Publishing Software,

GitHub & zenodo

...

Megan Potterbusch ~ Dan Kerchner
GW Libraries & Academic Innovation

March 6, 2018

Agenda

- Goals: Citable, durable, publishable software
- Collaborating with GitHub (Hands-on)

What is Software Publication?

- [image]

Levels of “Publication” for Software

- Publically available generally
- Citable online resource
- Citable and Peer reviewed
- Supplement to traditional article publishing

Make your work citable!

DOI == Digital Object Identifier

Persistent, externally managed, standardized

http://dx.doi.org/

Standard: ISO 26324

Software as: Supplemental to Publication

Software as: Peer-Reviewed Scholarship

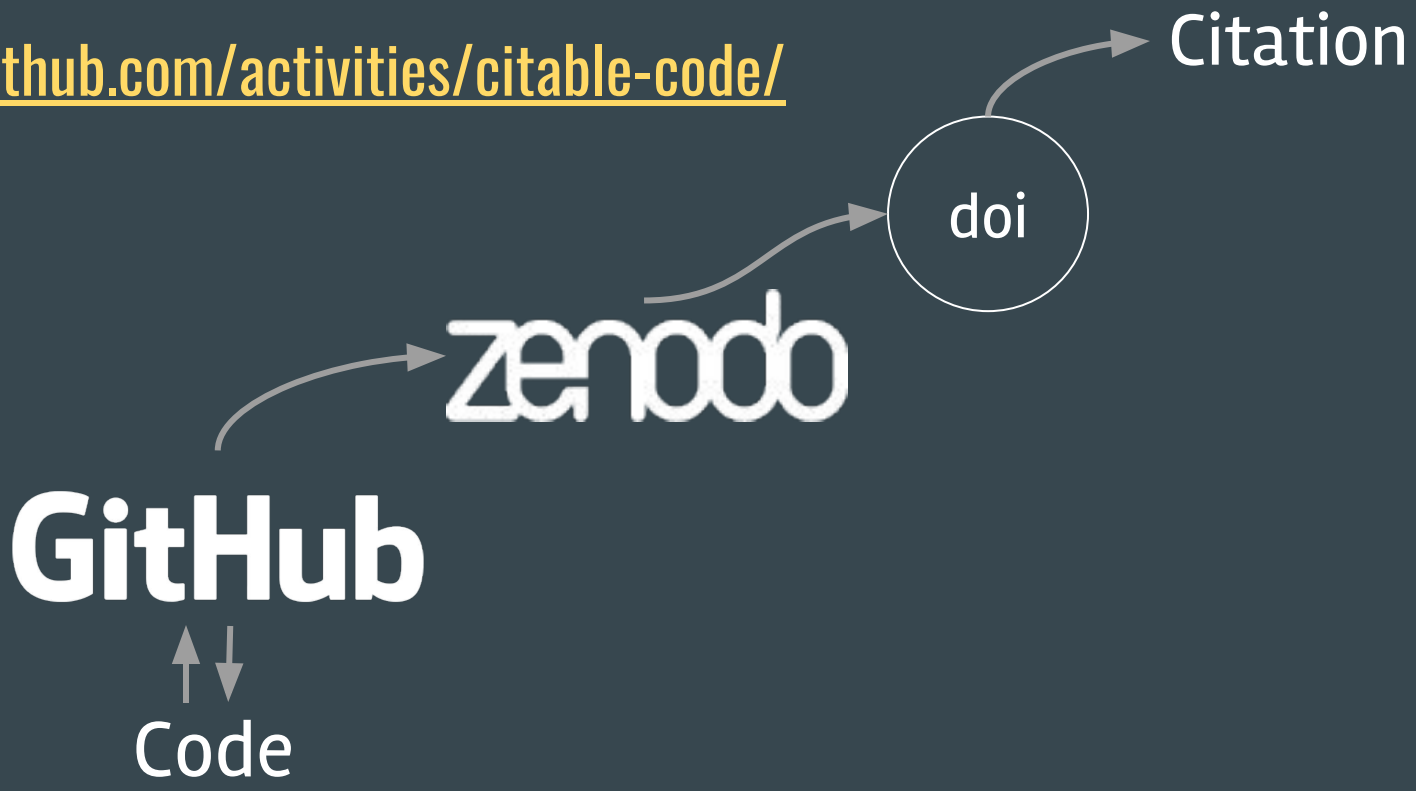
Software as: A Citable Resource

Software as: An Online Resource

Collaborating with Github

Context: Making Your Code Citable

guides.github.com/activities/citable-code/



GitHub Agenda

- Why GitHub?
- A tour of a repository
- Hands-on:
  - Create your own new repository
  - Collaborate on a repository
  - Make a release of your code
  - Fork an existing repository

Why GitHub?

- Version Control - uses `git`
- Collaboration
- Open (or not)
- Pricing - free for open, cheap for private, and free for students education.github.com (and /pack)
- Code with context

Tour of a Repository: About your code

- **README.md** ← showcase your work, this is your home page
- **LICENSE** file ← Important! choosealicense.com
- (Optional) Wiki pages

These use Markdown - lightweight formatting

Tour of a Repository - The Code

- Code
  - history of commits on a branch
  - history of commits on a file
- Branches
  - Master branch vs. others
  - Visually represented with the Insights → Network graph

Tour of a Repository - Code as a Project

- Issues - tags, assignment, releases, comments
- Milestones and Releases
- Pull requests
- Comment threads - on issues, commits, etc.
- Insights (graphs)

Tour of a Repository - Connections

Between Github users/repositories

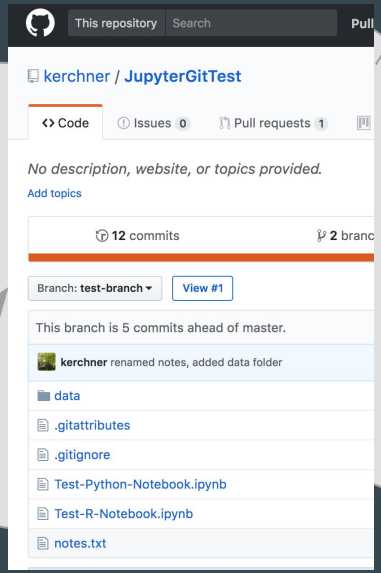
- Watch/Star/Follow
- Mentions

With other software development tools

- Integrations & Webhooks (under repository Settings)

Section Title Goes Here

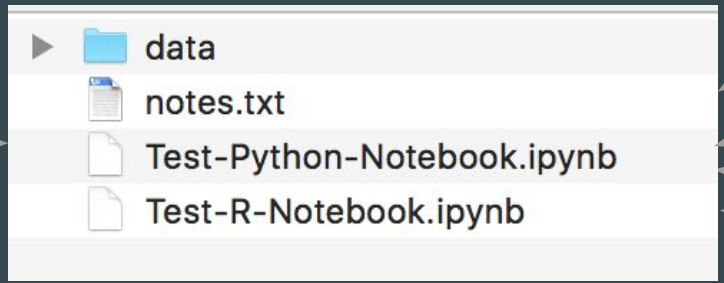
Your repository on GitHub



Apps you use that modify your files

GitHub Desktop

Files on your computer - your *local* git repository



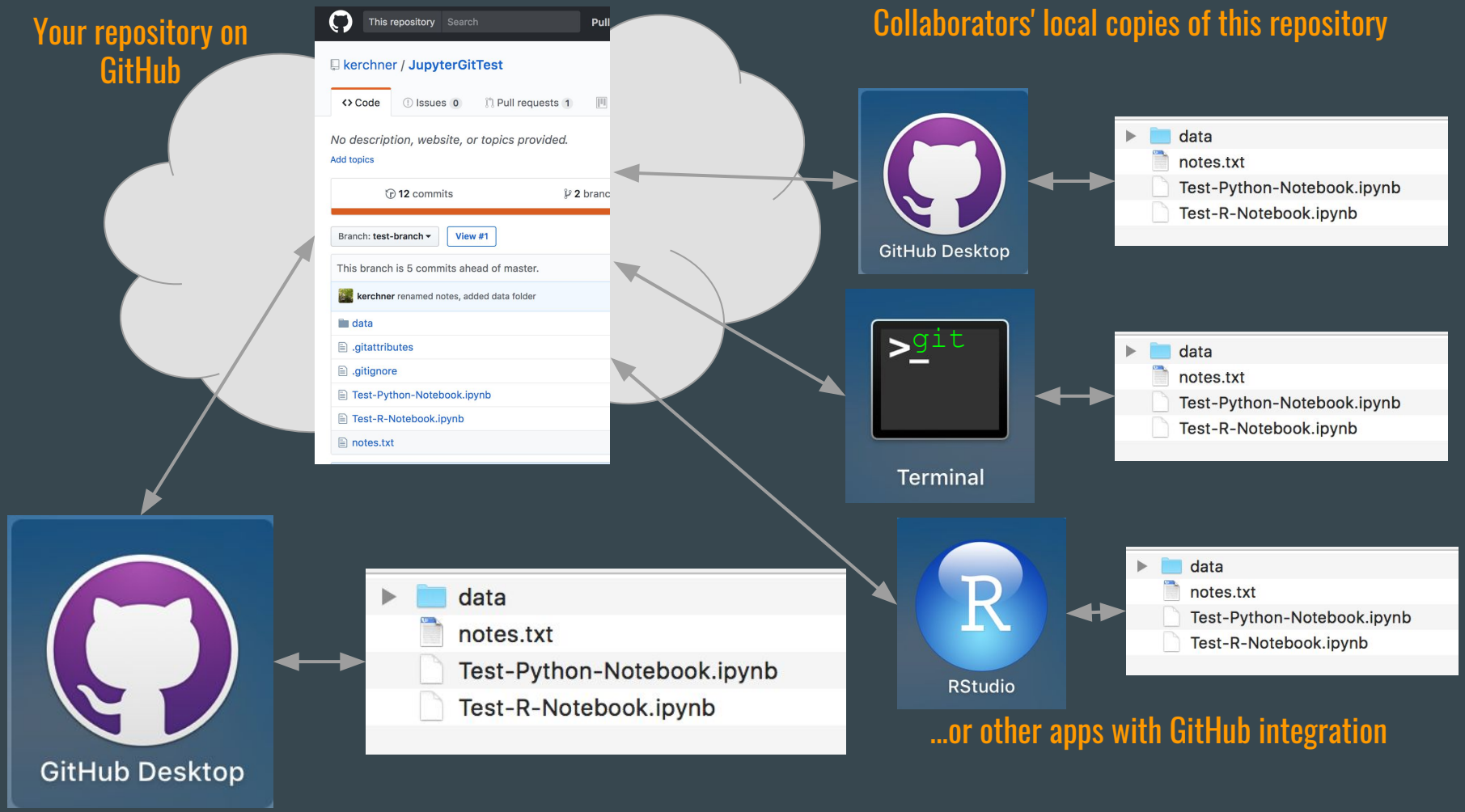
jupyter

Atom

RStudio

Microsoft Excel

** or command-line git, RStudio, Atom, etc.*


Your repository on GitHub
This repository
Search
Pull
kerchner / JupyterGitTest
Code
Issues 0
Pull requests 1
No description, website, or topics provided.
Add topics
12 commits
2 branc
Branch: test-branch
View #1
This branch is 5 commits ahead of master.
kerchner renamed notes, added data folder
data
.gitattributes
.gitignore
Test-Python-Notebook.ipynb
Test-R-Notebook.ipynb
notes.txt
Collaborators' local copies of this repository
GitHub Desktop
data
notes.txt
Test-Python-Notebook.ipynb
Test-R-Notebook.ipynb
>_git
Terminal
data
notes.txt
Test-Python-Notebook.ipynb
Test-R-Notebook.ipynb
GitHub Desktop
data
notes.txt
Test-Python-Notebook.ipynb
Test-R-Notebook.ipynb
R
RStudio
data
notes.txt
Test-Python-Notebook.ipynb
Test-R-Notebook.ipynb
...or other apps with GitHub integration

Basic Terminology

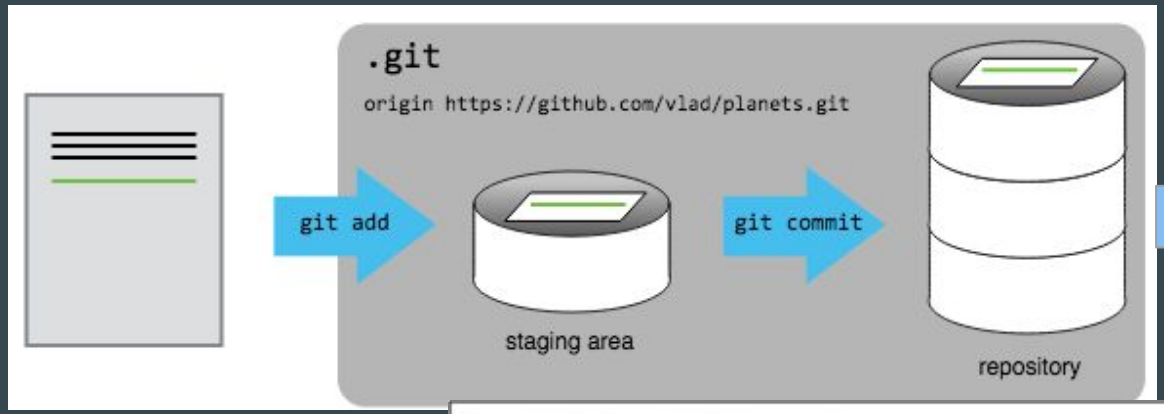
Add/remove - puts changes in a staging area

Commit - commits staged changes to the (local) repository

Push - pushes local repository changes to the remote repository

Pull - pulls remote repository changes to the local repository

add ~ commit ~ push



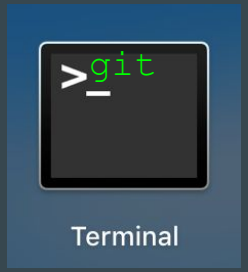
← local

git push origin

←on GitHu

Ways to use Git

*or other GUI



...and many IDEs

Git repository anatomy

- Open the folder containing your local git repository. It contains a `.git` directory which contains all the metadata about the repository (that is, the local copy)

Let's Try It!

Create your own GitHub repository

Prerequisites / Setup

EITHER

- Install **GitHub Desktop app** via desktop.github.com
- Under *Preferences...* sign in to your GitHub account

OR:

- Install **Git command line** via git-scm.com/downloads
- ```
  git config --global user.email "you@youremail.com"
  git config --global user.name "Your Name"
  ```
  ```

1. Create a repository, make some changes

2. Collaborate on a repository

Form groups of 2+

3. Make a release of your code

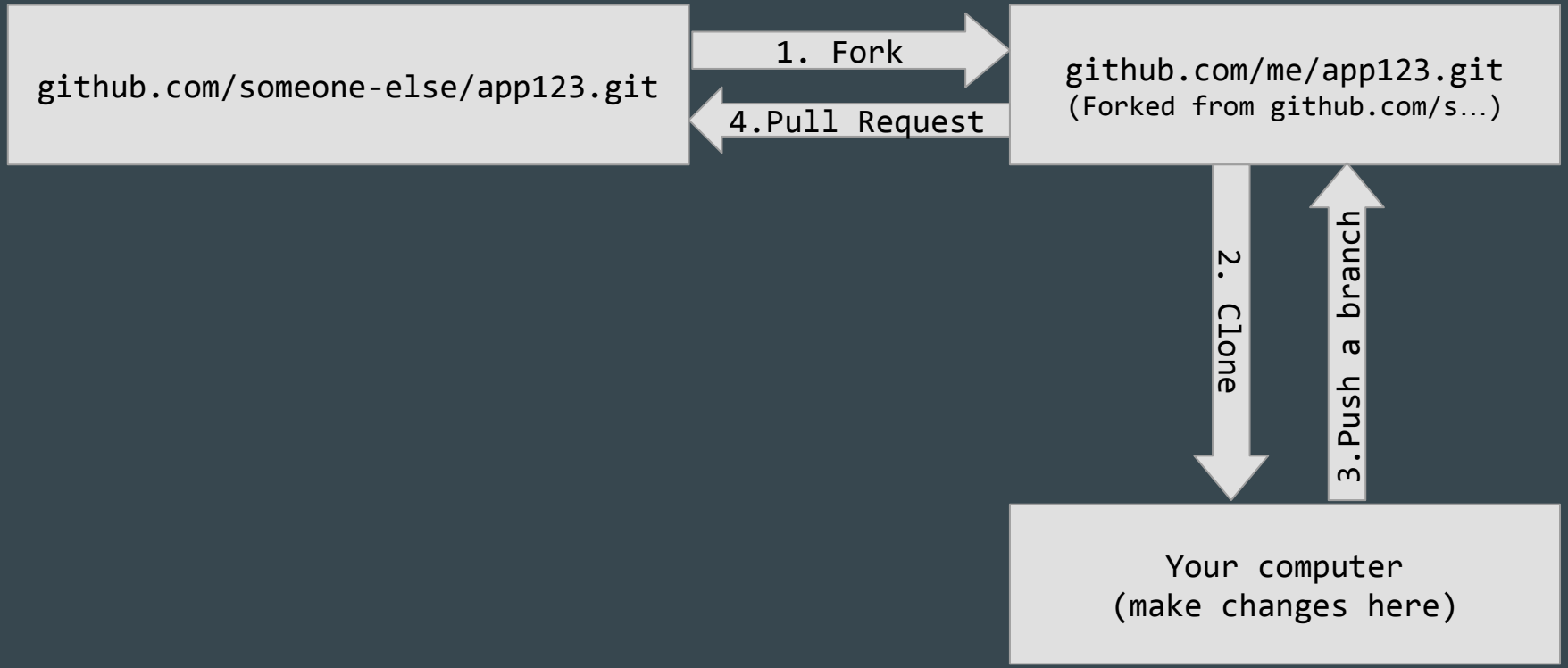
4. Fork a Repository

About forking...

Scenario: There is a Github repository on which you are *not* a collaborator, but you would like to:

- contribute a feature or fix to it, or
- use it, but with your own modifications

Forking a repository



Advanced Git Topics

- Merges that require manual merging
- Resetting
- Re-basing
- Squashing
- Reverting

Ways To Use Git

- Git command line
- GitHub Desktop App (one of many - see git-scm.com/downloads/guis)
- Github Mobile App
- Integrations w/Editors, IDEs, and other apps
  - R Studio
  - Atom
  - ...many others

Git - Command Line - some common commands

```
git config --global user.email
git config --global user.password
git init <name>  # creates new repository in this folder
git clone <URL for repo>
git status       # -s gives a short version
git branch <branch-name>    # creates a branch
git checkout <branch-name>  # switches to a branch
git diff    /  git diff --staged
git add <file(s)>  /  git rm <file(s)>  /  git mv <file>
git commit -m 'Comment goes here'
git push origin <branch> # When working with forks, you might
git pull origin <branch> # use remotes other than "origin"
```

Git - Command Line - continued

```
git log
git merge <branch to merge from>
git rebase <branch to rebase off of>
git reset --hard  # Warning! You'll lose your changes!
git stash
git fetch   # like git pull, but without merging
```

Other Uses for Github Repositories (besides code)

- Sharing data
- "Document" collaboration (e.g. legislation)
- Course sites (e.g. DSCN 6279; ISTM 6212)
- Github Sites (e.g. gwu-libraries.github.io/sfm-ui/)
- Publishing and Citing your code (you can use Zenodo to mint a DOI for a Github repository)
- and more

Key takeaways

- Make your code presentable in GitHub:

  * README.md
  * LICENSE
  * Commit messages
  * Releases - release notes should describe new features/ changes, upgrade instructions, known issues
  * Use issues - helps explain WHY you changed code. Open issues communicate desired fixes/enhancements.

More takeaways

- The GitHub Desktop app seems useful, but mainly for "basic" things. Ultimately, it's probably good to learn and use git shell commands. (Especially good on your resume!)

- For team projects, one person can create the repository on their git account and add others as collaborators.
  Or, you can create an "organization" account - go to your profile's Settings -> Organizations -> New Organization

Even more takeaways

- Insights → Network is a good way to visualize what's going on with the branches.
- When working together - or even alone - use branches. Merge a branch using a pull request.
- When you're working on a branch for an extended period, consider periodically merging (or rebasing) from the master branch, so your branch doesn't diverge too far from the code on master.

To Learn More...

- *Pro Git* book, free online: **git-scm.com**
- Lynda.com: *Up and Running with Git and GitHub*
- github.com:
  - help.github.com
  - guides.github.com
  - try.github.io (codeschool)
  - services.github.com/on-demand/
  - resources.github.com/webcasts/

Thanks!

- Dan Kerchner kerchner@gwu.edu
- Megan Potterbusch mpotterbusch@gwu.edu

Coding Consultations (GitHub, etc.) - schedule via:
go.gwu.edu/coding

Research Consultations (Data, etc.) - schedule with Megan via:
library.gwu.edu/reference/research-consultations