Collaborating with Git (and Github)

Chris Dzombak

* Chris Dzombak
* Work for Nutshell - nutshell.com
* iOS and Android
* other stuff as necessary

* I'll be walking through slides, talking, and demoing; slides and notes will be online later; watch FB group

Git

- "Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency."
- Git keeps track of your files and their history

Git tracks the files in your project and their history, so you can see what changed, when it changed, and who did it.

It makes collab easier by letting people work independently, but tracking their work and letting them merge it together.

Install & Setup

- https://help.github.com/articles/set-up-git

- It's easy

* github has a guide to installing and getting it set up
* it's easy, so I won't walk through it. you can all do it; you're smart
* for Linux: may be installed; your package manager has it
* for OS X: installed with Xcode, or if you use Homebrew or Macports, install with your package manager

Beginning:
Git for a Single User

* I know this talk is “Collaborating with Git” ; bear with me
* it’s easier to learn core concepts without adding other users and servers
* Git is useful for one person, too
* If I had a dollar for every time I started hacking on some little project, and ended up with a dysfunctional mess, I’d be retired
* believe me: use it even for simple one-student class projects.

git init

* creates a git repo in your current directory
* that's all you need to start using; no server

* demo: cd ~/mhgit/hello (contains beginner script) ; git init

Commits

- Changes are added as “commits” to your Git repository

* A set of related changes to one or more files is a commit
* Sets of changes you make are added as distinct commits in the repo
* Lets you see the development history
* can view this stream of commits, see what changed in each commit, see who wrote the commit, see when a commit happened
* can revert a commit
* can see the entire codebase as of a specific commit

Staging

- We add changes to a "staging area" before commiting
  - `git add file1 file2 ...`
  - `git status`

* to create a commit, we have to select what changes we want to include
* git status shows what in our local working directory has been Added, Deleted, Modified
* ...and what is "staged" to be included in next commit
* demo: /hello: status; add; status

Committing

- `git commit`
  - Prompts for a commit message
- `git commit -m "commit message"`

* note: good commit messages are essential for collaboration.
* bad messages only annoy your collaborators

* demo: in hello, git commit.
* external editor for message

git diff
git diff --staged

* git diff shows what has changed.
* helpful for figuring out what to stage
* and git diff --staged shows what you are about to commit

* demo: add some message to hello, diff, stage, diff, commit -m

.gitignore

- Generated files (like object code) don't need to be under source control
- Add a .gitignore file to your repo

/project/.gitignore:

```
*.o
build/*
```


gitignore causes files to be ignored in status, staging, commit, etc

Viewing History

- `git log`
- Commits are identified by hashes ("sha's")
- `git show SHA`
- `git checkout SHA`
- `git checkout master`

demo: in hello, git log, grab a sha, checkout sha, checkout master

Branching & Merging

- Split off onto a “branch” to make a series of commits working on one feature or fix
- `master` should always be releasable
- merge back into `master` (or another branch) when you’re done

* in a popular, but opinionated, workflow

git branch

- **Create a branch:** `git checkout -b branch_name`
- **Switch to a branch:** `git checkout branch_name`
- **List branches:** `git branch -l`
- **Merge branch_name INTO working branch:** `git merge branch_name`

* git merge merges INTO your current branch

* demo: checkout new branch, add something, merge into master

Merge Conflicts

- Question: what if I change something in two branches, then merge them?
- Git asks me to fix it myself
- Then commit the result

* demo: branch, ask for name & print name, fix typo on master, merge

DASH: pyname`

Questions?

Let’s pause for questions so far.

Working with remotes

- remotes: servers with Git repos on them
- Git lets you push your updates to a server and pull your (or others') work
- Create a Github account: https://github.com/signup/free
- Configure Git with your name/email: https://help.github.com/articles/set-up-git

Create a Github repo

- https://github.com/new
- Set up our existing repo to be aware of Github:

- `git remote add origin git@github.com:user/repo.git`
- `git push -u origin master`

* demo for hello
* note easy instructions in case your repo is totally new or already exists elsewhere

* -u tells git that from now on, `master` is associated with ("tracking") a branch on origin. don't worry about this too much for now.

`git push`

- Pushes all your commits to the server
- For all branches which are on the server already ("tracking" branches)
  - **use** `git push -u new_branch` **if you want to push a new branch**
- `git push origin HEAD` **is usually better practice**

* since we've already seen git push, let's talk about it
* push origin head only attempts to push the current branch.
* better just because there won't be any unexpected side effects

* demo: since we pushed, we can see our master on github
* demo: git branch -l ; git push -u other_branch, check github

Adding Collaborators

- https://github.com/USER/REPO/admin/collaboration
- Collaborators can interact with the repo just like you can
- Good for class projects, etc

demo: add `mhseiden`

Github Private Repos

- Repos on Github are public by default (anyone can read, your collaborators can write)
- Paid users can create some private repos (only collaborators can read/write)
- get a free upgrade to have a few private repos: https://github.com/edu

git clone

- So let's say you've been added as a collaborator on a class project.
  - `git clone git@github.com:user/repo.git`

demo: login to tesla, git clone our repo

`git pull`

- Pull someone's changes and merge them into your current branch

- **Remember, `git status` shows the current branch**

* demo: make some change on tesla on master, commit & push, pull to local repo

Checkout someone else's branch

- `git fetch origin pulls` down the latest branches and commits from your remote
- First time: `git checkout -t origin/someones_branch` to check out `someones_branch` and associate it with the remote branch
- After: `git checkout someones_branch`

* once you check it out, you work with it just like any other branch

* demo: create a branch on tesla, commit and push that branch. fetch and checkout on Mac, commit something
* then merge into master and push. note both branches are pushed

Fun Fact

- `git pull`
- **is the same as**
- `git fetch && git merge HEAD origin/HEAD`
- **(unless you're using a different** `remote` **than origin)**
- **for more,** `man git-pull`

Github Workflow

- Contributing to OSS projects via Github is a little different
- Create a "fork" of someone's repo (via Web interface)
- Add feature in your fork, in a feature branch
- Request repo owner to pull and merge your branch into their repo

* adding a feature in your fork means:
git clone your repo
git checkout -b feature
git add, git commit
git push -u origin yourbranch

Walk through: add link to TM2 OSS project to https://github.com/andrewsardone/istextmate2outyet

Github Workflow

- https://help.github.com/articles/fork-a-repo

Additional Reading

- The Git Parable: http://tom.preston-werner.com/2009/05/19/the-git-parable.html
- A Note About Git Commit Messages: http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html

Parable walks through how Git was developed; good intro to the problems Git solves and paradigm it uses

Note goes over best practice for commit messages

Additional Resources

- Help for Git and Github: https://help.github.com/
- Cheat sheet: http://cheat.errtheblog.com/s/git/
- Pro Git book: http://git-scm.com/book
- My Git bookmarks: http://pinboard.in/u:cdzombak/t:git

Additional Resources

- Google and Stack Overflow

searching for an error message, or for something like “git change commit message” will almost always exactly answer your question

Additional Tools

- gitk - cross-platform GUI: http://lostechies.com/joshuaflanagan/2010/09/03/use-gitk-to-understand-git/
- GitX(L) - git GUI for OS X: http://gitx.laullon.com/
- tig - terminal-based GUI for *nix: http://gitready.com/advanced/2009/07/31/tig-the-ncurses-front-end-to-git.html
- Github's site has a lot of cool stuff built in

Questions?

Contact

- chris@chrisdzombak.net
- @cdzombak
- github.com/cdzombak