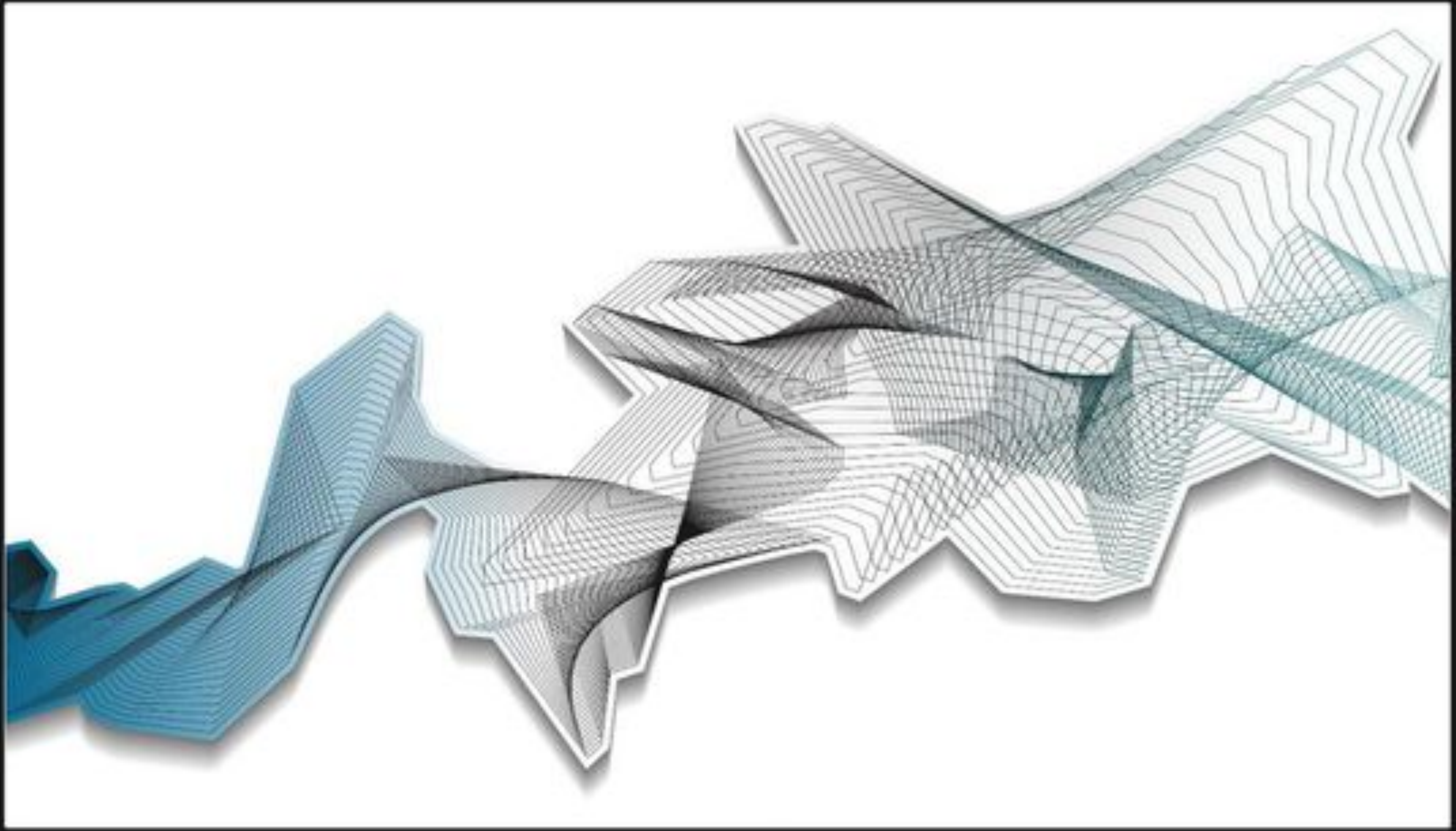
git at scale

brett koonce
asparagui
2017-02-23

overview

- who uses homebrew, git, open source?
- purpose: convince you to make a contribution to an open source project
- how to make a good pr, what a maintainer does
- xcode + git, project/release management



VERSION CONTROL

git gets easier once you get the basic idea that branches are homeomorphic endofunctors mapping submanifolds of a Hilbert space.

version control

- why use it? what is the purpose?
- project history, save progress
- concentrate on a subset of codebase
- reason about changes more simply
- allows us to collaborate with others, future self

st. linus quote

- what he thinks a good pr looks like
- https://github.com/torvalds/subsurface-for-dirk/blob/master/README#L92

Also, please write good git commit messages. A good commit message looks like this:

```
    Header line: explain the commit in one line (use the imperative)

    Body of commit message is a few lines of text, explaining things
    in more detail, possibly giving some background about the issue
    being fixed, etc etc.

    The body of the commit message can be several paragraphs, and
    please do proper word-wrap and keep columns shorter than about
    74 characters or so. That way "git log" will show things
    nicely even when it's indented.

    Make sure you explain your solution and why you're doing what you're
    doing, as opposed to describing what you're doing. Reviewers and your
    future self can read the patch, but might not understand why a
    particular solution was implemented.

    Reported-by: whoever-reported-it
    Signed-off-by: Your Name <youremail@yourhost.com>
```

where that header line really should be meaningful, and really should be just one line. That header line is what is shown by tools like gitk and shortlog, and should summarize the change in one readable line of text, independently of the longer explanation. Please use verbs in the imperative in the commit message, as in "Fix bug that...", "Add file/feature ...", or "Make Subsurface..."

what i like

- single tree with release tags
- clean descriptions, common keywords
- lots of little commits and branches
- rapid merges and releases
- my config: git + textmate + gitx + github

how to make a pr

- basic demo of pr with github + hb + package

WRITES ALL THE DNA CODE FOR ALL SPECIES
DOESN'T WRITE A SINGLE COMMENT
quickmeme.com

maintenance

you’re gonna forget

- why was this bug a big deal?
- what hardware did you test it on?
- which other tools did we use along the way?
- what other stuff were we doing at the same time?
- coding a solution is step one, make it future proof
- a little time now will save you a bunch down the road

maintenance

- code cleanup and style
- document history of project
- gatekeeper for features, project releases
- grow project => need more contributors
- don’t scare the newbies!

git rebase

- moving commits around
- rewording descriptions
- fixing spelling mistakes
- fixing whitespace issues
- rebasing onto master
- merging/splitting a branch

release workflow

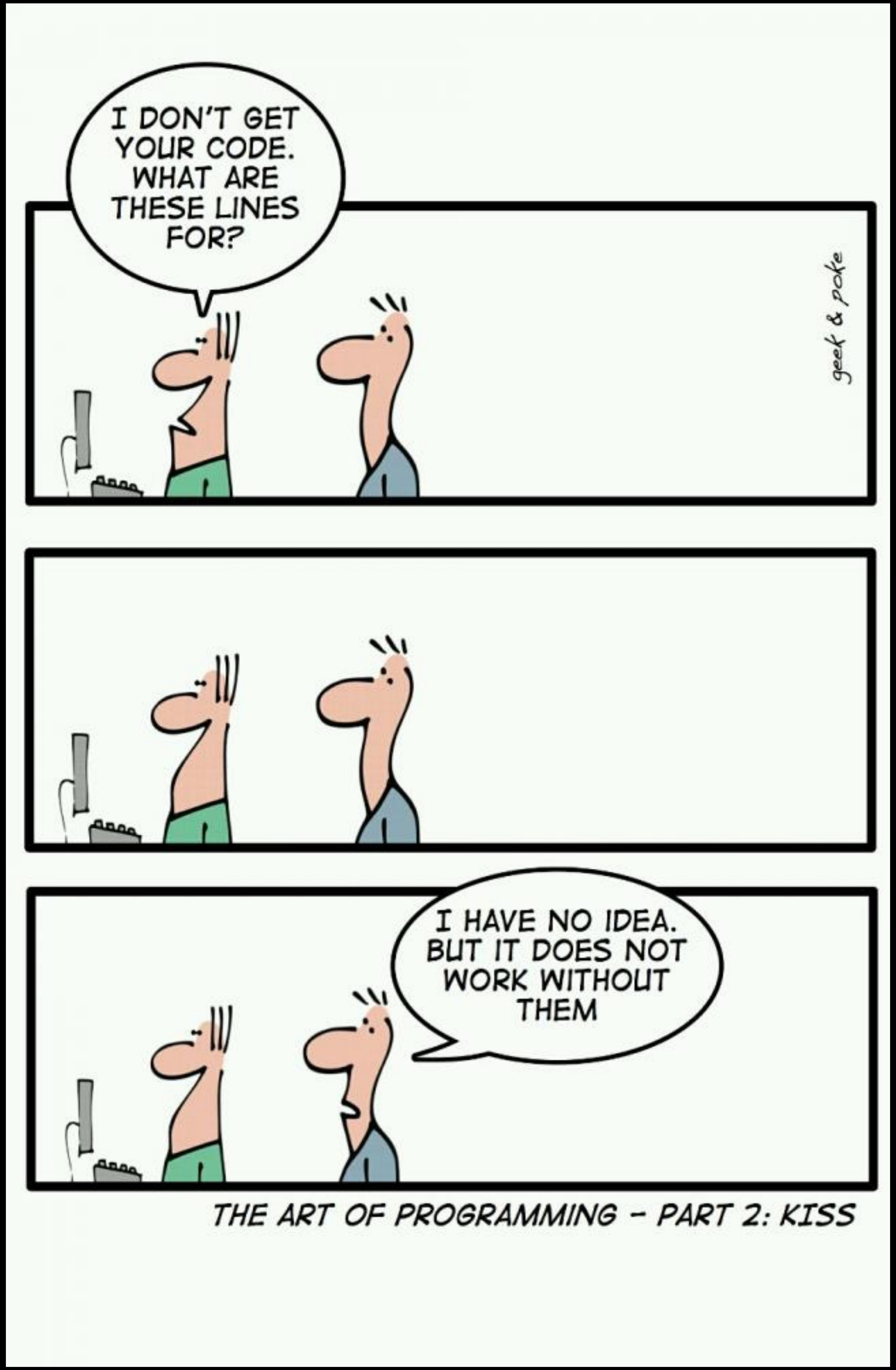
release workflow

- release current version, prep next version
- bump api/version number
- tag commit
- rebase release branch
- hand/fastlane/travis, upload first build
- daily release cycle

xcode + git

- xcode vs. git, choose a version
- things that will break on you
- project files, provisioning settings
- file imports, coredata files, plist files
- whitespace, soft reset, merging commits
- demo of fixing a project file

upgrading projects


I DON'T GET YOUR CODE. WHAT ARE THESE LINES FOR?
geek & poke
I HAVE NO IDEA. BUT IT DOES NOT WORK WITHOUT THEM
THE ART OF PROGRAMMING - PART 2: KISS

pod install

- pod install vs. pod upgrade
- only upgrade once each release cycle
- commit info/description
- checkin source, pin problem libraries
- static libraries/carthage vs online/gradle

coredata/realm models

- duck model of upgrading models
- what coredata does when we make a change
- how not to change project files if you want
- cowboy mode: how to go around xcode

Not this time Junior! Today is the release day.
Bug, bug there is a bug!
Project Manager
Junior Tester

releasing projects

readme.md

- add a license to a new repo
- markdown all the things
- document your build process
- give to somebody else, have them make build
- checkin keys, try to keep docs up to date

git grabbag

- unclean trees, how to clean up
- stupid head reset tricks
- git stash and pop
- .gitignore file
- branch naming schemes

git gui

- github, gitlab
- phabricator
- gitx, gitup
- sourcetree, github
- slack + integrations

Programmer
What my friends think I do
What my mom thinks I do
What society thinks I do
What my boss thinks I do
What I think I do
What I actually do

goto end

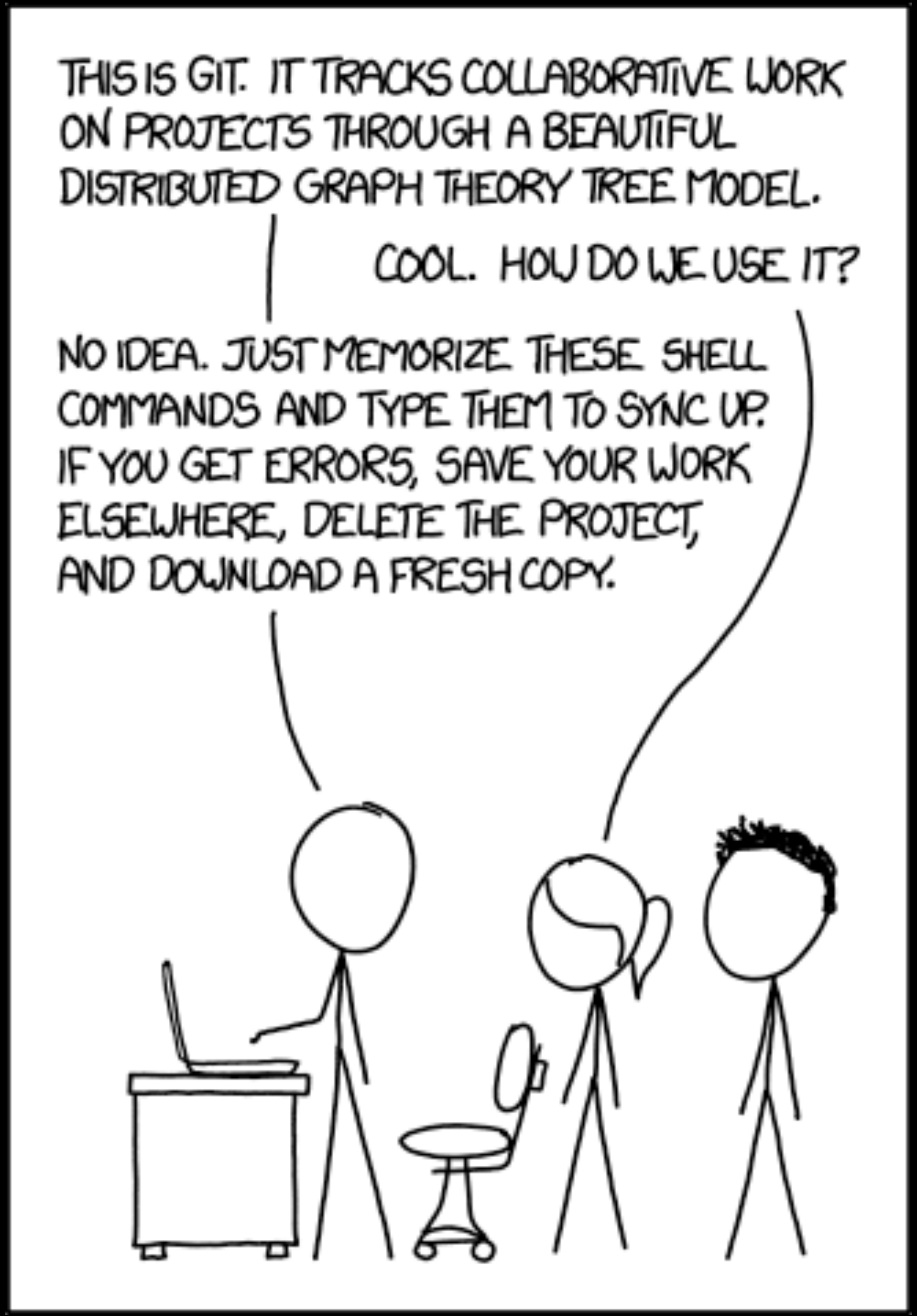
why do i code?

- code for self
- code for $$
- code for others
- show first commit to homebrew
- show own projects

your homework

- make a commit to something
- find a project you like/use and help it
- doesn’t have to be code
- docs, wiki, faq, documentation (or $$)
- hand out guides

golden rule of git


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

“if you’ve never force pushed
to master, you’re doing it wrong!”

-asparagui

sources

- brettkoonce.com/presentations/git.pdf
- mike’s book: git in practice (manning)
- git docs (e.g. stackoverflow)
- quarkworks.co

git at scale

brett koonce
asparagui
2017-02-23