```
[]$ git clone ...
[]$ git checkout -b my_new_feature
[]$ emacs crazy_feature.py
```

Git

- Github Flow

```
[]$ git clone ...
[]$ git checkout -b my_new_feature
[]$ emacs crazy_feature.py
[]$ git commit ...
[]$ git rebase master
[]$ git push -u origin my_new_feature
Tell someone about your new branch and iterate... until
```

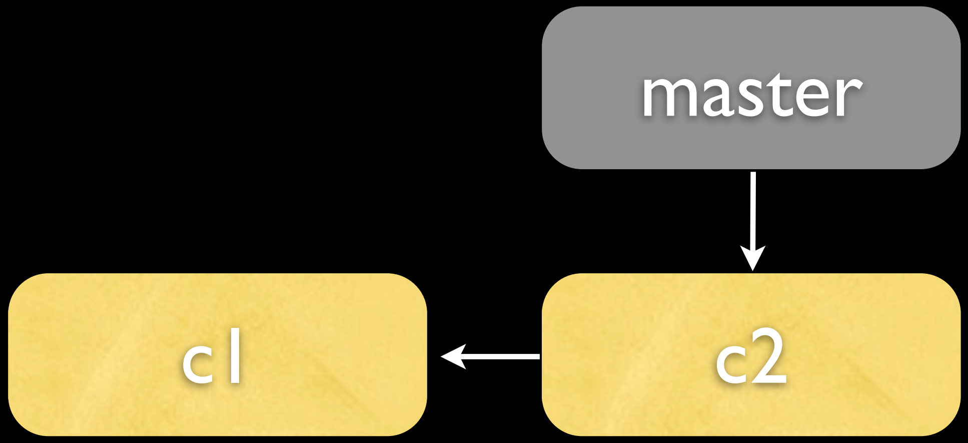
Git

- Github Flow

```
[]$ git clone ...
[]$ git checkout -b my_new_feature
[]$ emacs crazy_feature.py
[]$ git commit ...
[]$ git rebase master
[]$ git push -u origin my_new_feature
Tell someone about your new branch and iterate... until
On branch master
[]$ git merge my_new_feature
[]$ git pull
[]$ git push
Anything in master is deployable
```

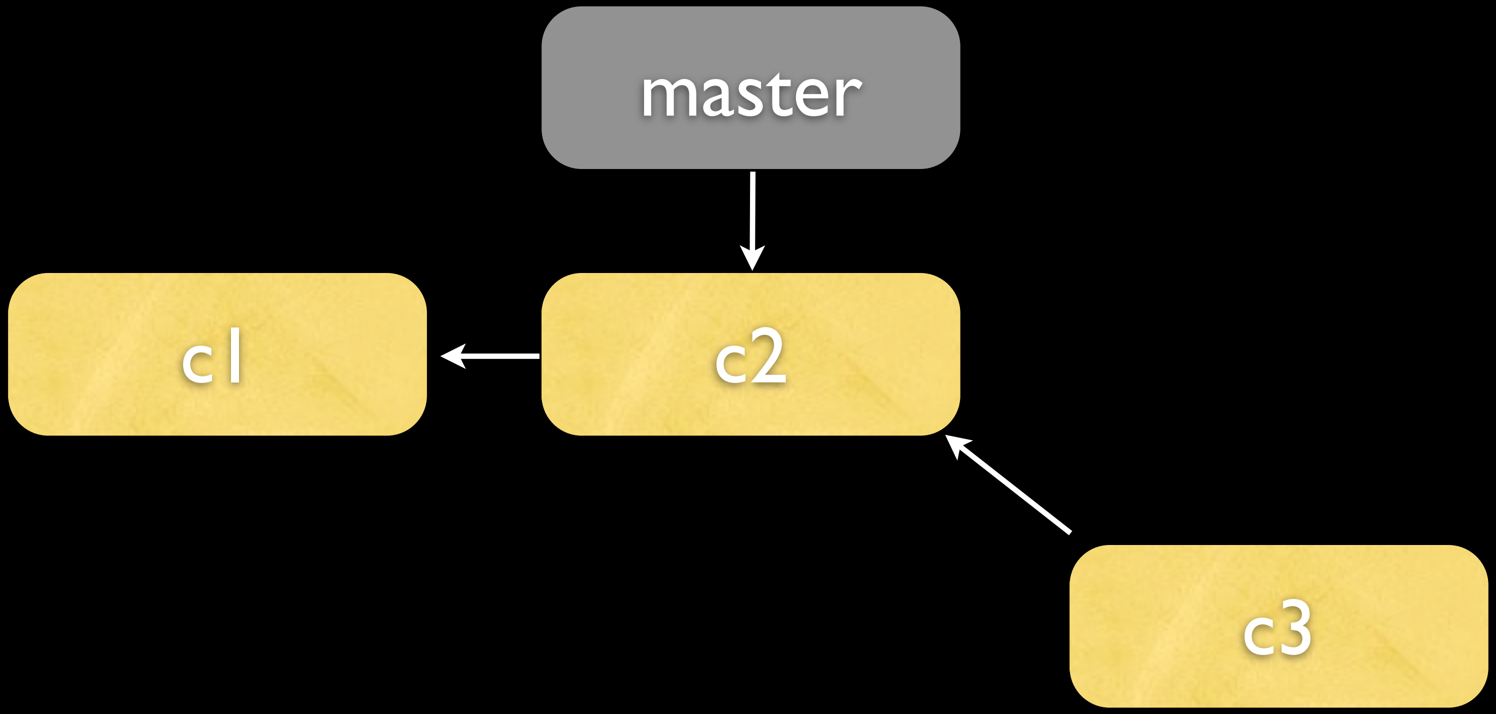
Git

- Merge



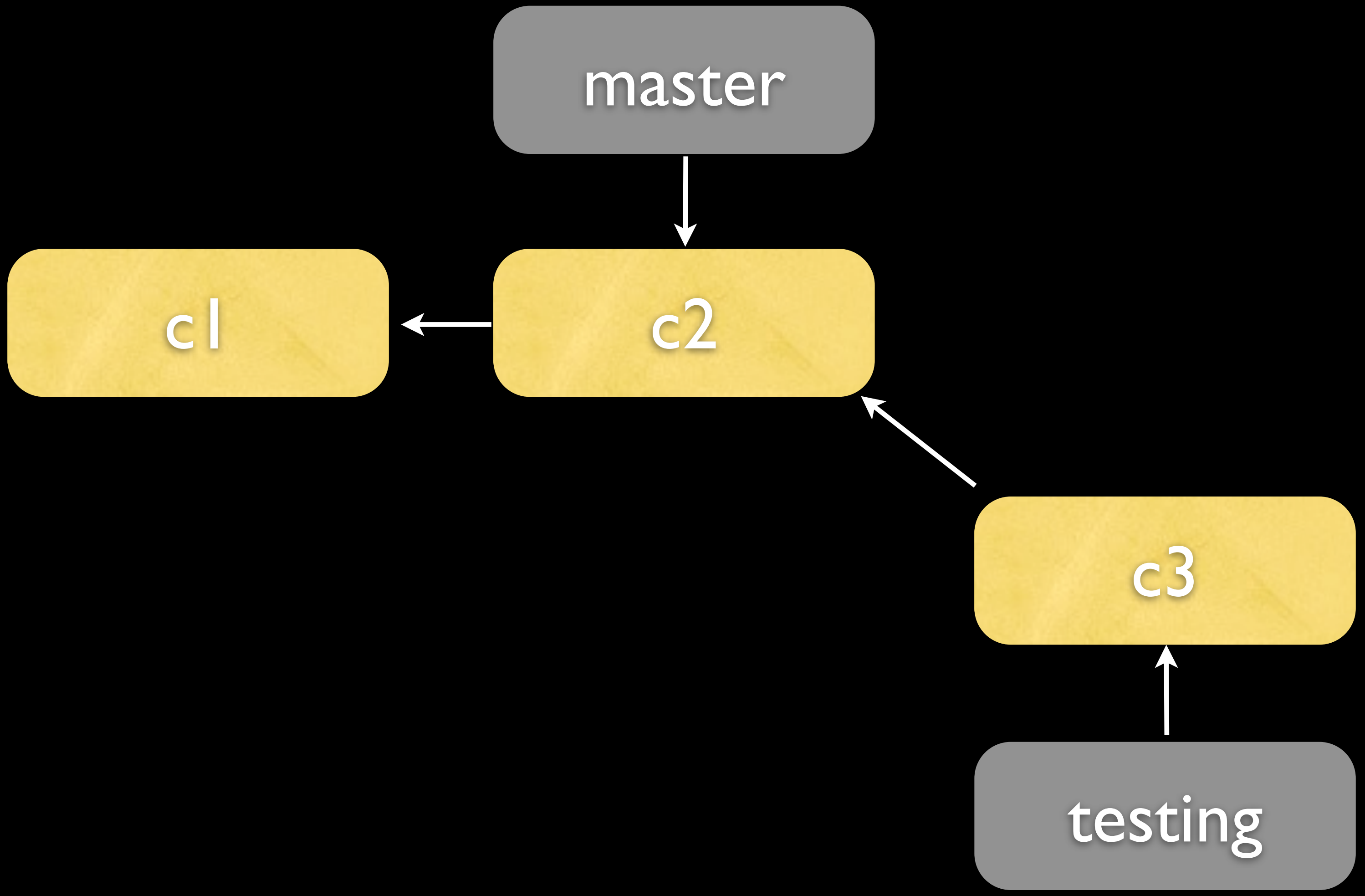
Git

- Merge



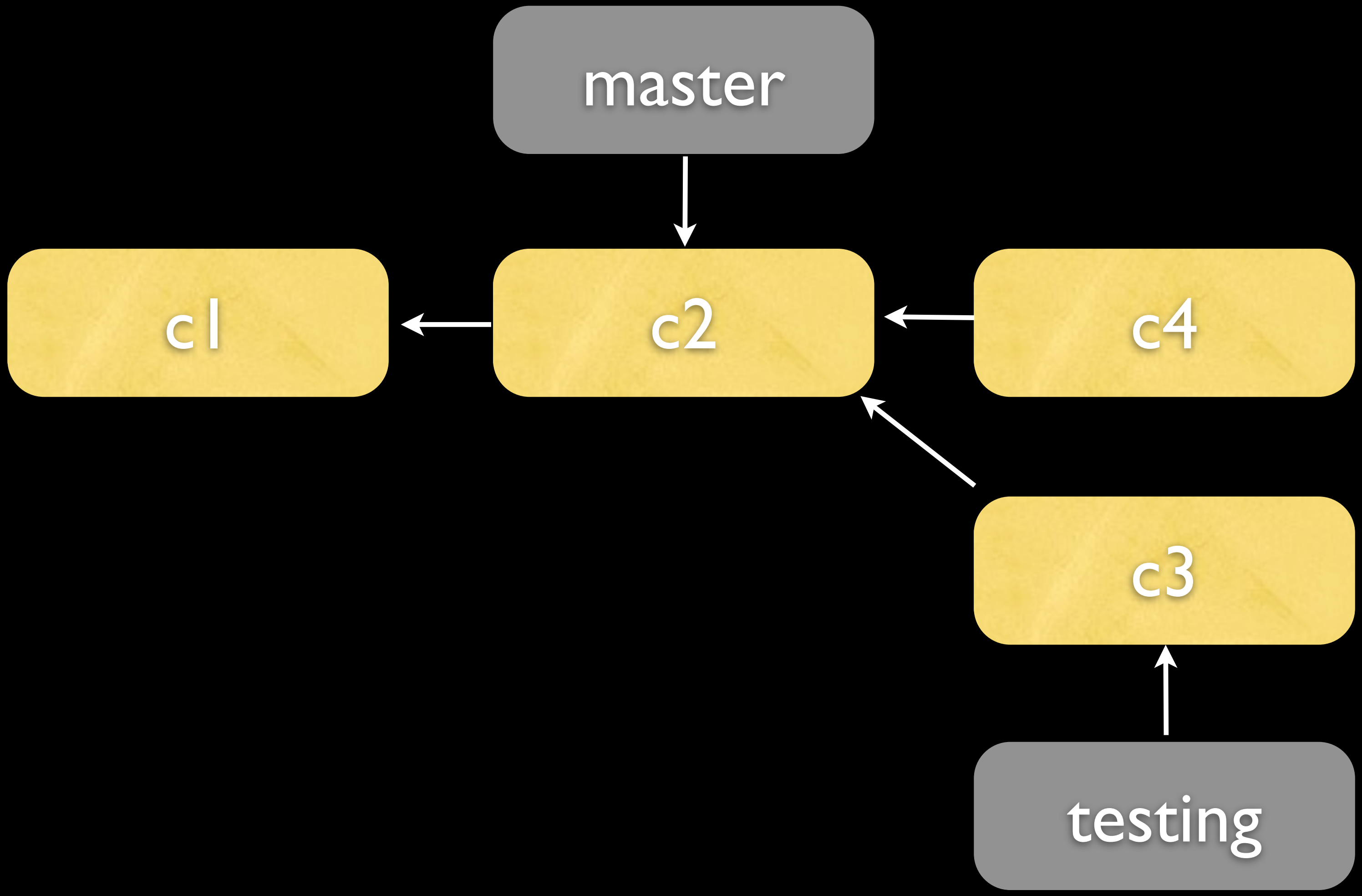
Git

- Merge



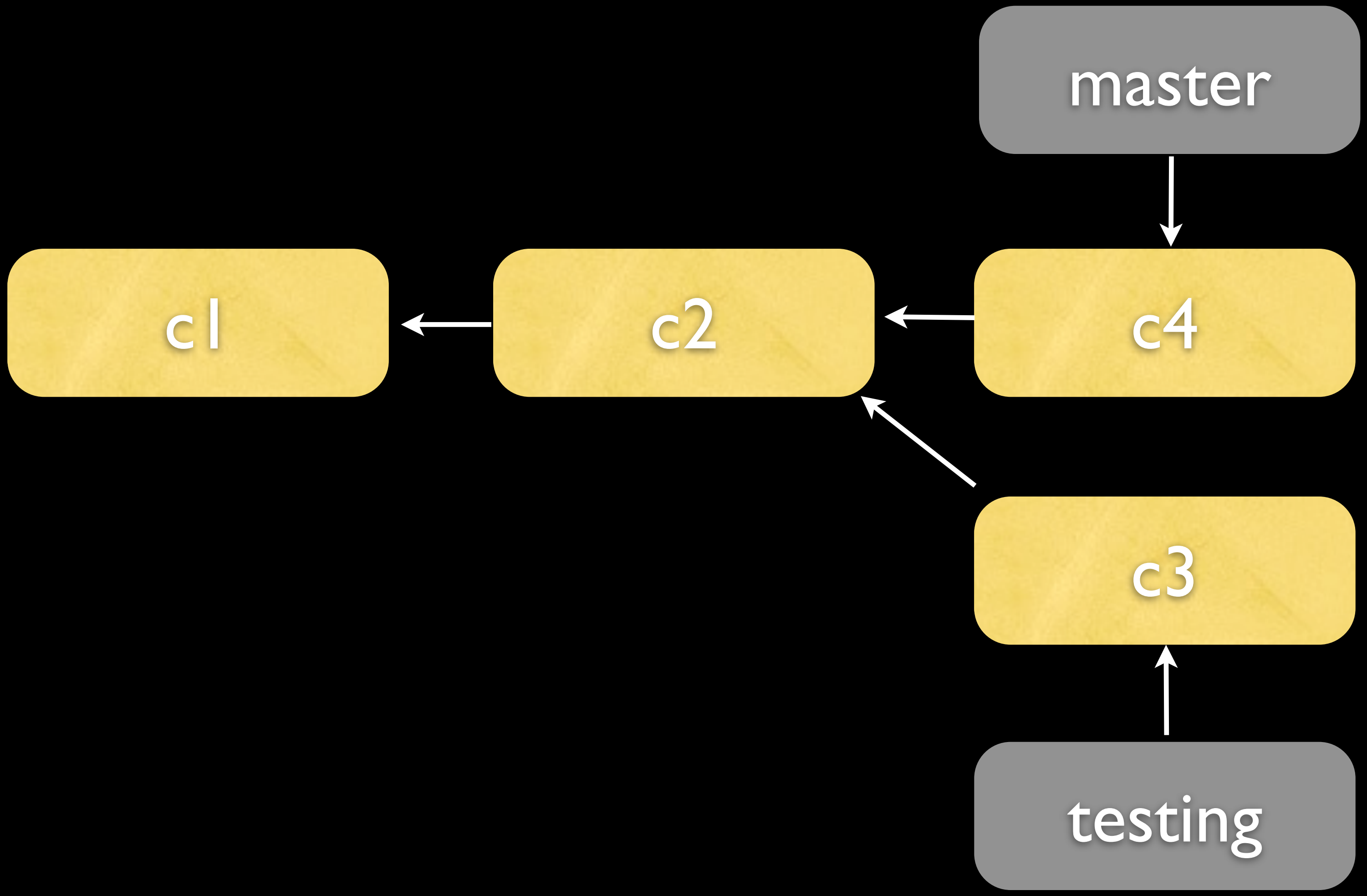
Git

- Merge



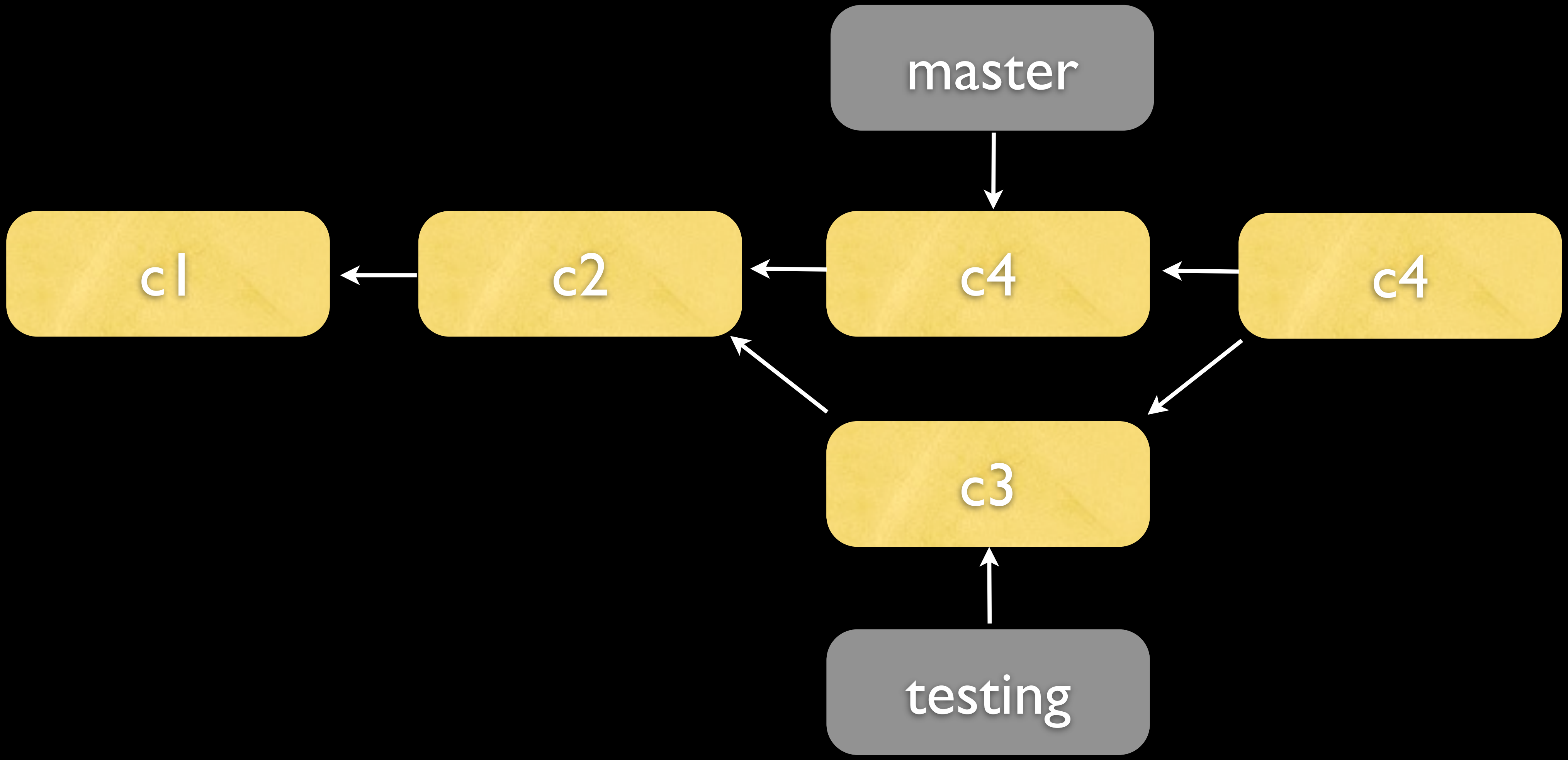
Git

- Merge



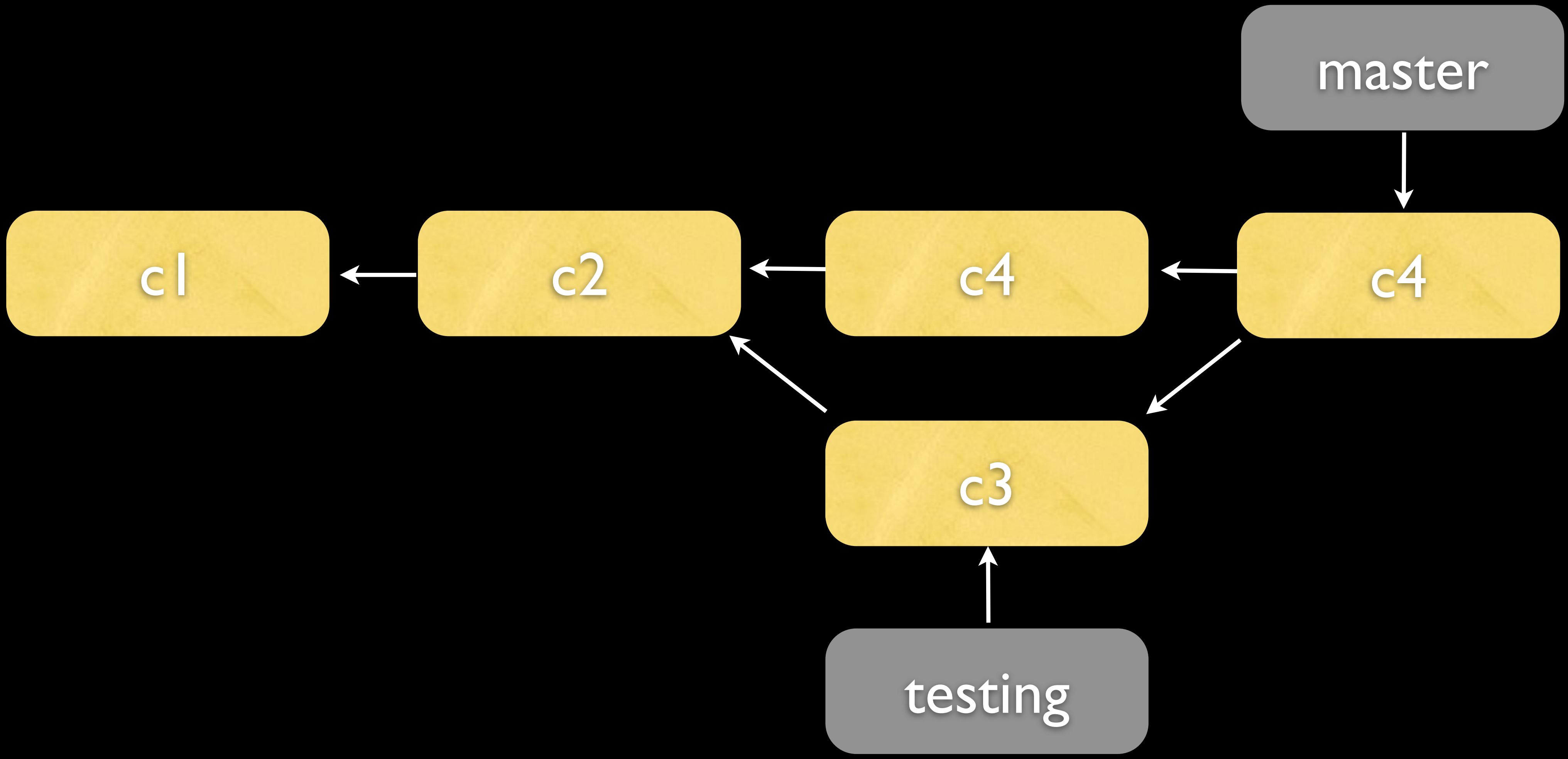
Git

- Merge



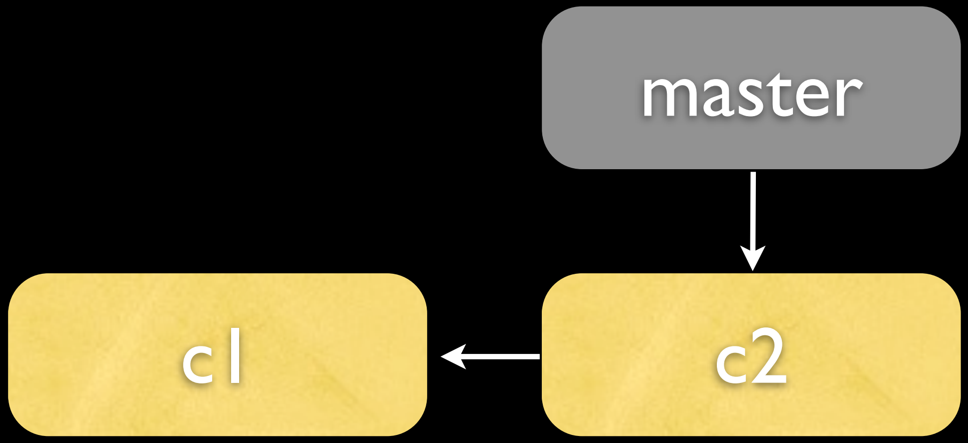
Git

- Merge



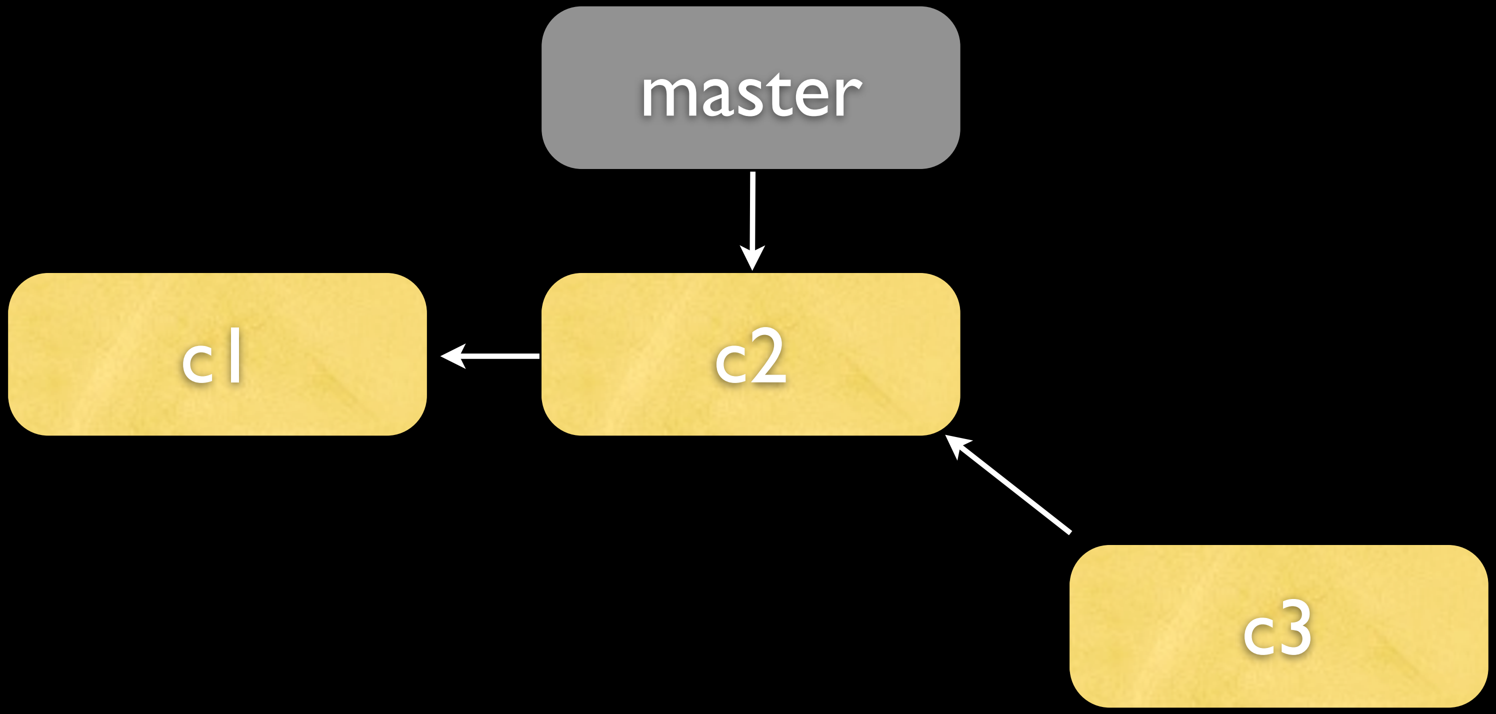
Git

- Rebase



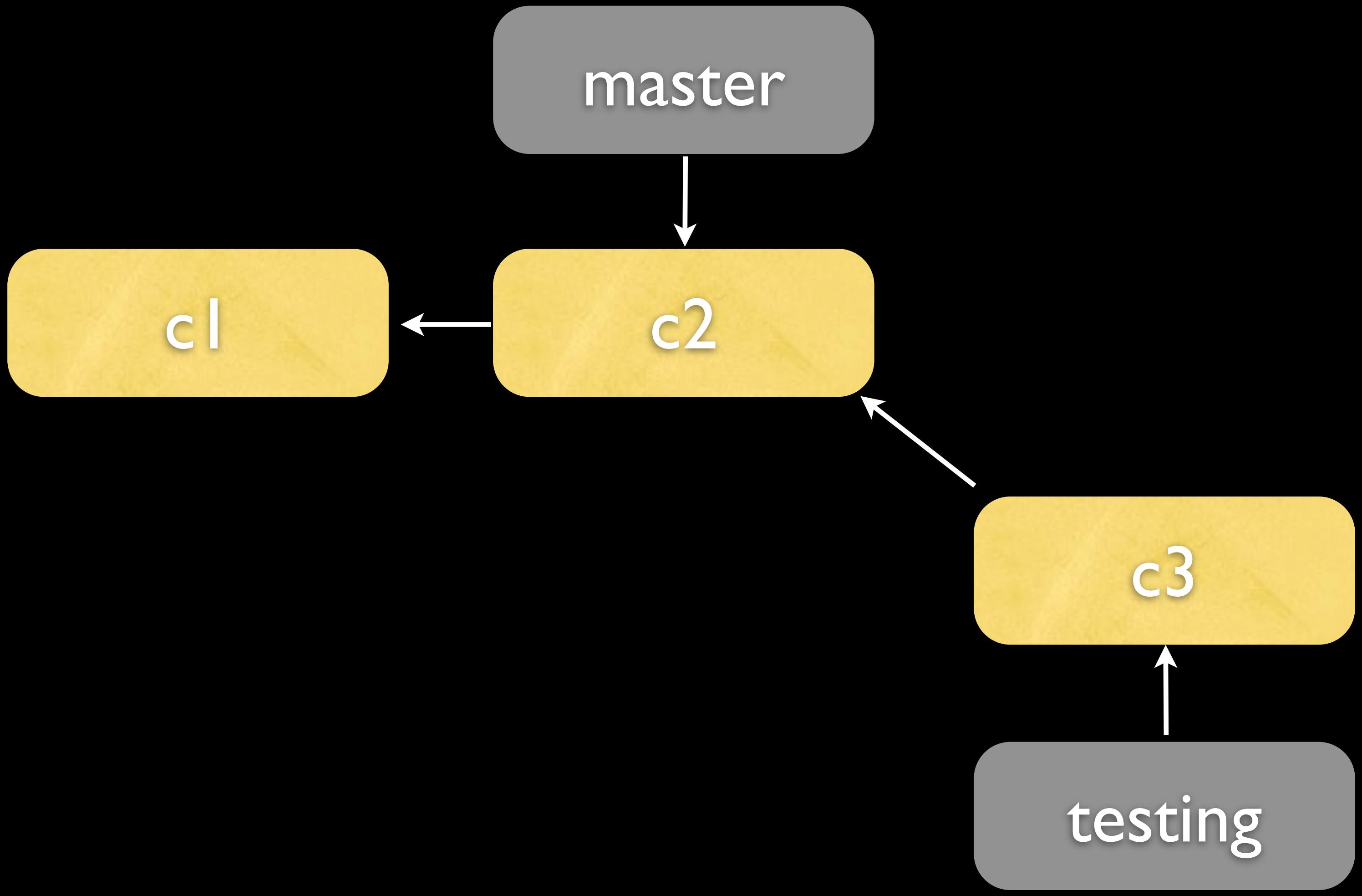
Git

- Rebase



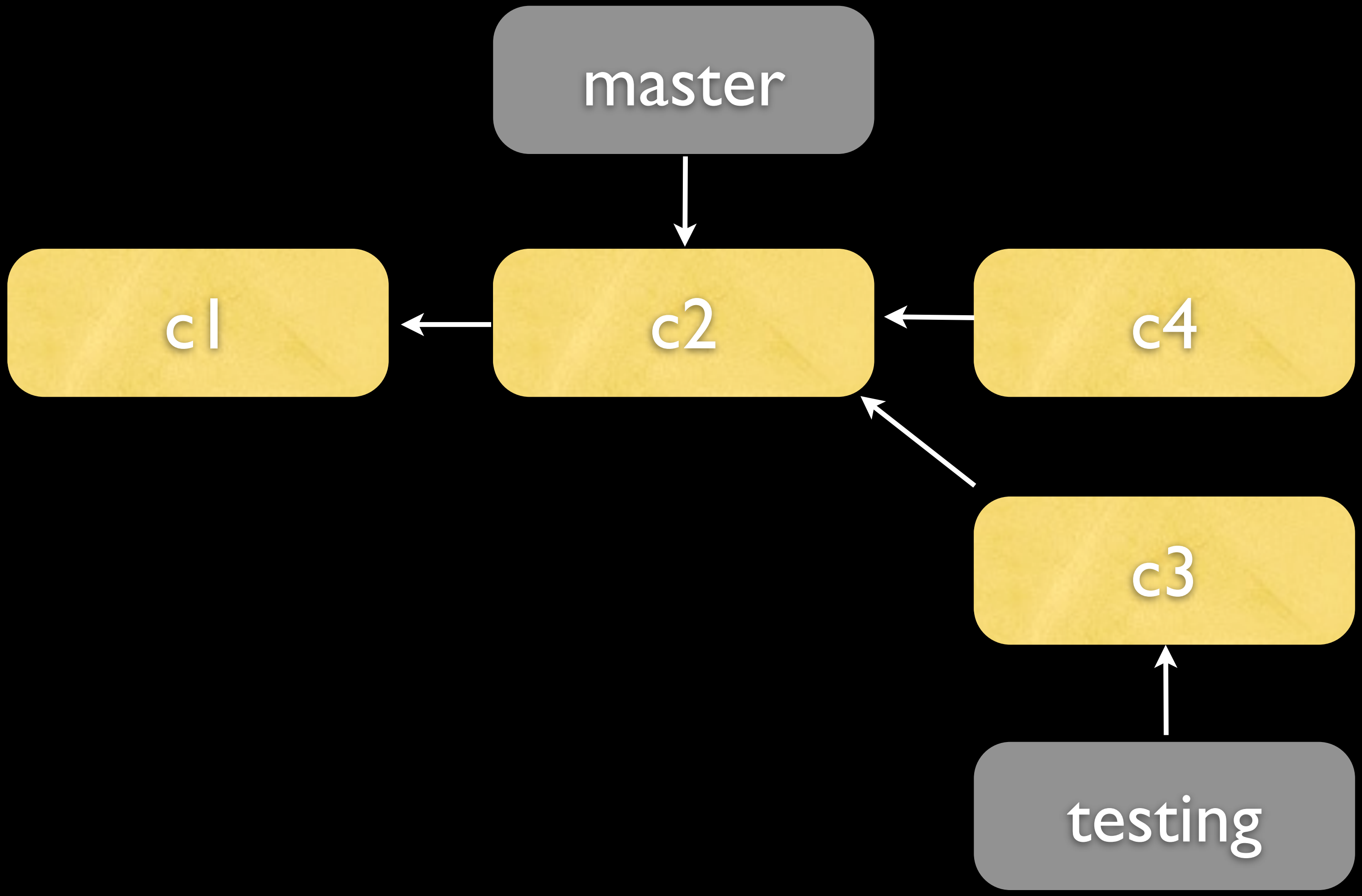
Git

- Rebase



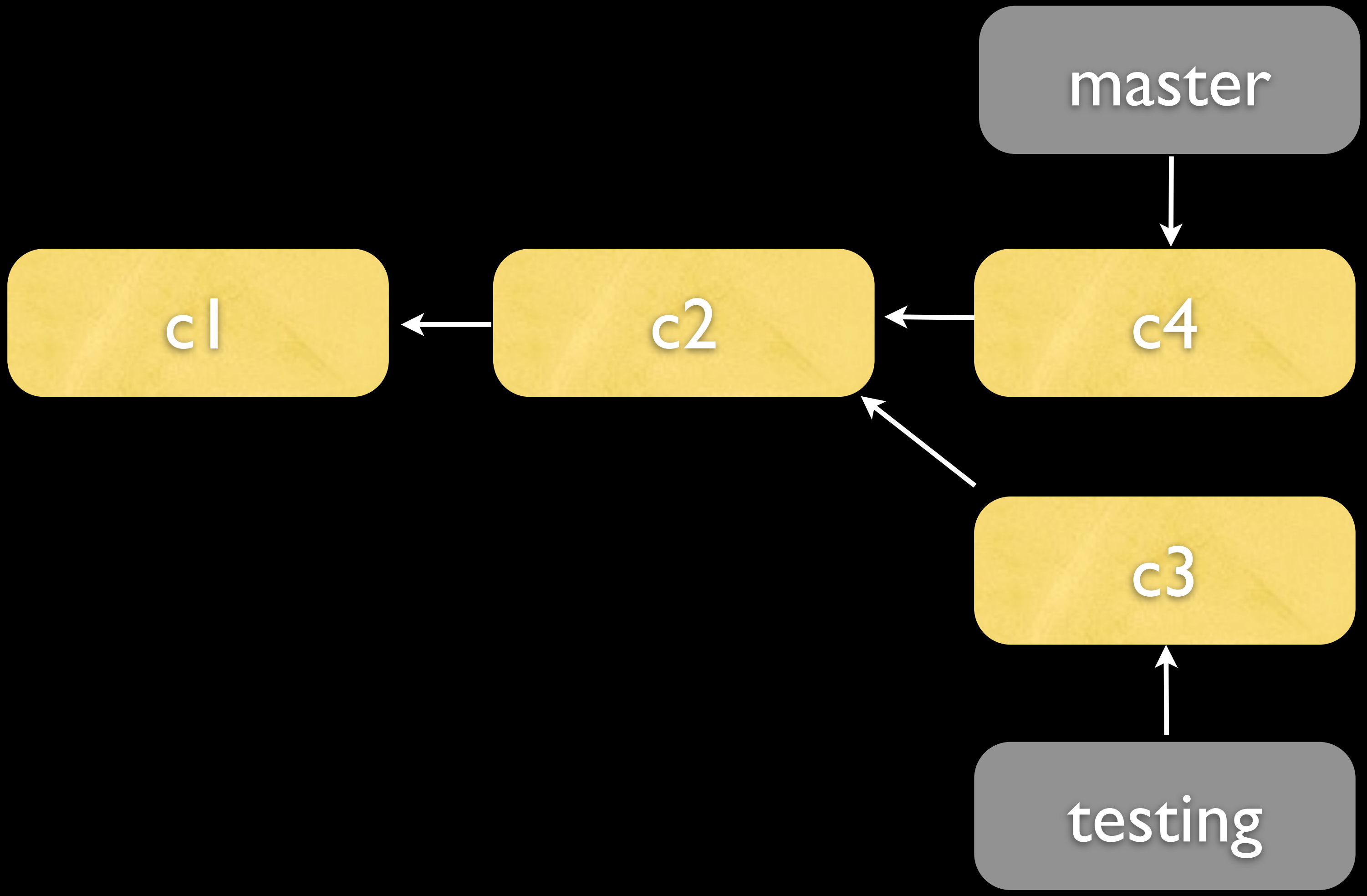
Git

- Rebase



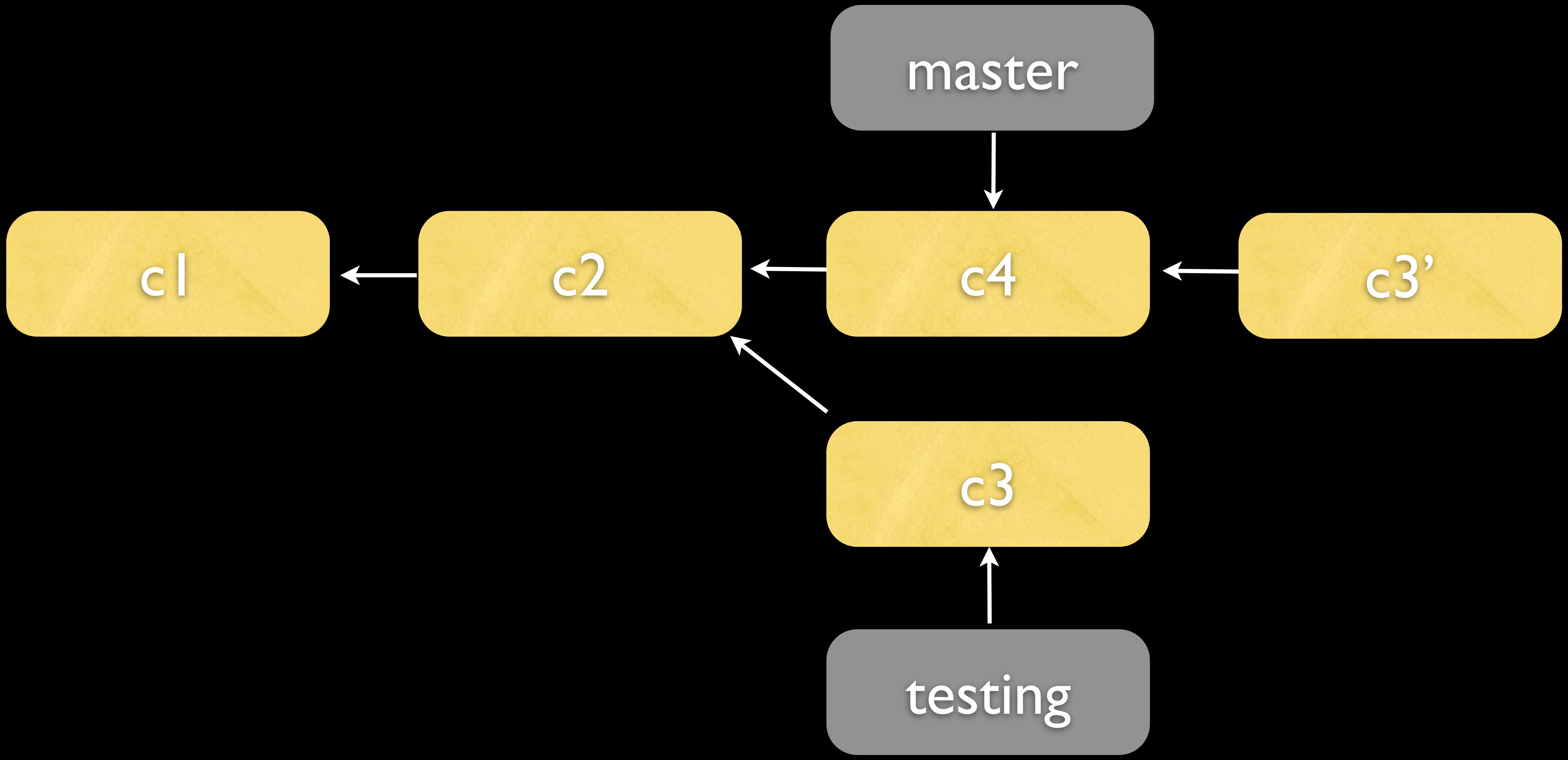
Git

- Rebase



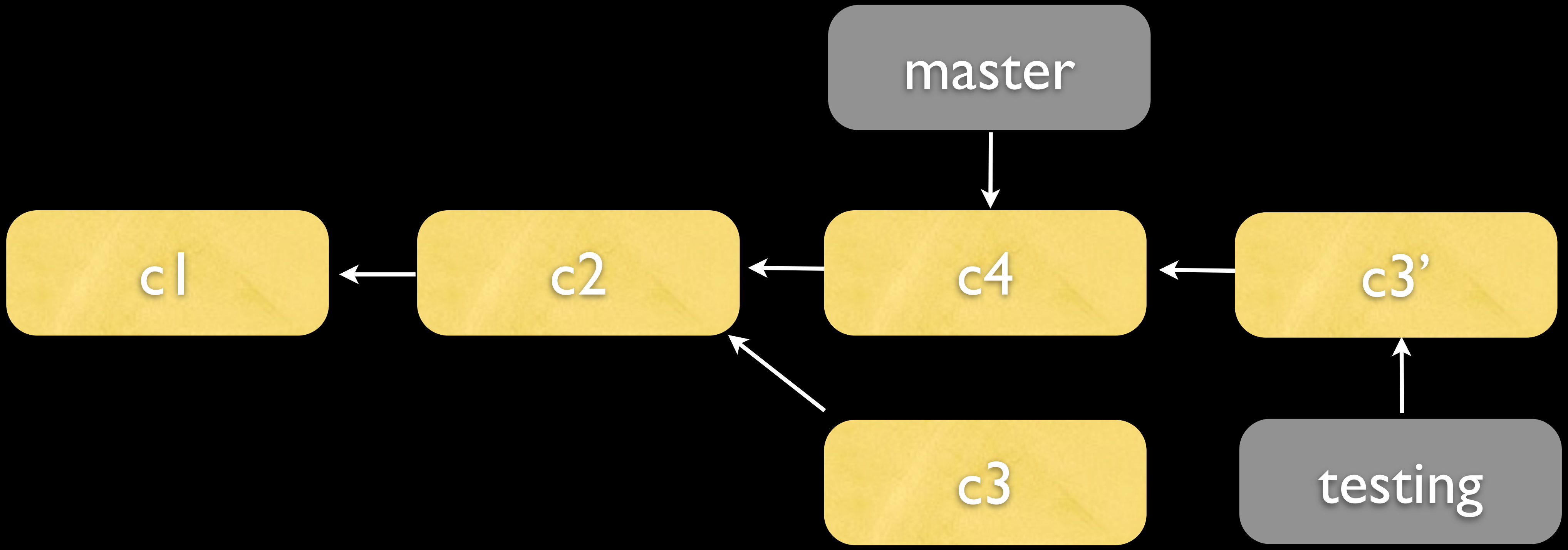
Git

- Rebase



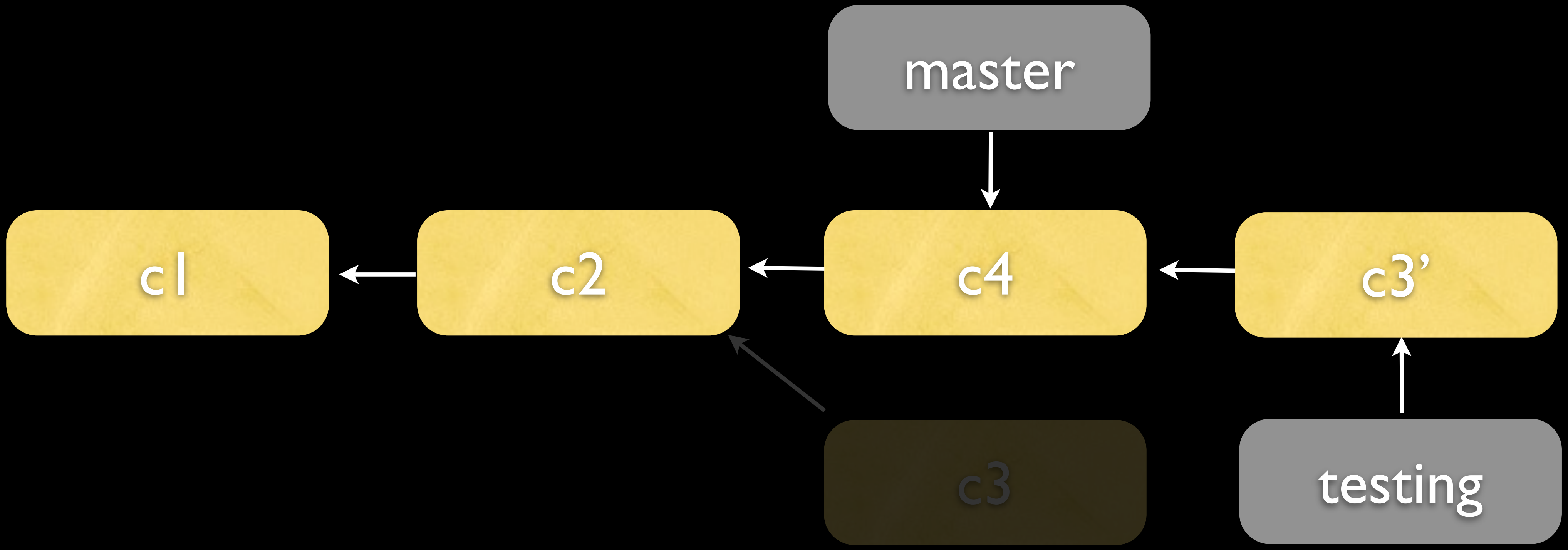
Git

- Rebase



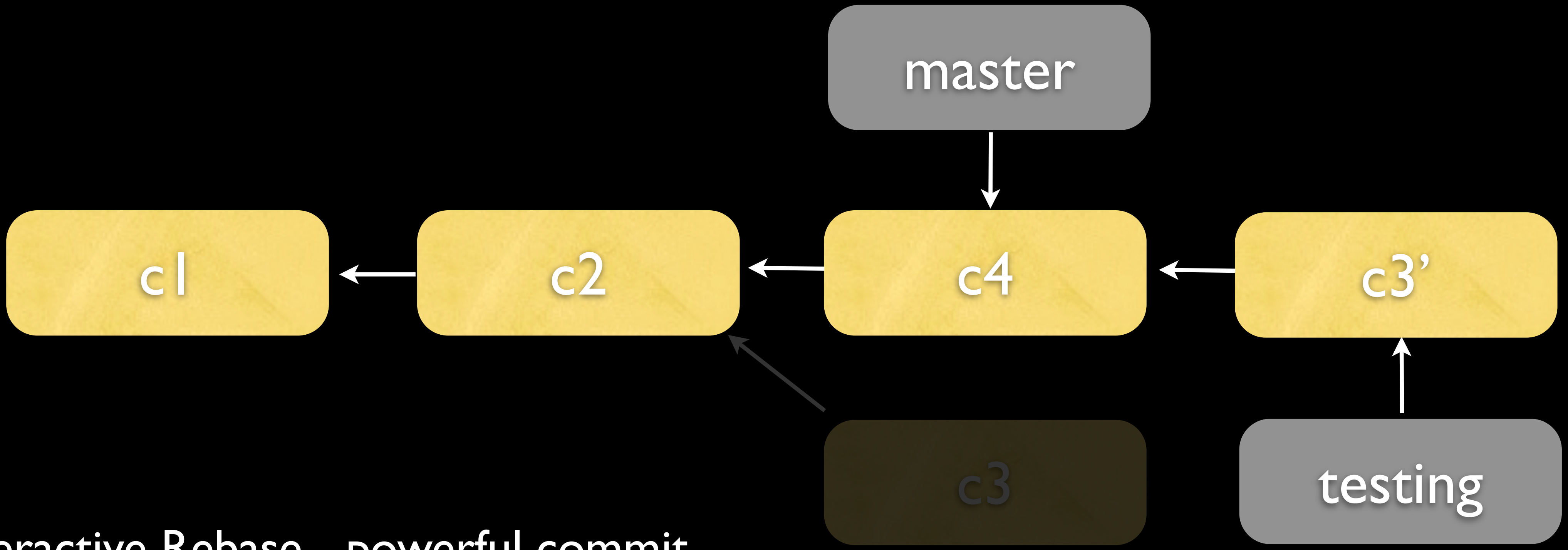
Git

- Rebase



Git

- Rebase



- Interactive Rebase - powerful commit management
- “Do not rebase commits that you have pushed to a public repository!”

Git

- Tagging

```
[]$ git tag -a v1.0 -m "1.0 release."
[]$ git tag
v1.0
[]$ git show v1.0
tag v1.0
...
```

Git

- Tagging

```
[]$ git tag -a v1.0 -m "1.0 release."
[]$ git tag
v1.0
[]$ git show v1.0
tag v1.0
...
[]$ echo "New tag info." >> README
[]$ git commit -a -m "README for new tag."
[master 9747131] README for new tag.
 1 file changed, 1 insertion(+)
[]$ git tag -a v2.0 -m "2.0 release."
[]$ git show v2.0
...
```

Git

- Tagging

```
[]$ git tag -a v1.0 -m "1.0 release."
[]$ git tag
v1.0
[]$ git show v1.0
tag v1.0
...
[]$ echo "New tag info." >> README
[]$ git commit -a -m "README for new tag."
[master 9747131] README for new tag.
 1 file changed, 1 insertion(+)
[]$ git tag -a v2.0 -m "2.0 release."
[]$ git show v2.0
...

[]$ git push origin v2.0
```

https://github.com

Git

- Github?
  - You code is in the cloud
  - Handles all of the plumbing of code collaboration
  - Adds project management and social components
  - Free for open source
  - There’s a NASA group (https://github.com/nasa)