The slide features decorative horizontal lines at the top and bottom. These lines consist of a series of small, empty circles connected by colored lines. The colors used include green, orange, blue, yellow, and purple. The lines are arranged in a way that they appear to be branching and merging, similar to a version control system's history. The top line starts with a green line, then branches into orange, blue, and yellow, before merging back into green. The bottom line starts with a blue line, then branches into orange, green, yellow, and purple, before merging back into blue.

# git started with git

Nick Quaranto

[nick@quaran.to](mailto:nick@quaran.to)

NH.rb April 2009

The slide features a decorative border at the top and bottom. The top border consists of a series of small white circles connected by horizontal lines in green, orange, blue, and yellow. The bottom border is similar, with additional horizontal lines in purple and blue. The word "whoami" is centered in a large, black, sans-serif font.

# whoami

- 5th year Software Engineering Major at RIT
- Intern at Thoughtbot
- Blogger at
  - <http://github.com/blog>
  - <http://gitready.com>
  - <http://litanyagainstfear.com>

A diagram illustrating Git commit history. It features two horizontal timelines of black circles representing commits. The top timeline has a green line above it, the middle has an orange line, and the bottom has a blue line. These lines represent branches. The top branch (green) has a commit that branches off from the middle branch (orange), followed by several commits. The middle branch (orange) has a commit that branches off from the bottom branch (blue), followed by several commits. The bottom branch (blue) has a commit that branches off from the top branch (green), followed by several commits. The lines are colored green, orange, and blue. The text "git pull origin slides" is centered in the middle of the diagram.

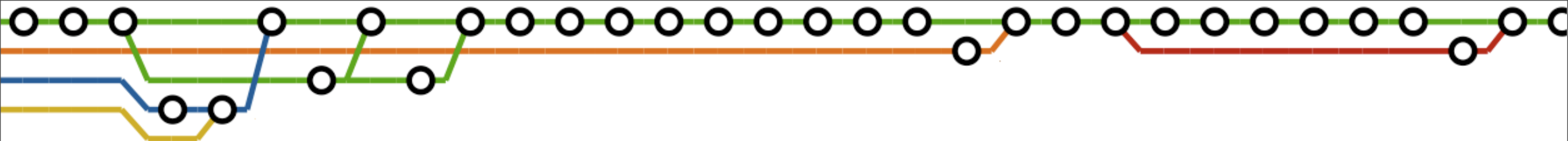
git pull origin slides

<http://drop.io/gitstarted>



# what's in store

- ok, no more puns
- i'm a ruby developer, not a kernel hacker
- history lesson
- concepts and principles
- basic workflow

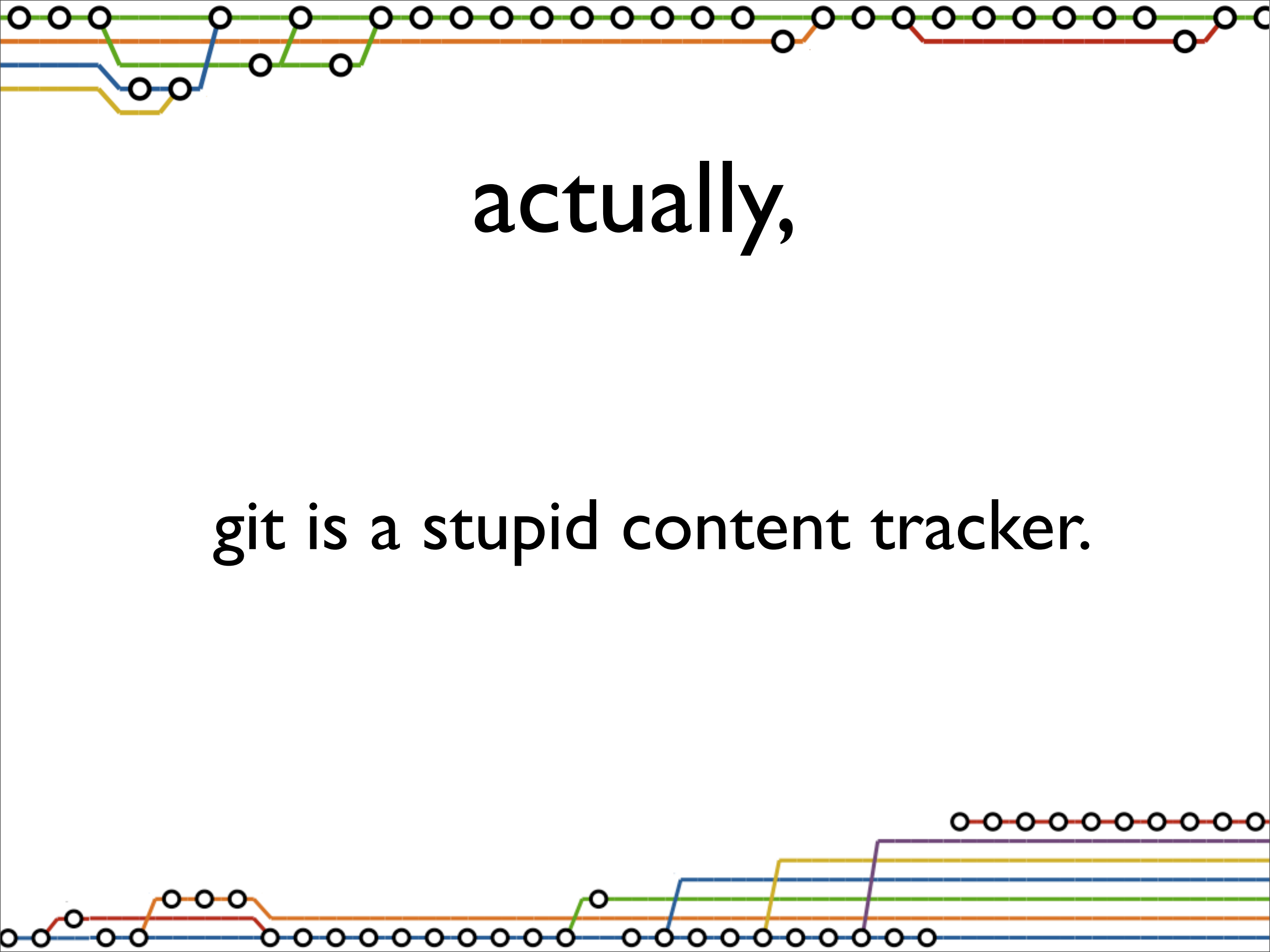


# what is git?

“Git is a free distributed revision control, or software source code management project with an emphasis on being fast.”

[http://en.wikipedia.org/wiki/Git\\_\(software\)](http://en.wikipedia.org/wiki/Git_(software))





actually,

git is a stupid content tracker.

The slide features decorative horizontal lines with small circles at the top and bottom. The top line is green and has several branches (orange, blue, yellow) that merge back into it. The bottom line is blue and has several branches (red, orange, green, blue, yellow, purple) that merge back into it.

# git's history

- Originally created to handle Linux kernel development
- Everything else sucked.
- Now used by plenty of other projects and web frameworks that don't scale



# design motives

- Do exactly the opposite of CVS/SVN
- Support distributed workflow
- Guard against data corruption
- Be ridiculously fast

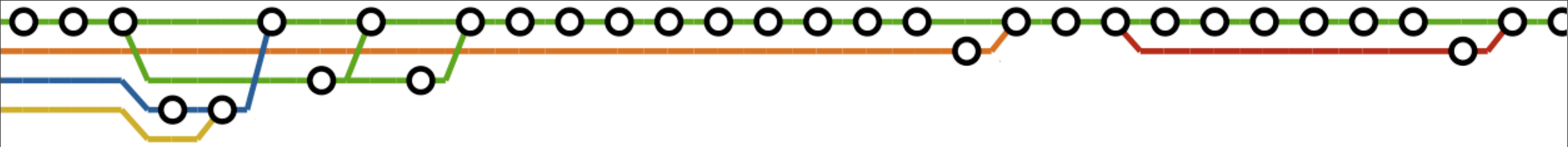




# principles behind git

- distributed and parallel development
- one project, one repository
- never lose data without intervention
- unix philosophy built in





# repos

- In a `.git` directory:
  - A set of commit objects
  - A set of references to commits (heads)
  - More stuff, but don't worry about it.





# commits

- A snapshot of the project at a given time
- trees: subdirectories
- blobs: files
- Link to parent commit(s)
- 40 character SHA1 hash

# the object model

pointer to a  
snapshot

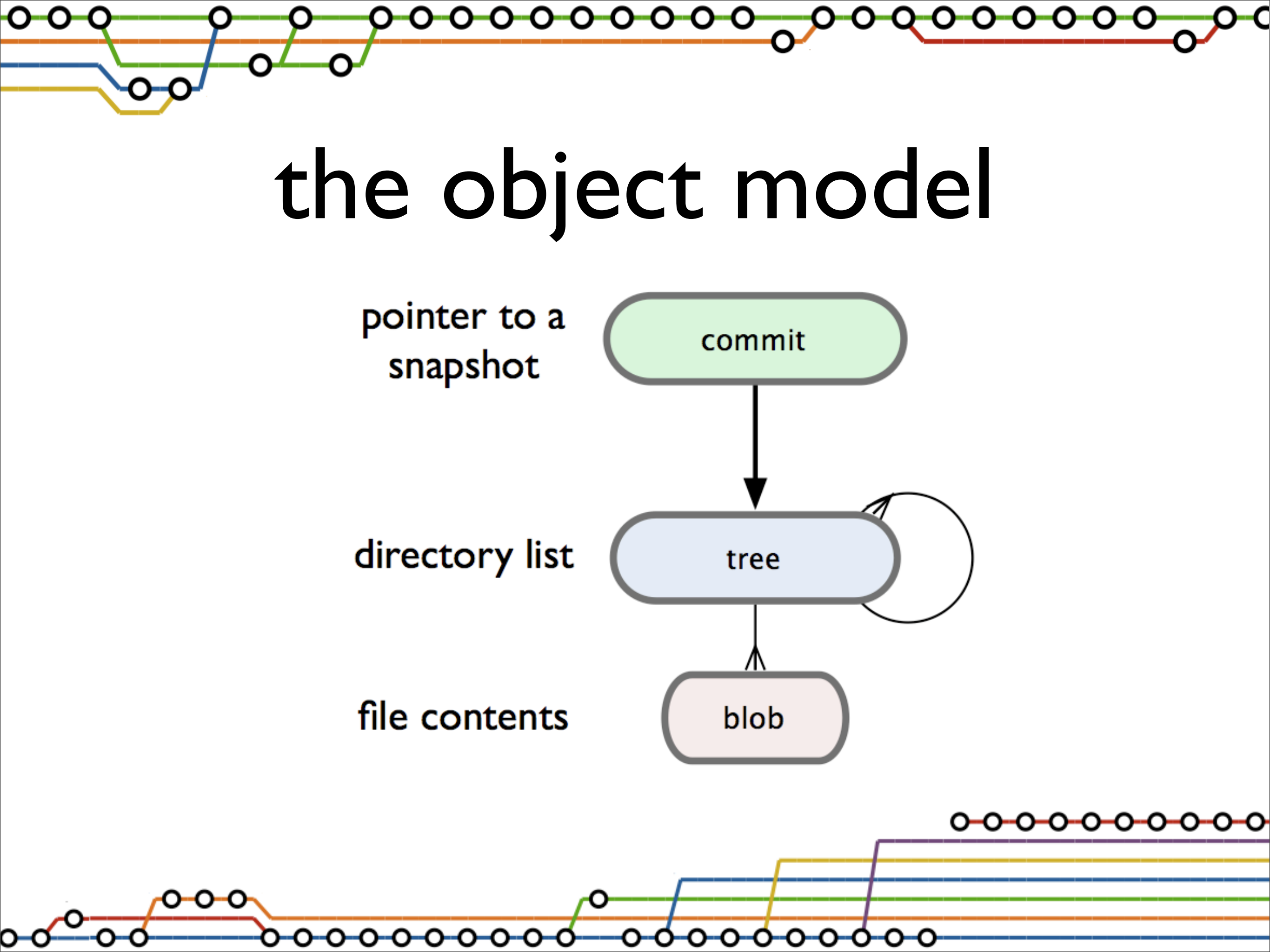
commit

directory list

tree

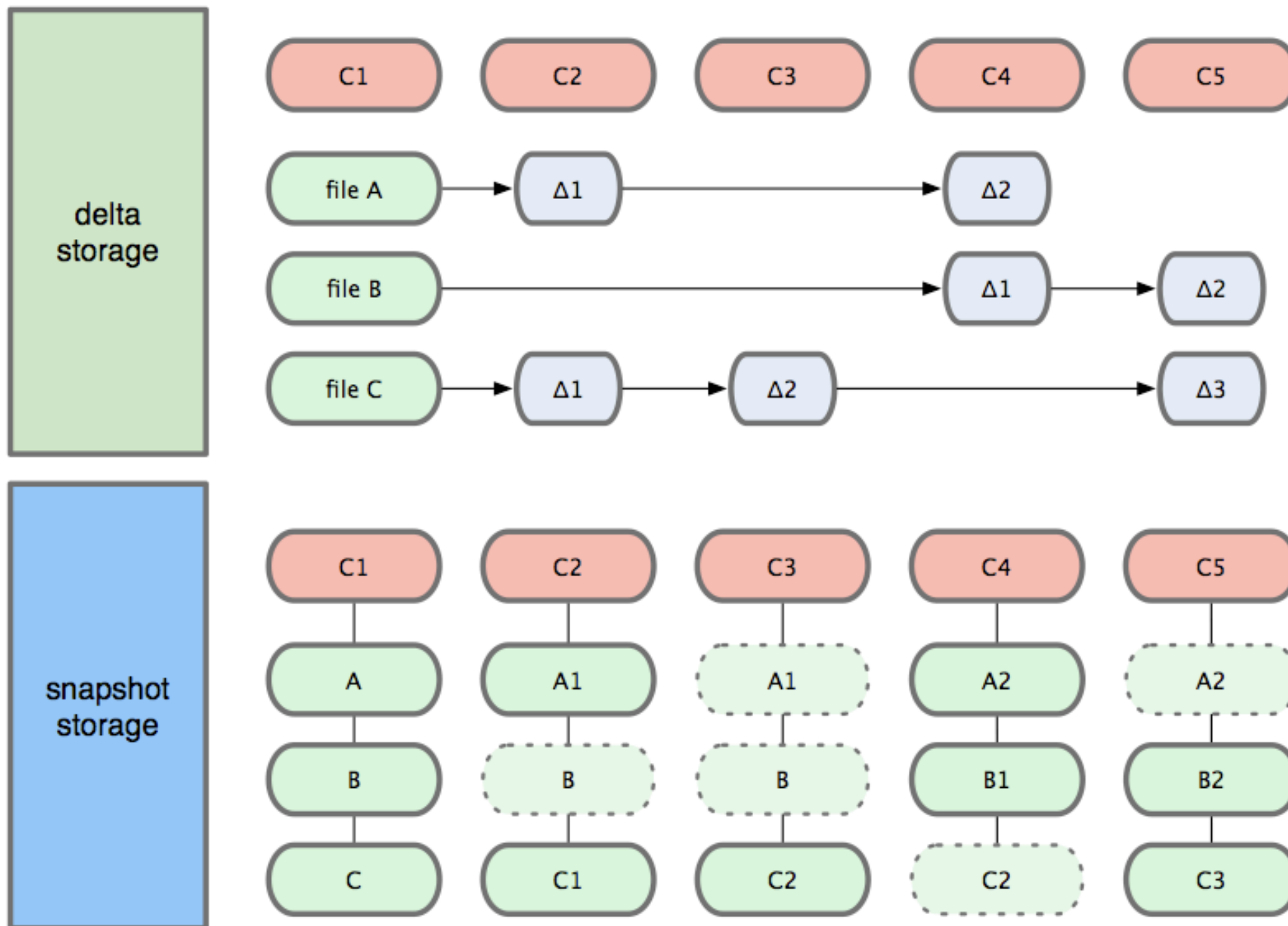
file contents

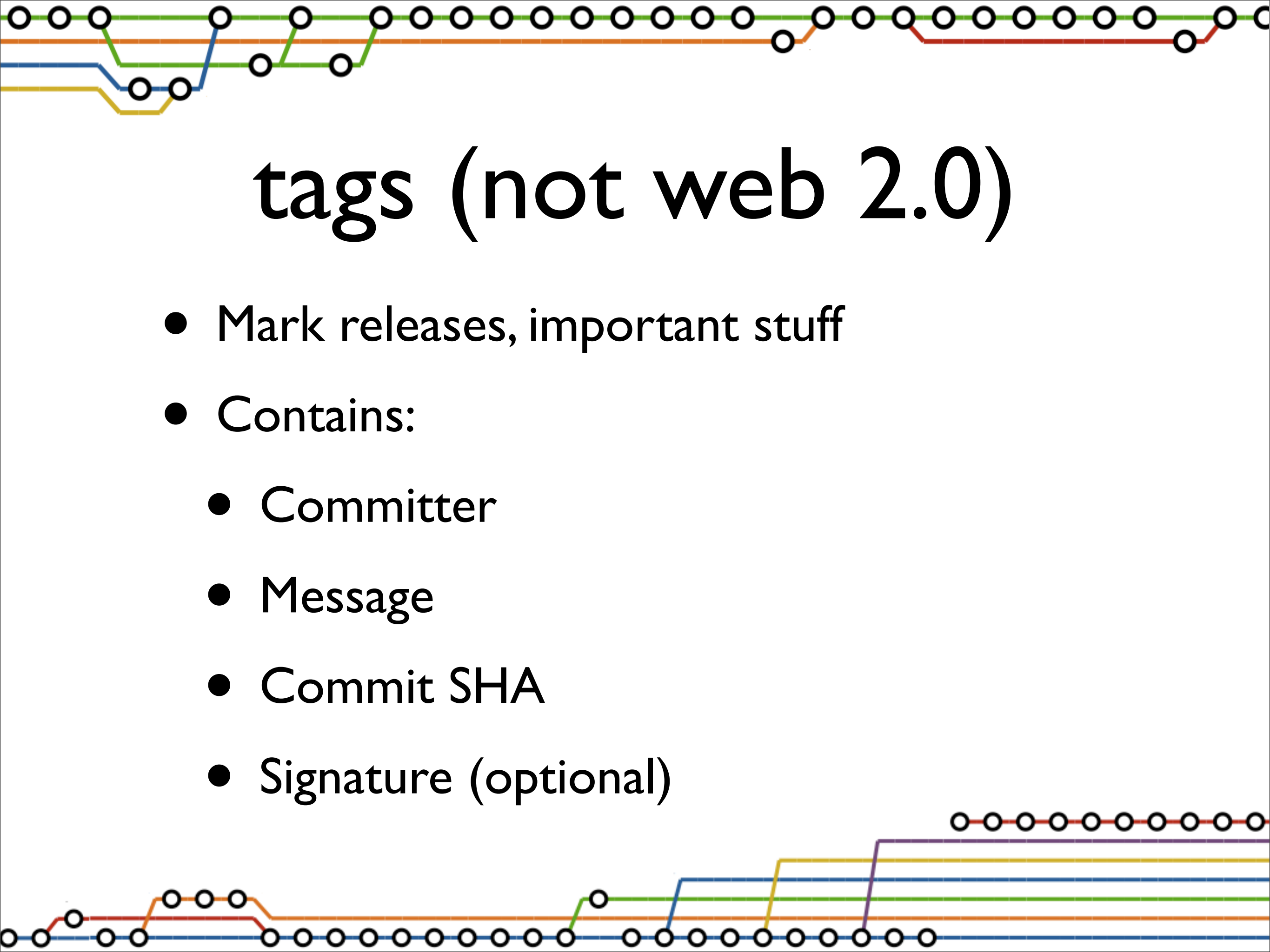
blob





# the big difference




The slide features a decorative header and footer. The header consists of several horizontal lines in green, orange, blue, and yellow, each with a series of small white circles. The footer is similar, with lines in blue, orange, green, yellow, and purple, also featuring white circles. The title 'tags (not web 2.0)' is centered in the middle of the slide in a large, black, sans-serif font.

# tags (not web 2.0)

- Mark releases, important stuff
- Contains:
  - Committer
  - Message
  - Commit SHA
  - Signature (optional)




# local commands

- Creating repositories
  - Viewing history
  - Performing a diff
  - Merging branches
  - Switching branches
- 



# actually using git

- porcelain vs. plumbing
  - don't be scared of your terminal
  - GUI support is not 100% there yet
  - yes, it works on Windows.
  - plenty of import/interop tools
- 





# workflows

- simple & centralized
- hardcore forking action

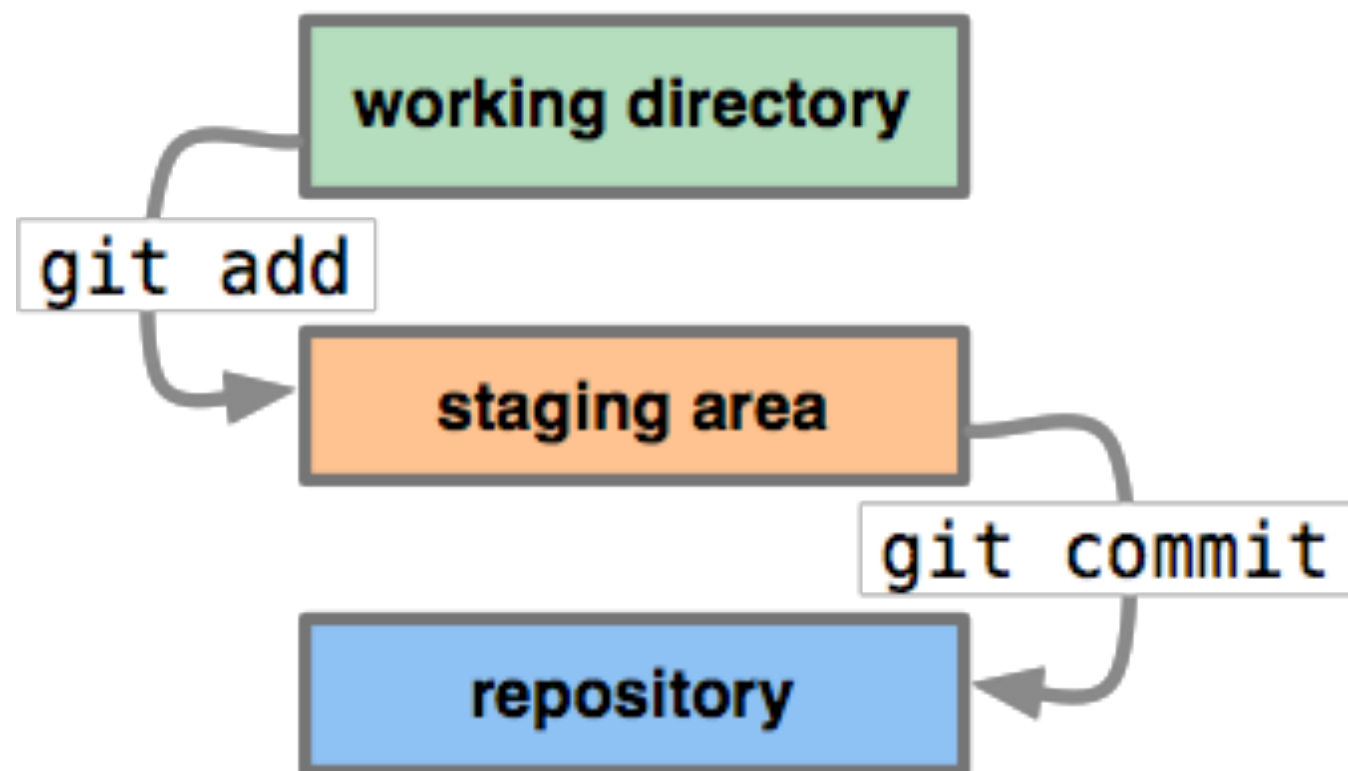


# simple & centralized

- basic workflow for most projects
- host your code at github, gitosis, etc



# the staging area





# create your project


```
$ rails toast2.0
```

```
[... blah blah blah ...]
```

```
$ cd toast2.0
```

```
$ git init
```

```
Initialized empty Git repository in /Users/qrush/Dev/toast2.0/.git/
```





# more setup

```
$ edit .gitignore
```

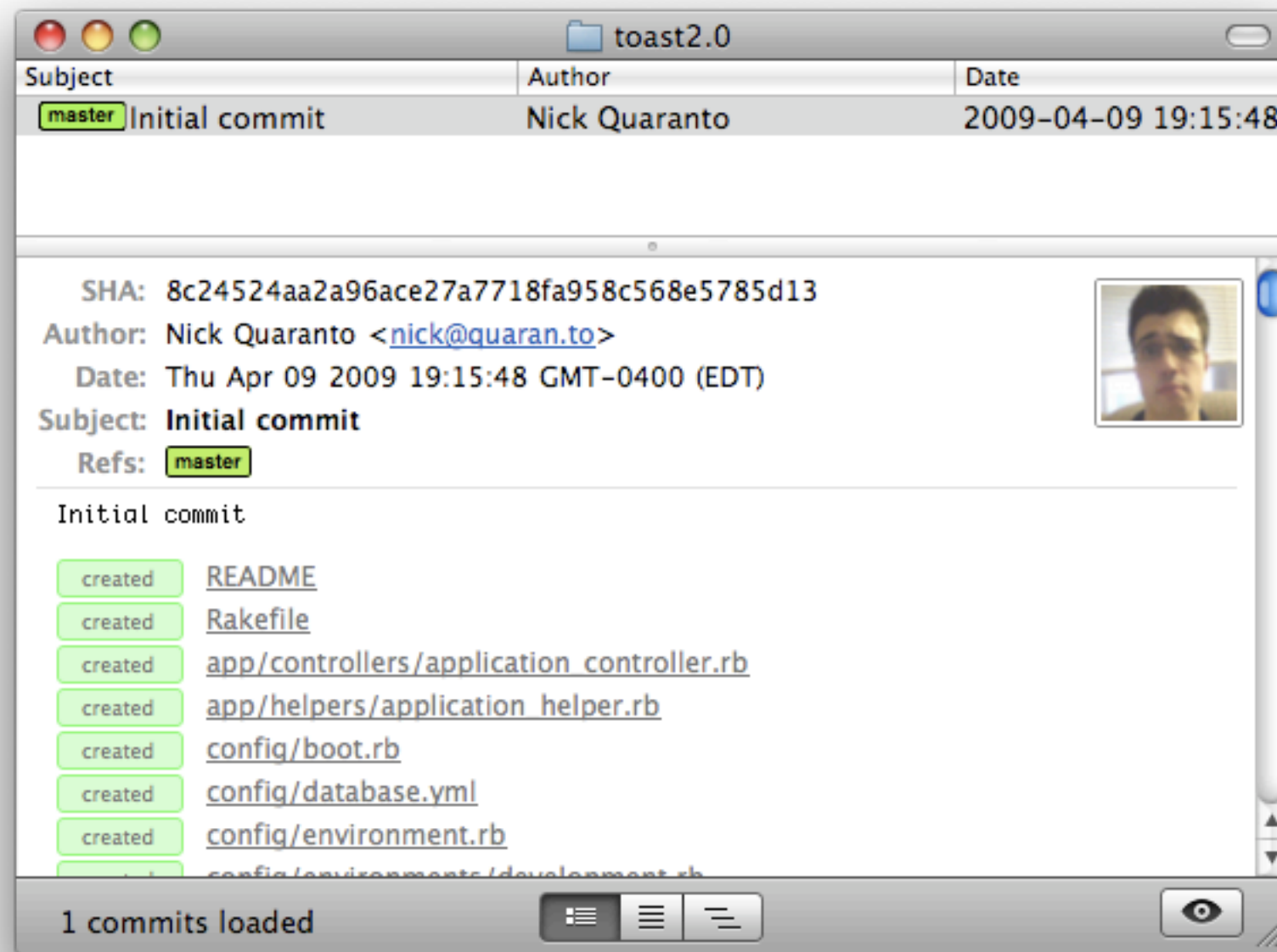
```
$ git add .
```

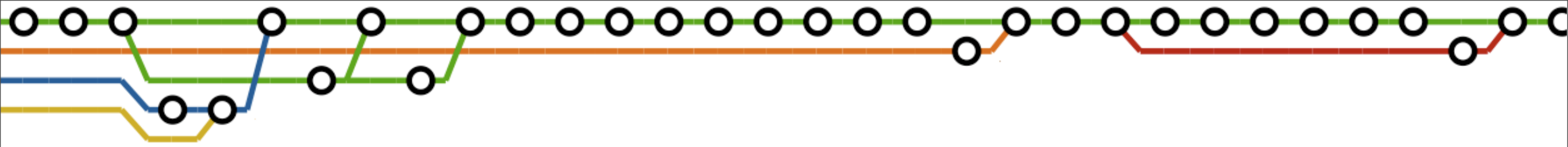
```
$ git commit -m "first commit"
```

```
[master (root-commit)]: created 8c24524: "Initial commit"  
41 files changed, 8452 insertions(+), 0 deletions(-)  
[.. files files files ..]
```



# DONE.





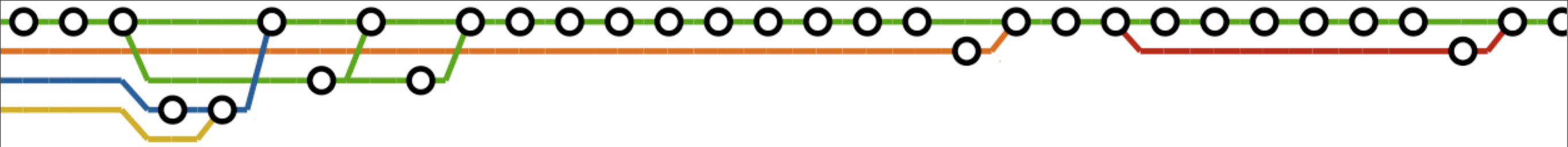
Ok, maybe not.

```
$ edit config/environment.rb
```

```
$ script/generate migration
```

```
AddPosts
```





```
$ git status
```

```
# On branch master
```

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

```
#   (use "git add <file>..." to update what will be committed)
```

```
#   (use "git checkout -- <file>..." to discard changes in working  
directory)
```

```
#
```

```
#   modified:   config/environment.rb
```

```
#
```

```
# Untracked files:
```

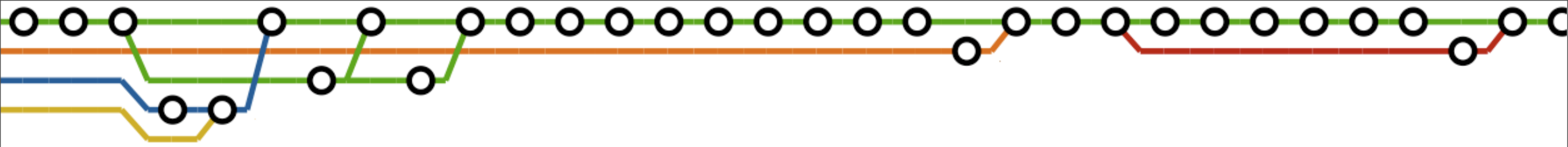
```
#   (use "git add <file>..." to include in what will be committed)
```

```
#
```

```
#   db/
```

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

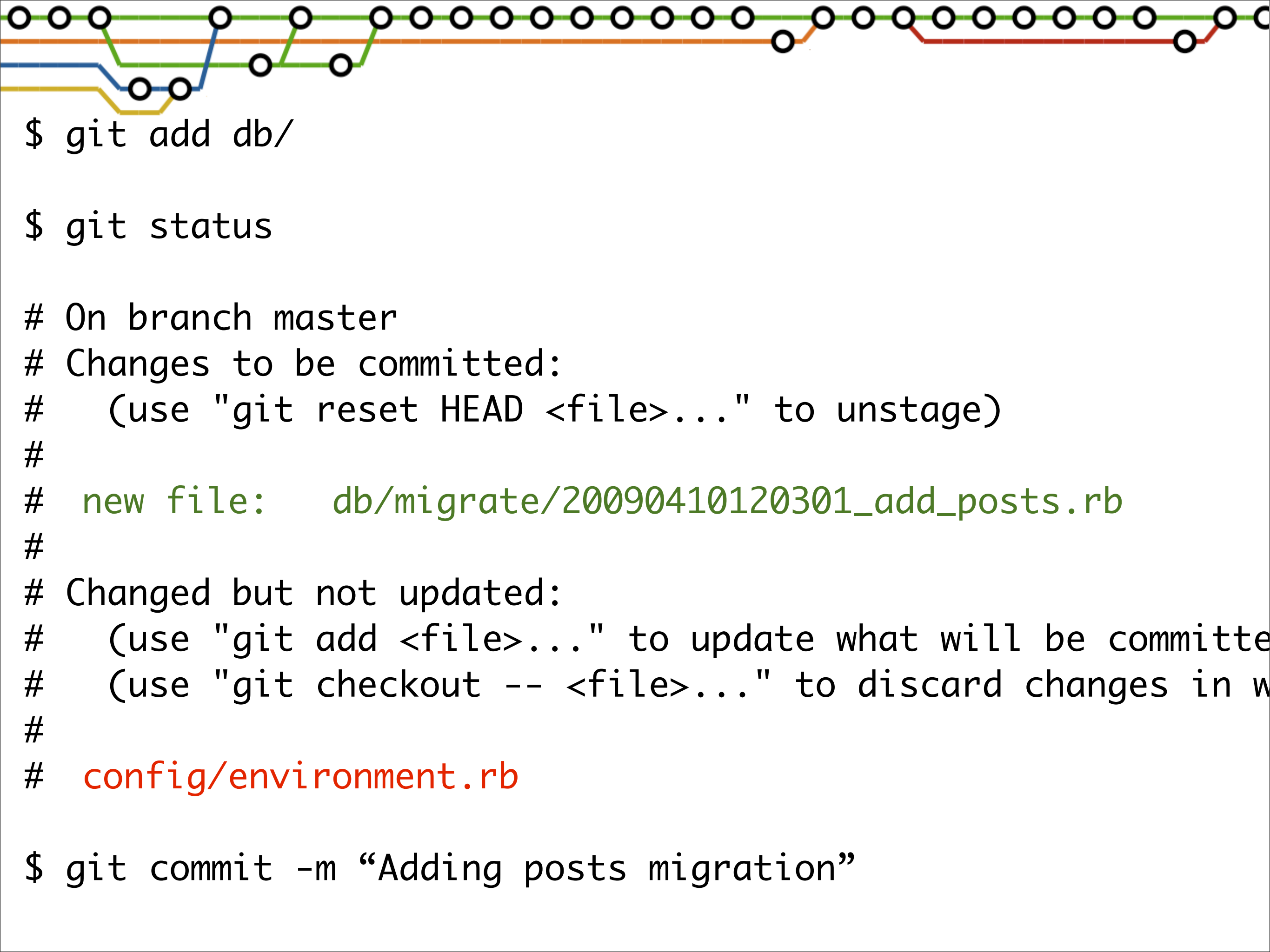




```
$ git diff
```

```
diff --git a/config/environment.rb b/config/environment.rb
index 631a3a3..dfc184b 100644
--- a/config/environment.rb
+++ b/config/environment.rb
@@ -15,10 +15,7 @@ Rails::Initializer.run do |config|
   # config.load_paths += %W( #{RAILS_ROOT}/extras )

   # Specify gems that this application depends on
-  # config.gem "bj"
-  # config.gem "hpricot", :version => '0.6', :source => "ht
-  # config.gem "sqlite3-ruby", :lib => "sqlite3"
-  # config.gem "aws-s3", :lib => "aws/s3"
+  config.gem "thoughtbot-factory_girl", :lib => "factory_gi
```



\$ git add db/

\$ git status

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# new file: db/migrate/20090410120301\_add\_posts.rb

#

# Changed but not updated:

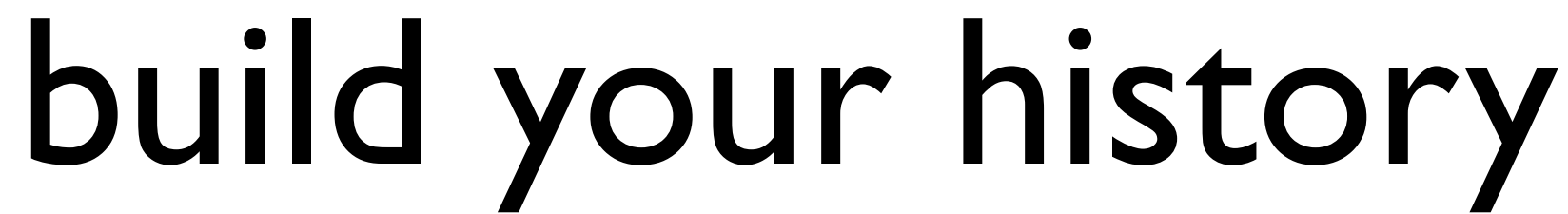
# (use "git add <file>..." to update what will be committed

# (use "git checkout -- <file>..." to discard changes in working directory

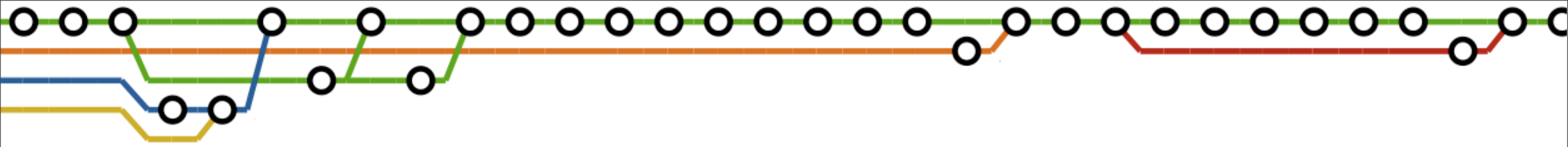
#

# config/environment.rb

\$ git commit -m "Adding posts migration"



- **master** origin/master Adding posts migration
- Alright, maybe not
- Adding a new route
- Fixing that bug
- Adding some stuff
- Initial commit



# the grind

Hack away, fix bugs

`vim / mate / etc`

Stage changes

`git add`

Review changes

`git status/diff`

Store changes

`git commit`



## temporarily store changes

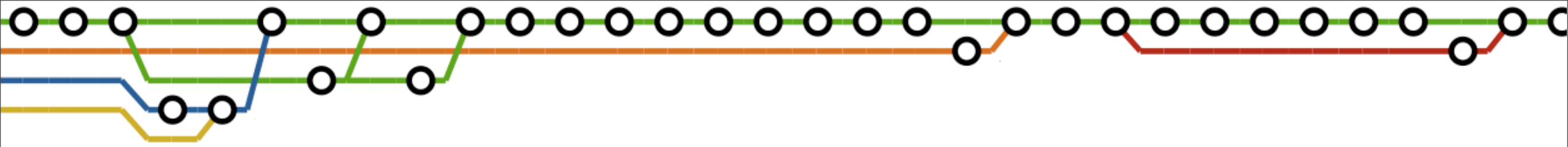
## restore to last commit

## unstage file

## unstage & restore file

# restore everything

## undo a changeset



# hardcore forking action

- Fork means another repo
- Multiple repos means multiple branches

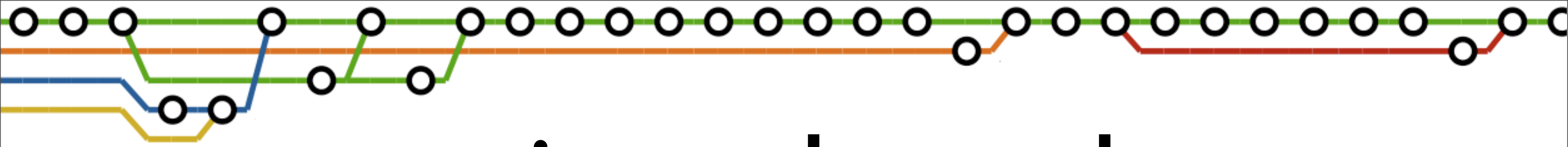




# branches



- Cheap and local
- Easy to switch
- Try out new ideas
- Merging is way smarter



# using a branch

`git checkout -b feature2`

create new branch

`git commit`

save some work

`git checkout master`

switch back

`git merge feature2`

work is merged in

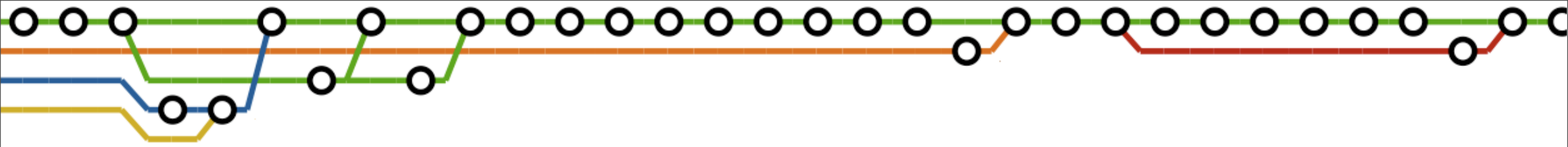
`git rebase feature2`

work played on top

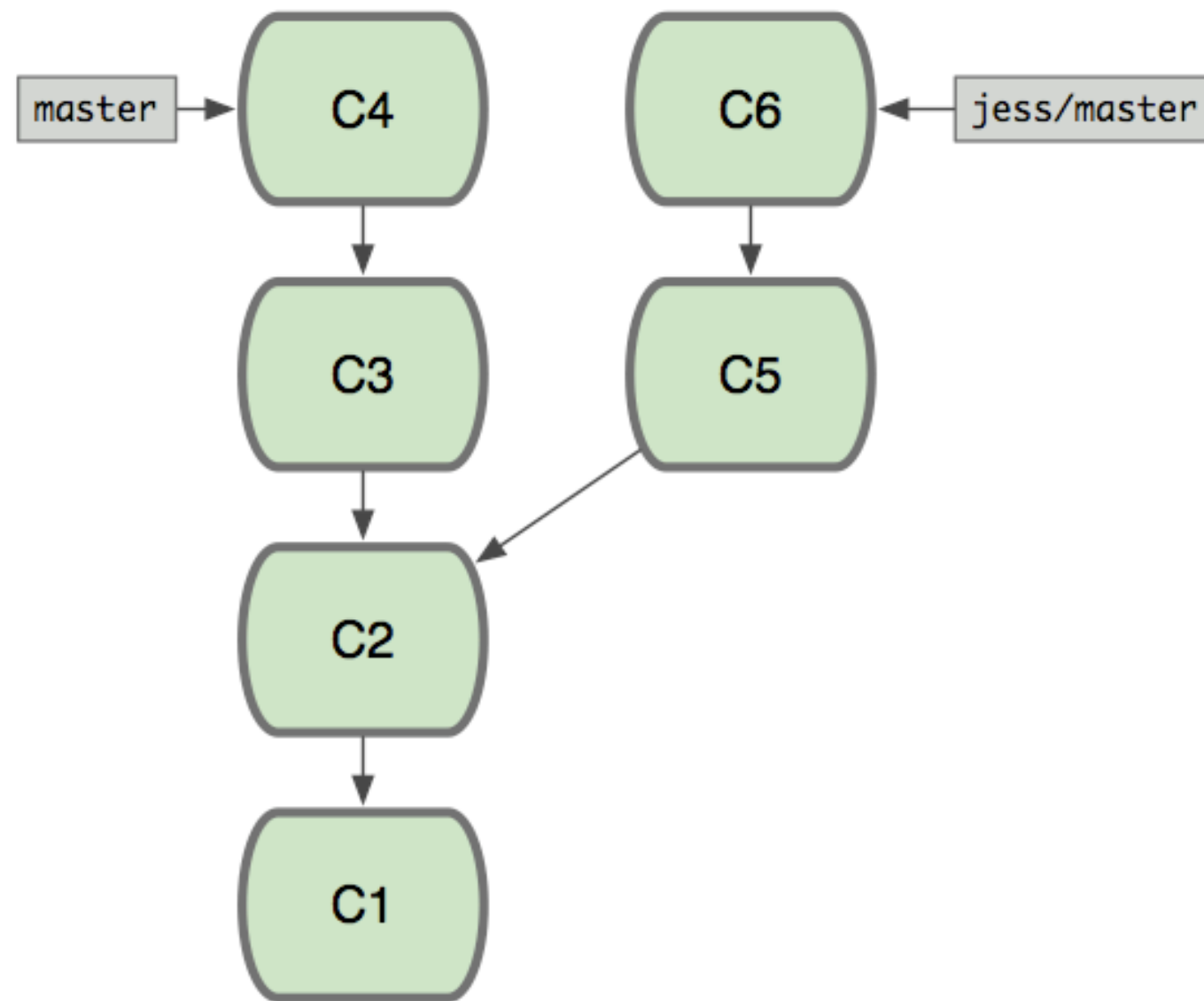
`git branch -d feature2`

delete branch



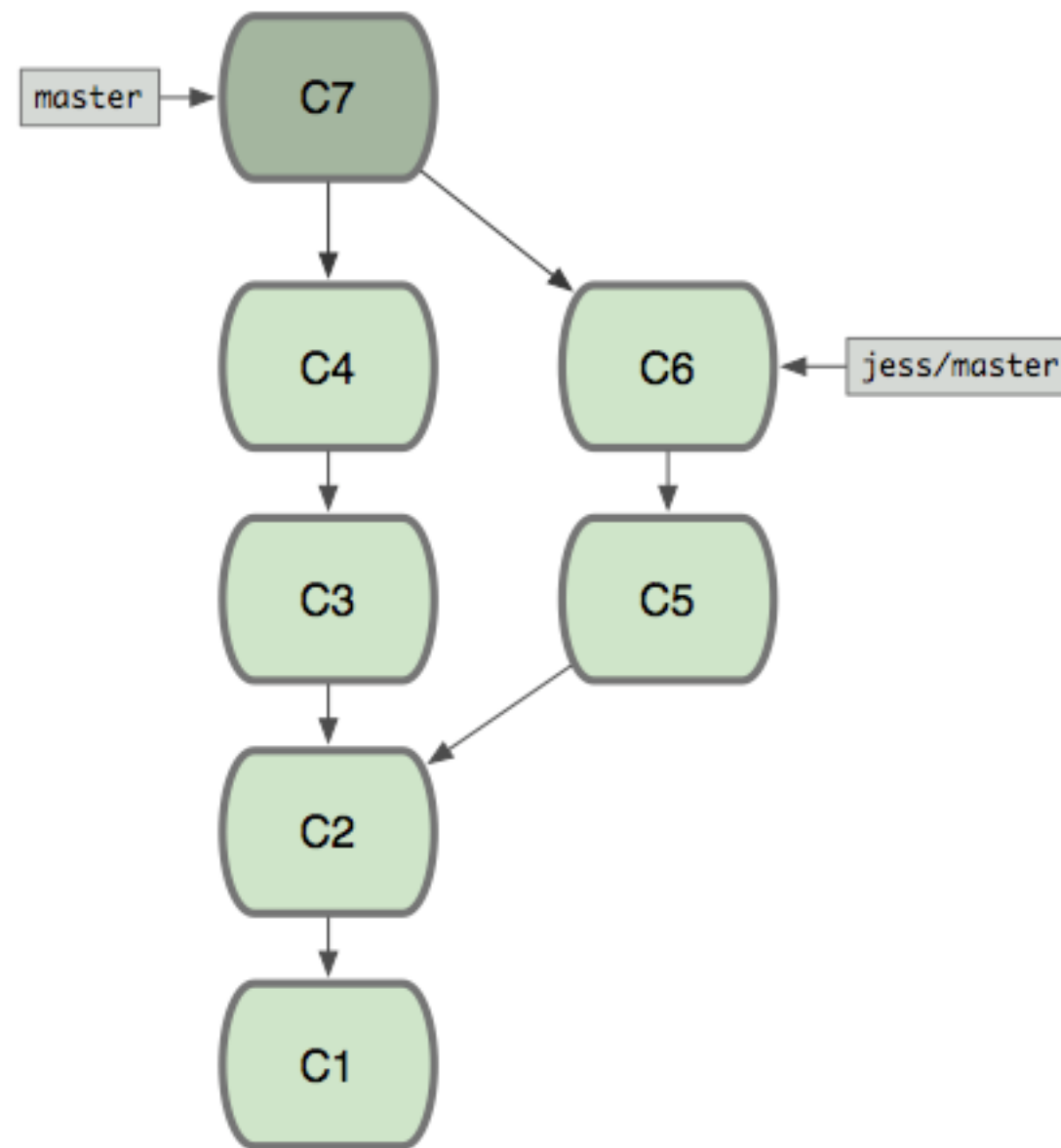


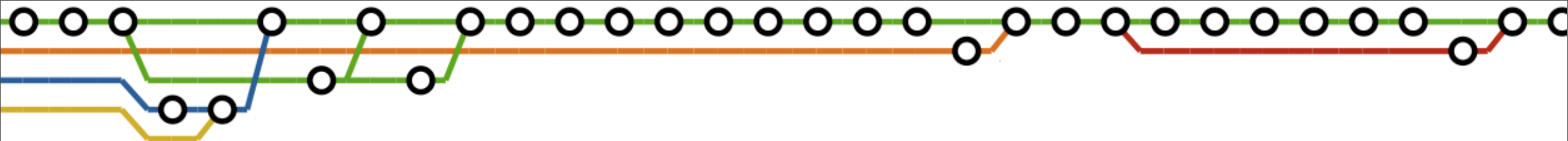
# merging (before)



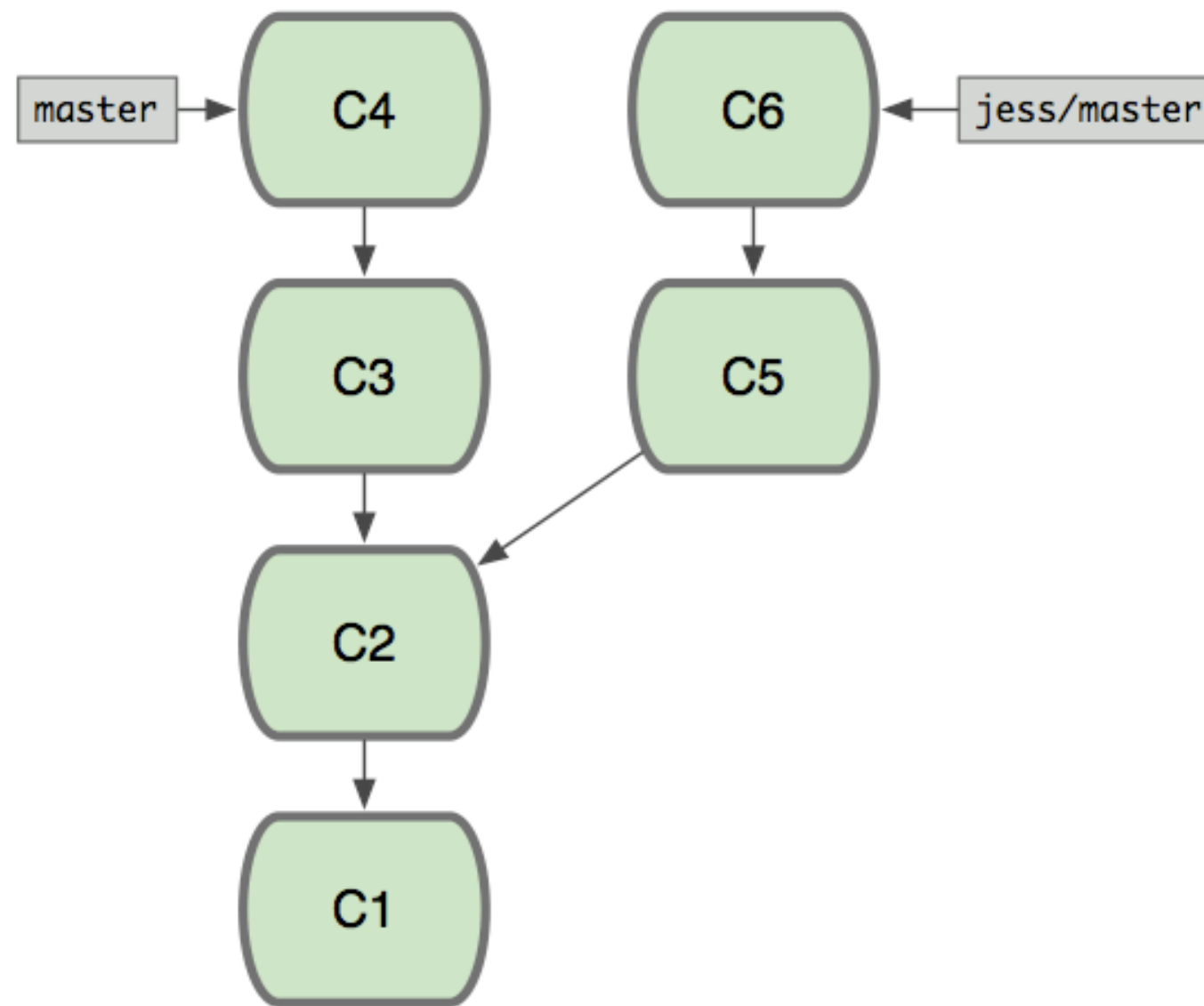


# merging (after)



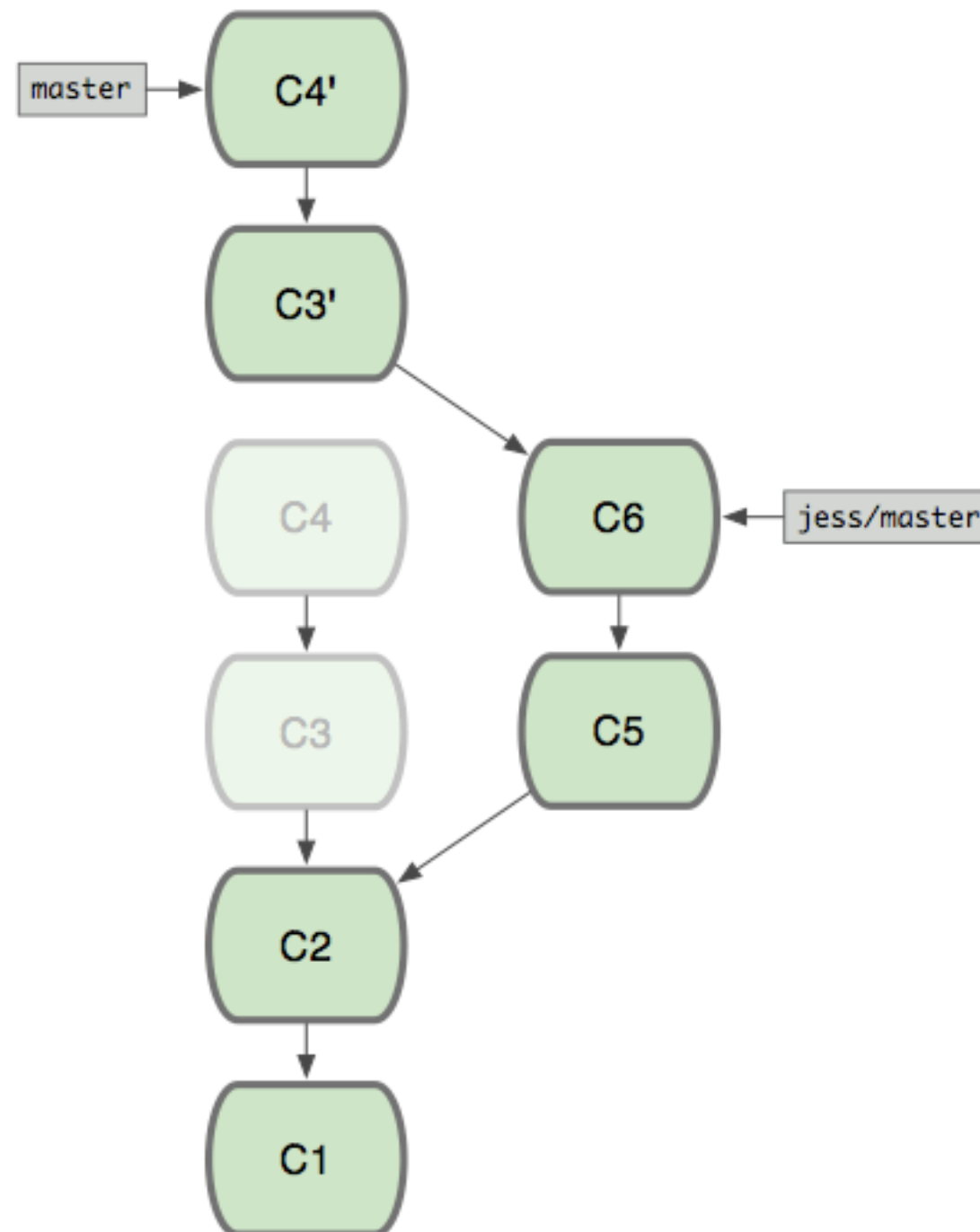


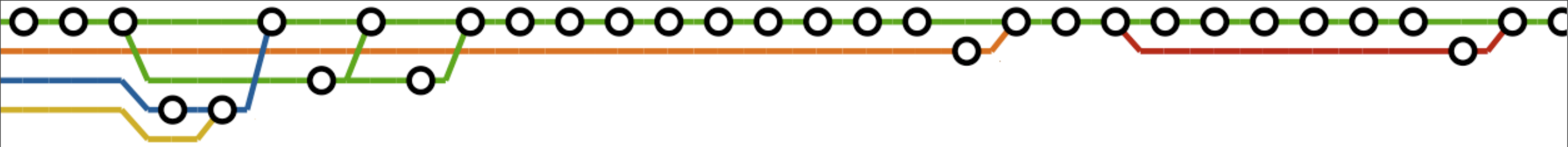
# rebasing (before)



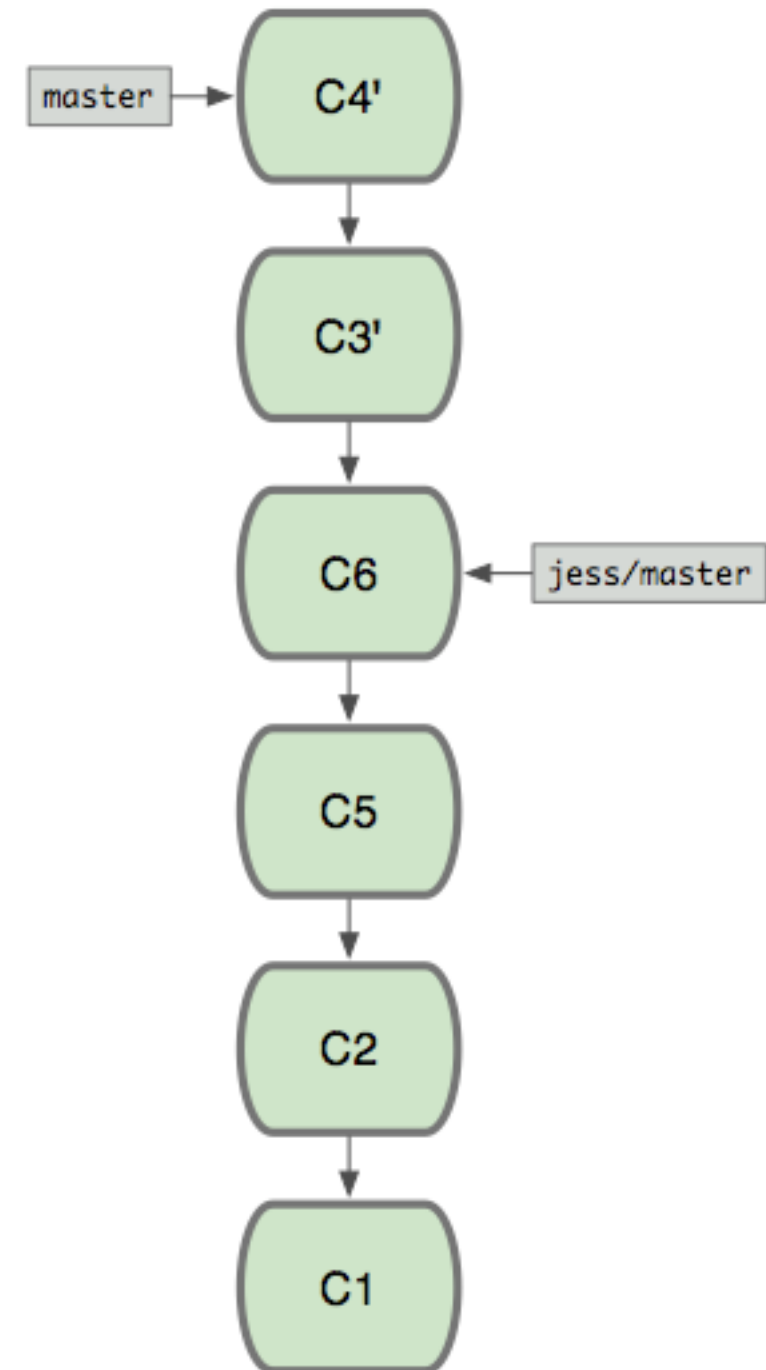
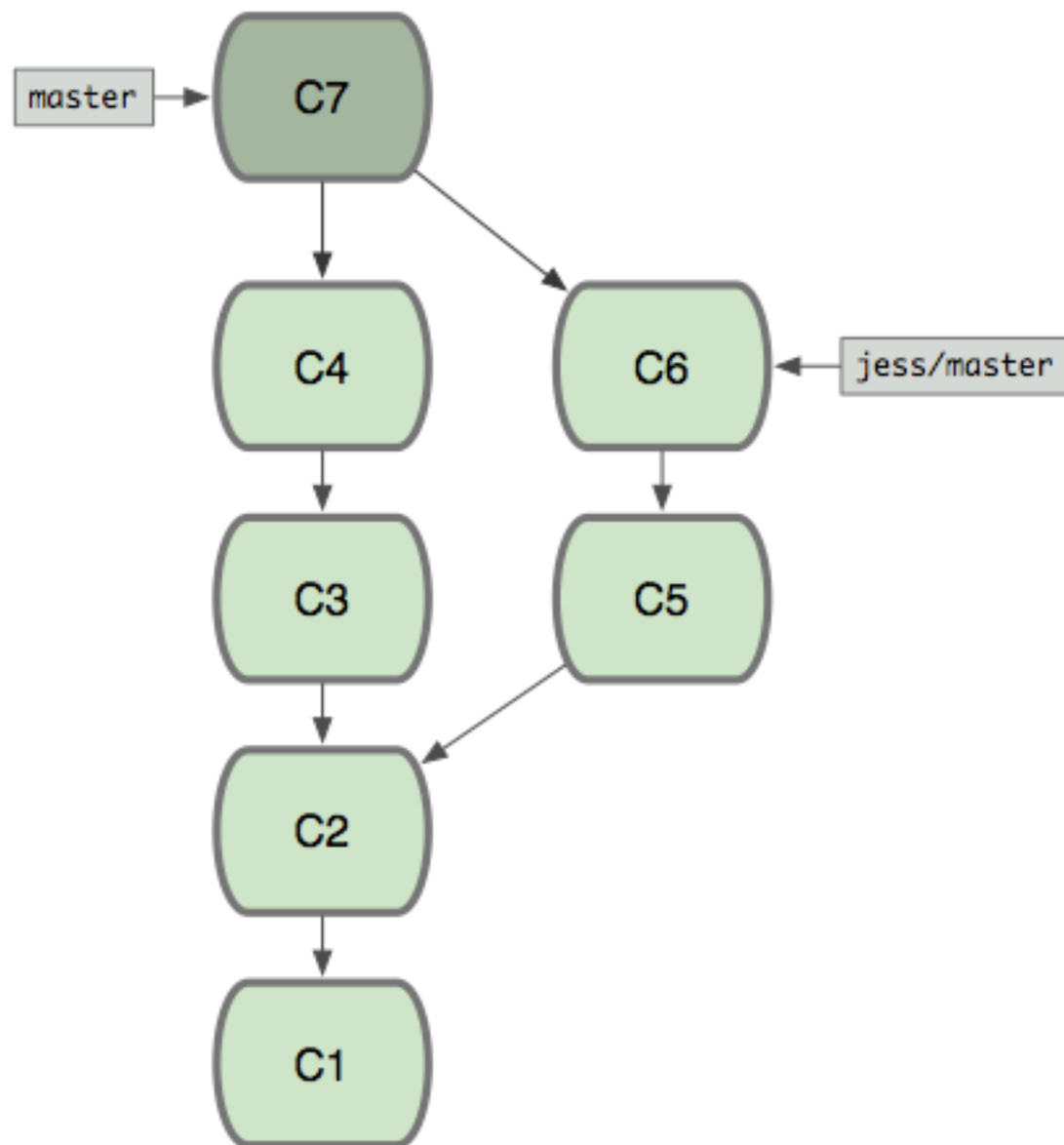


# rebasing (after)





# merge vs. rebase

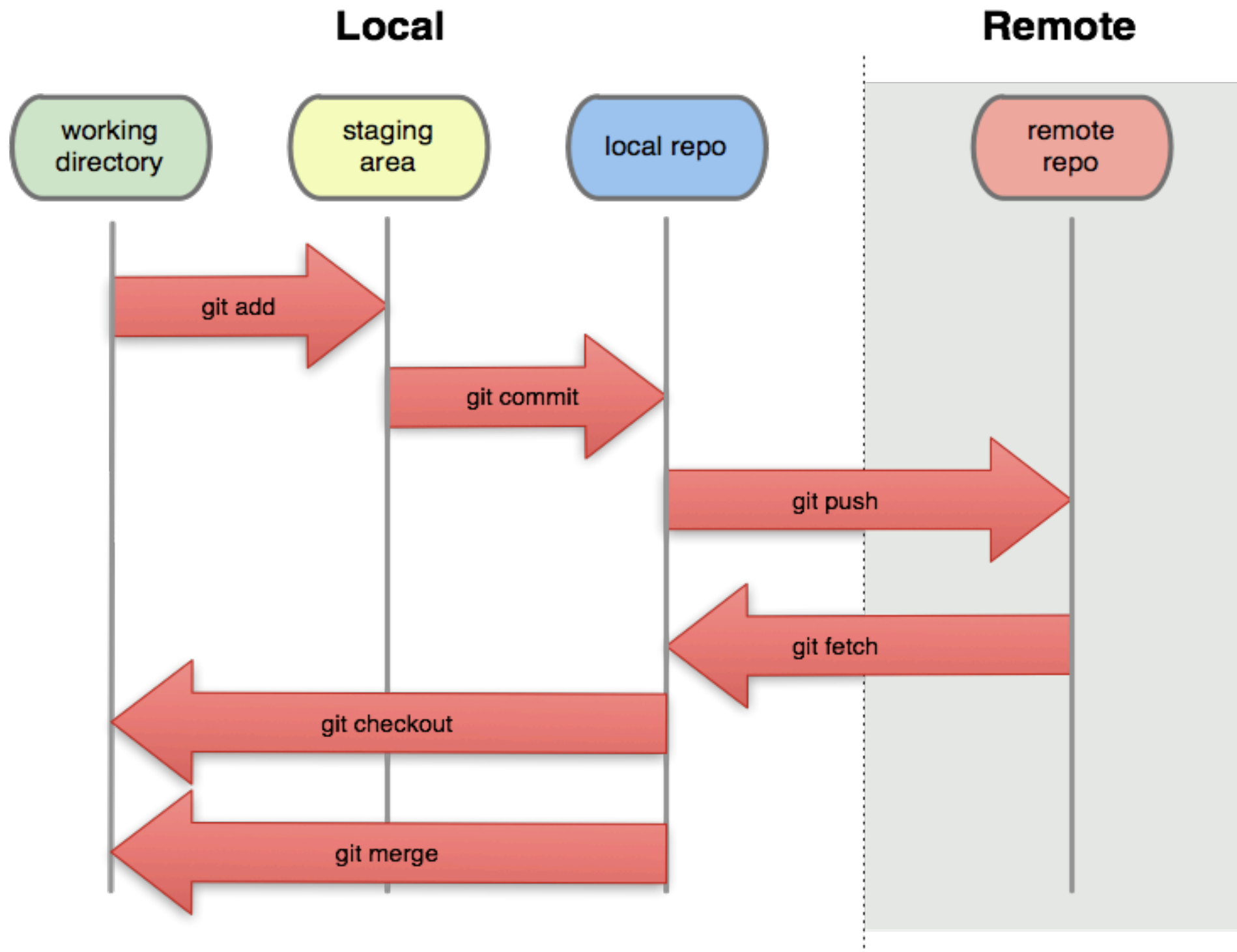




# warning!

- Rebasing is rewriting history
- BAD for pushed commits
- Keep the repo in fast-forward
- (This doesn't mean rebase is bad!)

# syncing up





# push away

```
$ git remote add origin git@github.com:qrush/toast2.0.git
```

```
$ git push origin master
```

```
Counting objects: 78, done.
```

```
Compressing objects: 100% (71/71), done.
```

```
Writing objects: 100% (78/78), 80.49 KiB, done.
```

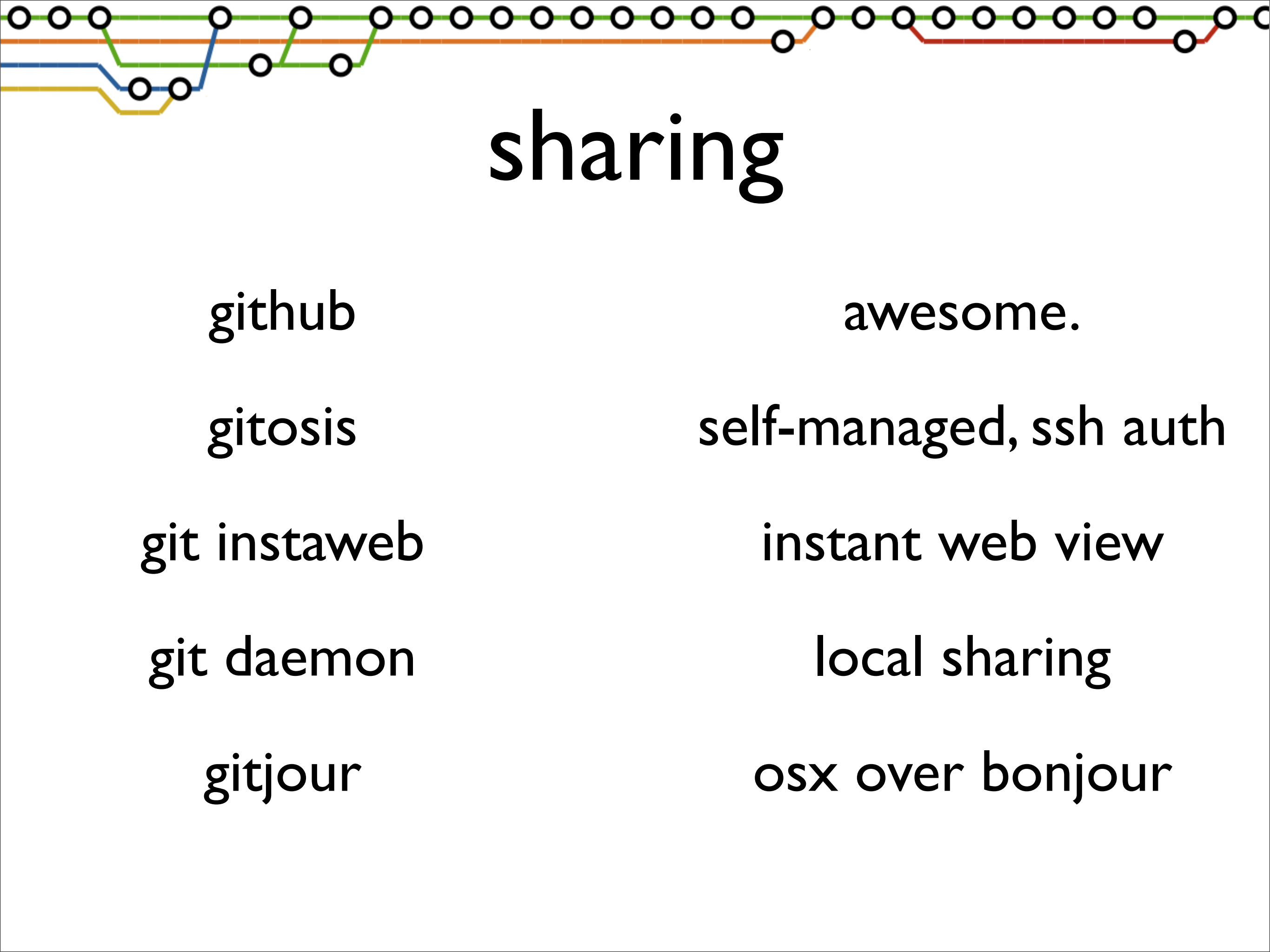
```
Total 78 (delta 21), reused 0 (delta 0)
```

```
To git@github.com:qrush/toast2.0.git
```

```
* [new branch]      master -> master
```







# sharing

github

awesome.

gitosis

self-managed, ssh auth

git instaweb

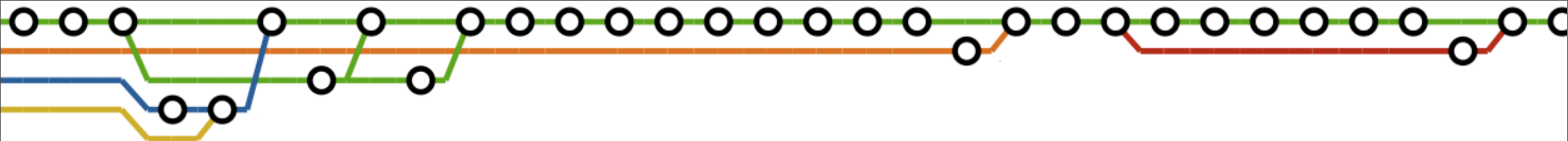
instant web view

git daemon

local sharing

gitjour

osx over bonjour



# bringing down code

git fetch **remote**: get updates

git pull **remote** **branch**:

- get updates from **remote** for **branch**
- merge/rebase it into your current branch



A diagram illustrating Git commit history. It features several horizontal lines in different colors (green, orange, blue, yellow, red, purple) representing different branches. Black circles on these lines represent commits. The diagram shows a complex branching structure with merges and pull requests. The title "basic pulling" is centered in the middle of the image.

# basic pulling

```
$ git remote add mojombo git://  
github.com/mojombo/jekyll.git
```

```
$ git pull mojombo master
```



# go fetch

```
$ git remote add mojombo git://  
github.com/mojombo/jekyll.git
```

```
$ git fetch mojombo
```


```
$ gitx
```

```
$ git merge mojombo/master
```






# looking at upstream

- `mojombo/master` `v0.5.0` Regenerated gemspec...
  - Version bump to 0.5.0
  - update history with 0.5.0 release
  - exit from rakefile if wrong version of je...
  - Updated to use jeweler 0.11.0, which let...
  - `master` `origin/HEAD` `origin/master` Removing test...
  - Reset posts, layouts, and categories bef...
  - Regenerating gemspec so rake build is h...
- 



# after merge

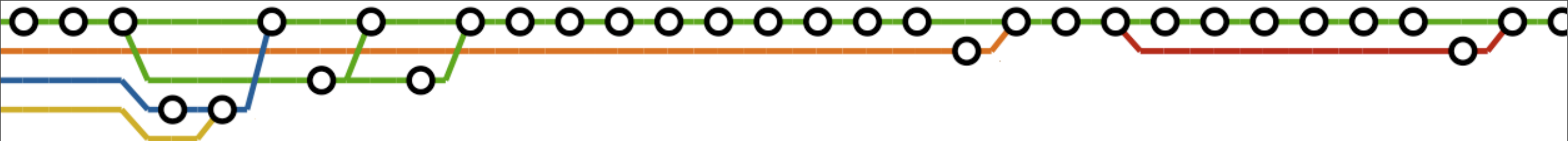
- **master** **mojombo/master** **v0.5.0** Regenerated ge...
  - Version bump to 0.5.0
  - update history with 0.5.0 release
  - exit from rakefile if wrong version of je...
  - Updated to use jeweler 0.11.0, which let...
  - **origin/HEAD** **origin/master** Removing test/dest ...
  - Reset posts, layouts, and categories bef...
  - Regenerating gemspec so rake build is h...
- 



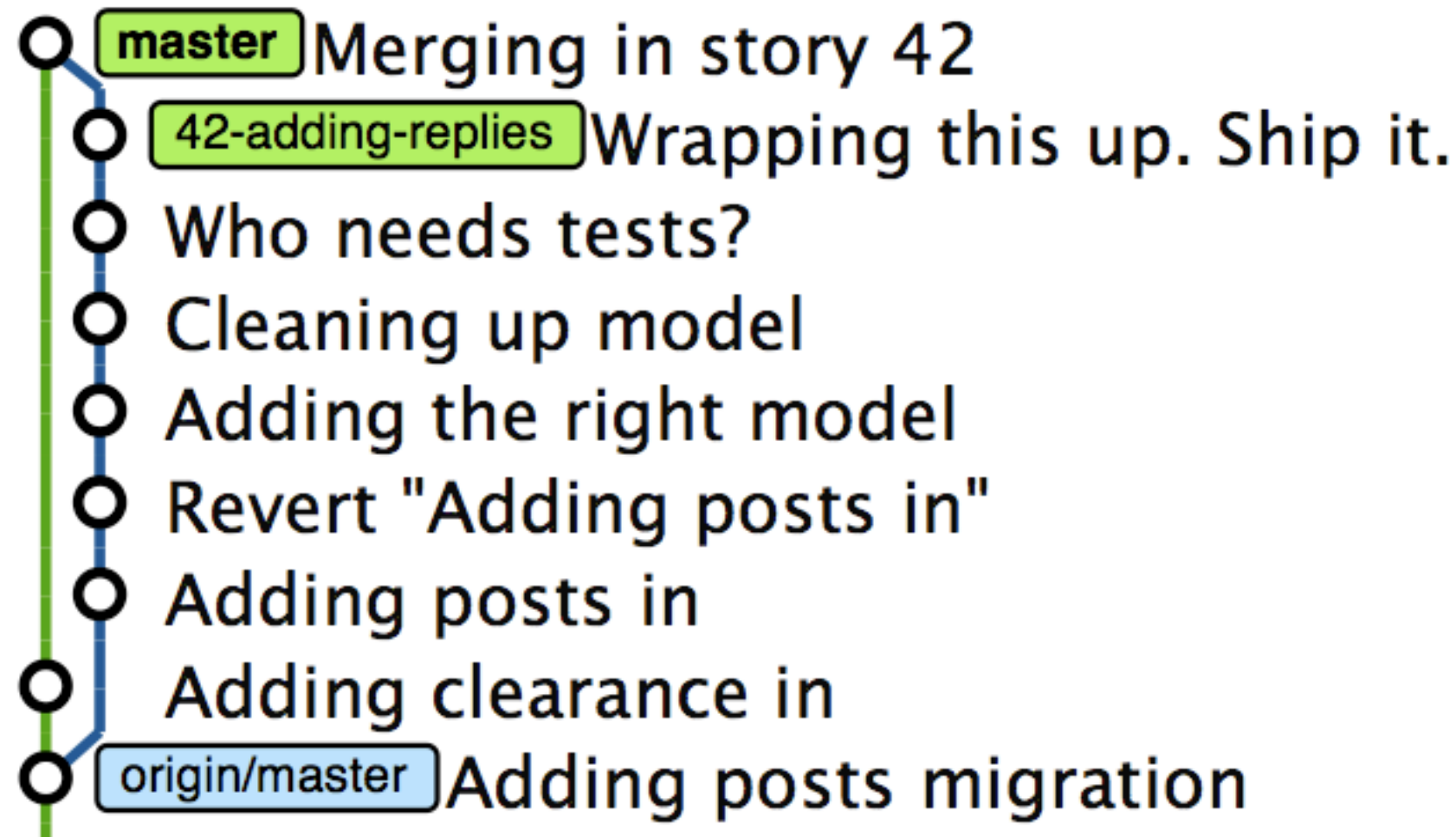


# topic branch

- **master** Adding clearance in
  - **42-adding-replies** Wrapping this up. Ship it.
  - Who needs tests?
  - Cleaning up model
  - Adding the right model
  - Revert "Adding posts in"
  - Adding posts in
- **origin/master** Adding posts migration
  - Alright, maybe not
  - Adding a new route
  - Fixing that bug
  - Adding some stuff
  - Initial commit



with a merge




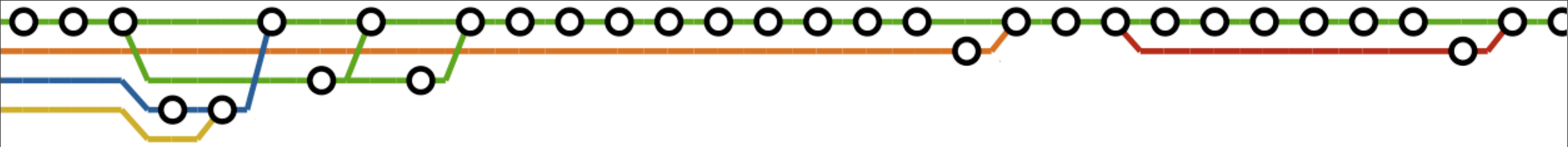




# Interactive Awesome.

`git rebase -i`

- Reordering commits
  - Splitting up changesets
  - Editing commits
  - Dropping them completely
  - Squashing multiple commits into one
- 



```
$ git rebase -i HEAD~6
```

```
pick a4d0f79 Adding posts in
pick 7e71afd Revert "Adding posts in"
pick 5e815ec Adding the right model
pick 956f4ce Cleaning up model
pick 6c6cdb4 Who needs tests?
pick c3481fd Wrapping this up. Ship it.
```

```
# Rebase bd0ceed..c3481fd onto bd0ceed
```

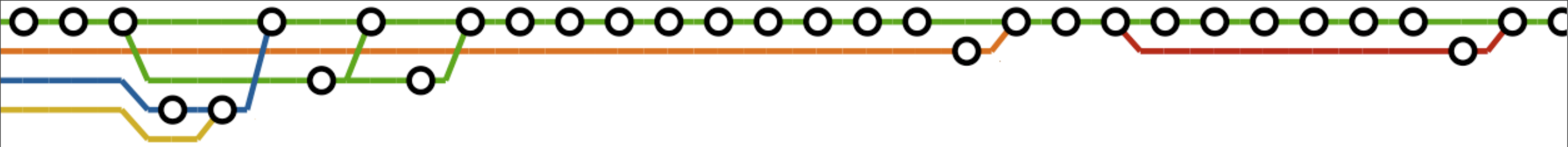
```
#
```

```
# Commands:
```

```
# p, pick = use commit
```

```
# e, edit = use commit, but stop for amending
```

```
# s, squash = use commit, but meld into previous commit
```



```
$ git rebase -i HEAD~6
```

```
pick a4d0f79 Adding posts in
squash 7e71afd Revert "Adding posts in"
squash 5e815ec Adding the right model
squash 956f4ce Cleaning up model
squash 6c6cdb4 Who needs tests?
squash c3481fd Wrapping this up. Ship it.
```

```
# Rebase bd0ceed..c3481fd onto bd0ceed
```

```
#
```

```
# Commands:
```

```
# p, pick = use commit
```

```
# e, edit = use commit, but stop for amending
```

```
# s, squash = use commit, but meld into previous commit
```



# squashed

- **42-adding-replies** Finishing up replies story
- **master** Adding clearance in
- **origin/master** Adding posts migration
- Alright, maybe not
- Adding a new route
- Fixing that bug
- Adding some stuff
- Initial commit


The slide features a decorative border at the top and bottom. The top border consists of a series of black circles connected by horizontal lines in green, orange, blue, and yellow. The bottom border is similar, with additional purple and blue lines. The central text 'learn more' is in a large, black, sans-serif font.

# learn more

- <http://book.git-scm.com>
- <http://gitready.com>
- <http://learn.github.com>
- <http://gitcasts.com>
- git internals peepcode

A series of horizontal lines in green, orange, blue, and yellow, each with small white circles at regular intervals. Some lines have small steps or dips.

# thanks

- kudos to:
    - Scott Chacon for awesome presentations
    - Charles Duan for his great git tutorial
    - Ben Hughes for bugging me to try git
    - You for listening/reading
- 
- A series of horizontal lines in blue, orange, green, yellow, and purple, each with small white circles at regular intervals. Some lines have small steps or dips.