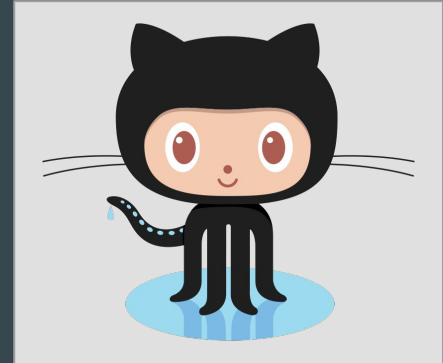# clone it, diff it, push it, pull it; branch it, merge it, git commit it

...

Why git is awesome and why we should all be using it

Sam Factor
GSPS 11/04/2016

Xkcd #1597

Questions to answer:

- What is git and how does it work?
- Why should I use it?
- What else can github do?
- How can I learn more?

What is git and how does it work?

Git is a version control software

- Records changes and tracks progress
- Allows for easy collaboration
- Local (git) and in the cloud (github, bitbucket or others)

No more naming files with dates or version numbers to make sure you keep a working version!
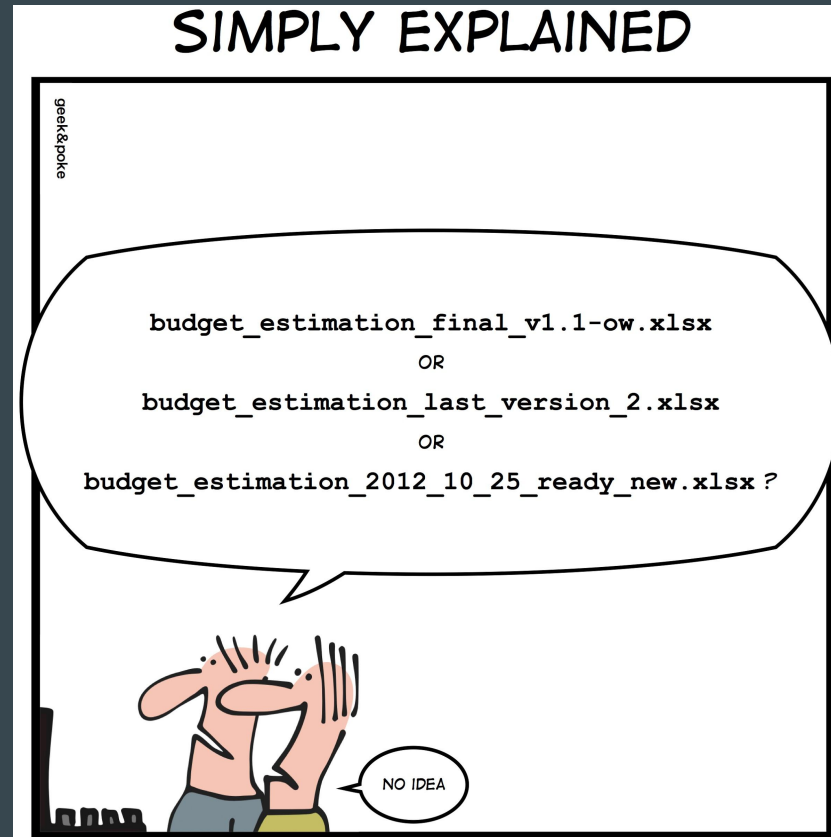
Why Should I Use git?

git is like your lab notebook for coding

git allows you to:

- "Go back in time" if you break your code
- Keep a working version of your code while you add features which may break it temporarily (branching, more on this later)

github is like your cv for coding

github (or other repository hosts) allows you to:

- Share or publish code for others to use
- Keep a backup in the cloud and keep code up to date on many different machines
- Collaborate without worrying about people editing a file at the same time

A simple git workflow

Create/download a git repository
  `git init`   `git clone <url>`
Modify/create files
Remind yourself what you changed:
  `git status`
Remind yourself how you changed it
  `git diff <filename>`
Stage files for a commit
  `git add <filename>`
Commit!
  `git commit -m '<commit message>'`
  `git commit` (this opens a text editor for longer messages)
Sync with remote repository (github)
  `git pull`
(Fix any merge conflicts that may occur)
  `git push`
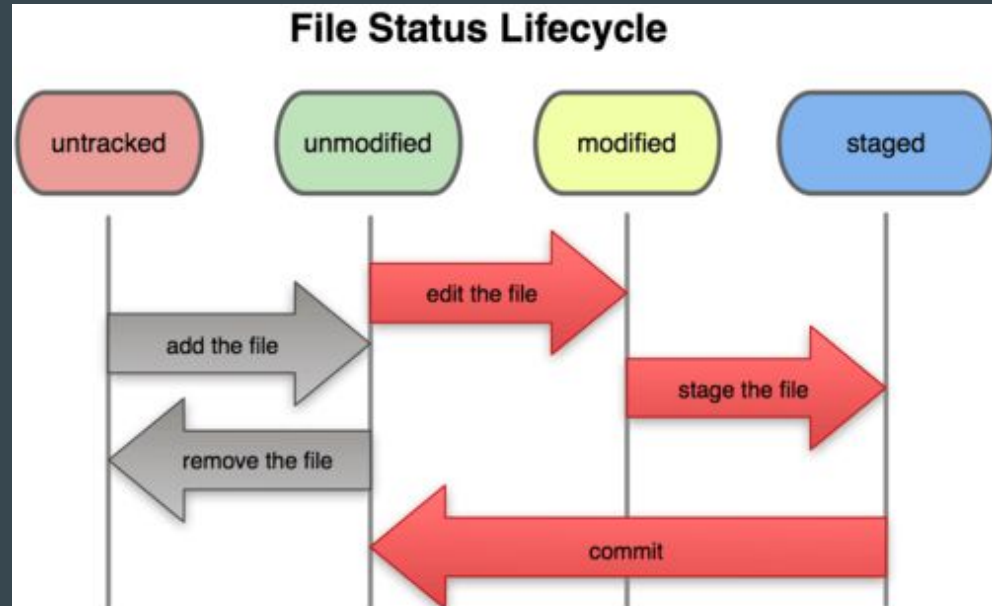
Oh shit! I screwed something up.
View recent commits: `git log`
Undo all commits back to a certain point
  `git reset <commit hash from log>`
  `git reset HEAD~<number of commits to reset>`

File Status Lifecycle

untracked   unmodified   modified   staged

add the file
edit the file
stage the file
remove the file
commit

https://git-scm.com/book/en/v1/Git-Basics-Recording-Changes-to-the-Repository

Best Practices

Commit often, but not too often: incremental progress

Keep commits to one topic: fix one bug at a time

Write useful commit messages

Don't track files which are produced by running the code

Use .gitignore to list files or file types which you don't want to track

Worst Practices

Xkcd #1296

COMMENT	DATE
CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
ENABLED CONFIG FILE PARSING	9 HOURS AGO
MISC BUGFIXES	5 HOURS AGO
CODE ADDITIONS/EDITS	4 HOURS AGO
MORE CODE	4 HOURS AGO
HERE HAVE CODE	4 HOURS AGO
AAAAAAAA	3 HOURS AGO
ADKFJSLKDFJSDKLFJ	3 HOURS AGO
MY HANDS ARE TYPING WORDS	2 HOURS AGO
HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

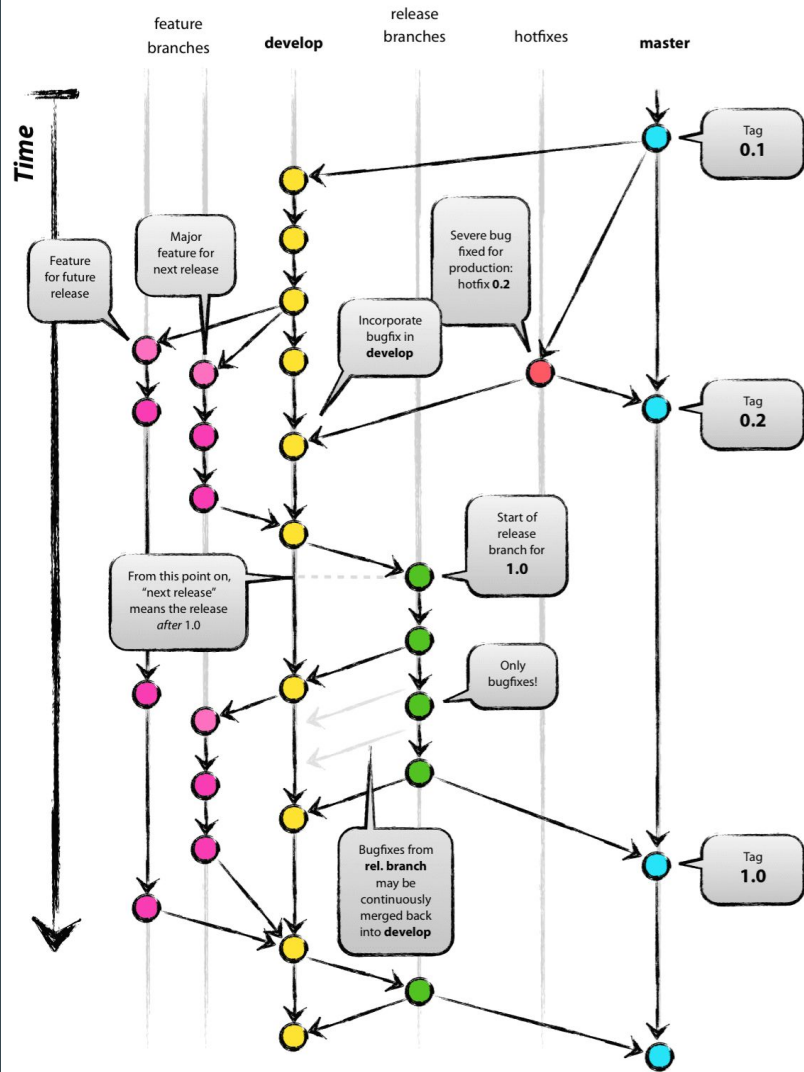PUSH REJECTED, REBASE OR MERGE

GIT PUSH --FORCE

Branching!

Allows you to keep track of multiple versions of the same code

One working version (master), one for development

Multiple branches for adding specific features

See the following link for more info and best practices involving branching:
http://nvie.com/posts/a-successful-git-branching-model/

What Else Can github do?

- Track issues, label them with "enhancement," "bug," "help wanted," etc.
- Licence your code
- Analytics
- Fork other repositories and submit pull requests
- Public and private repositories (free with an educational account)
- Wiki for your repository
- Host websites for all your repositories: branch called gh-pages will be hosted at <github username>.github.io/<repo name>
- Host a personal website <github username>.github.io

Links for your reference

Cheat Sheets:
- https://www.git-tower.com/blog/git-cheat-sheet/
- https://services.github.com/kit/downloads/github-git-cheat-sheet.pdf

Tutorials:
- http://gitimmersion.com/
- https://try.github.io/levels/1/challenges/1
- https://git-scm.com/docs/gittutorial

Documentation
- https://git-scm.com/ and https://git-scm.com/book/

Lots of other useful links can be found here:
- https://help.github.com/articles/good-resources-for-learning-git-and-github/