Enhancing collaboration and reproducibility…

… using GitHub and distributed version control

Jay Brodeur | brodeujj@mcmaster.ca

Portage Webinar | 2020-10-06

McMaster University sits on the traditional Territories of the Mississauga and Haudenosaunee Nations, and within the lands protected by the “Dish With One Spoon” wampum agreement

(Indigenous Education Council, May 2016).

Learning objectives

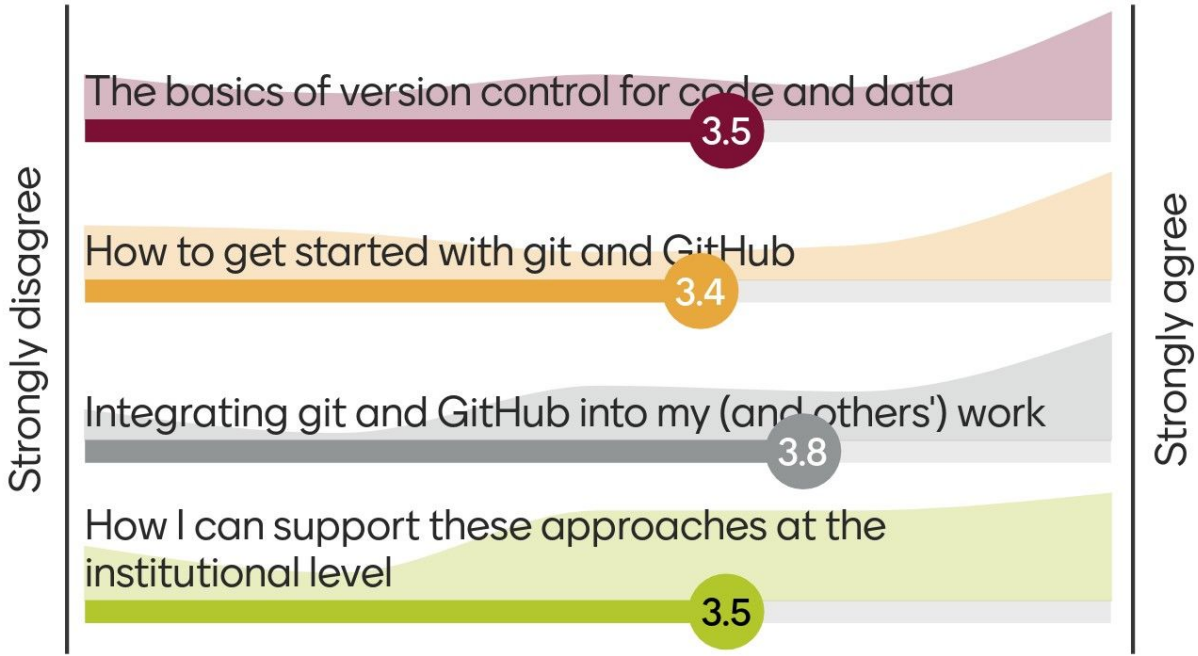
At the completion of this webinar, you should be able to:

- Explain the purpose and general function of version control systems
- Apply a variety of tools (git, GitHub, GitHub Desktop) to manage file versions within a ***repository***
- Apply best practices for efficiently managing and sharing repositories
- Describe how systems like GitHub can be used to support research collaboration and transparency
- Identify opportunities to implement these tools & practices to support research in your group or organization

But first …

A few questions for you

Generally, I'm interested in learning about



My familiarity with version control with git & GitHub

This is all new to me!

I have heard of some of these

I know about them, but haven't used them before

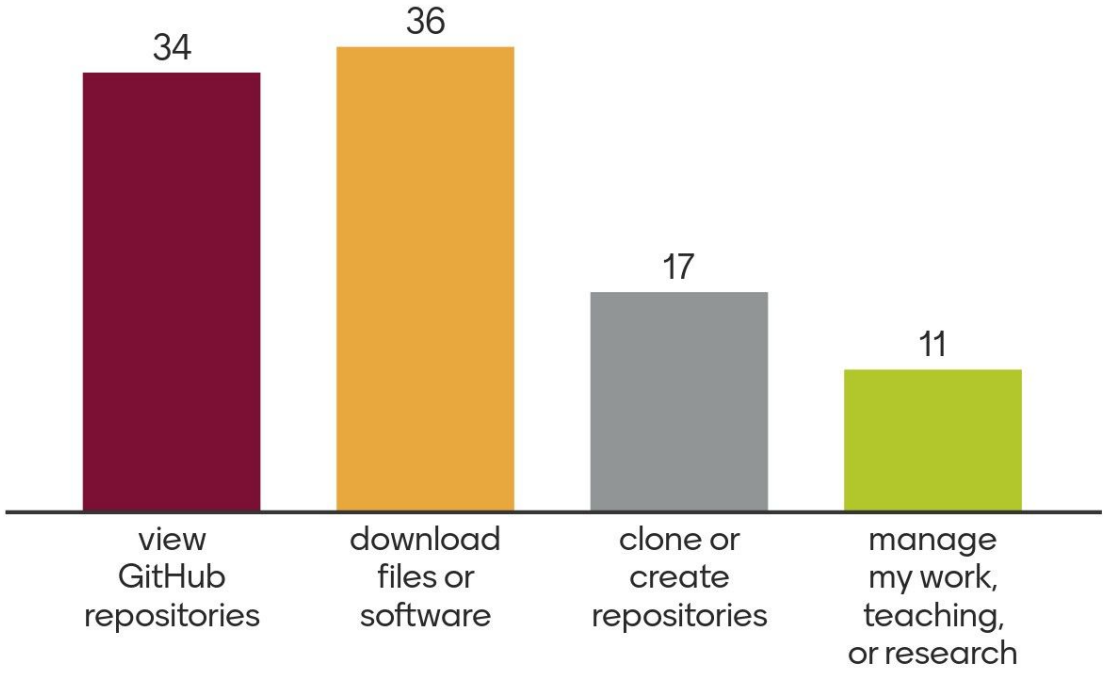
I'm familiar with them and I've used them

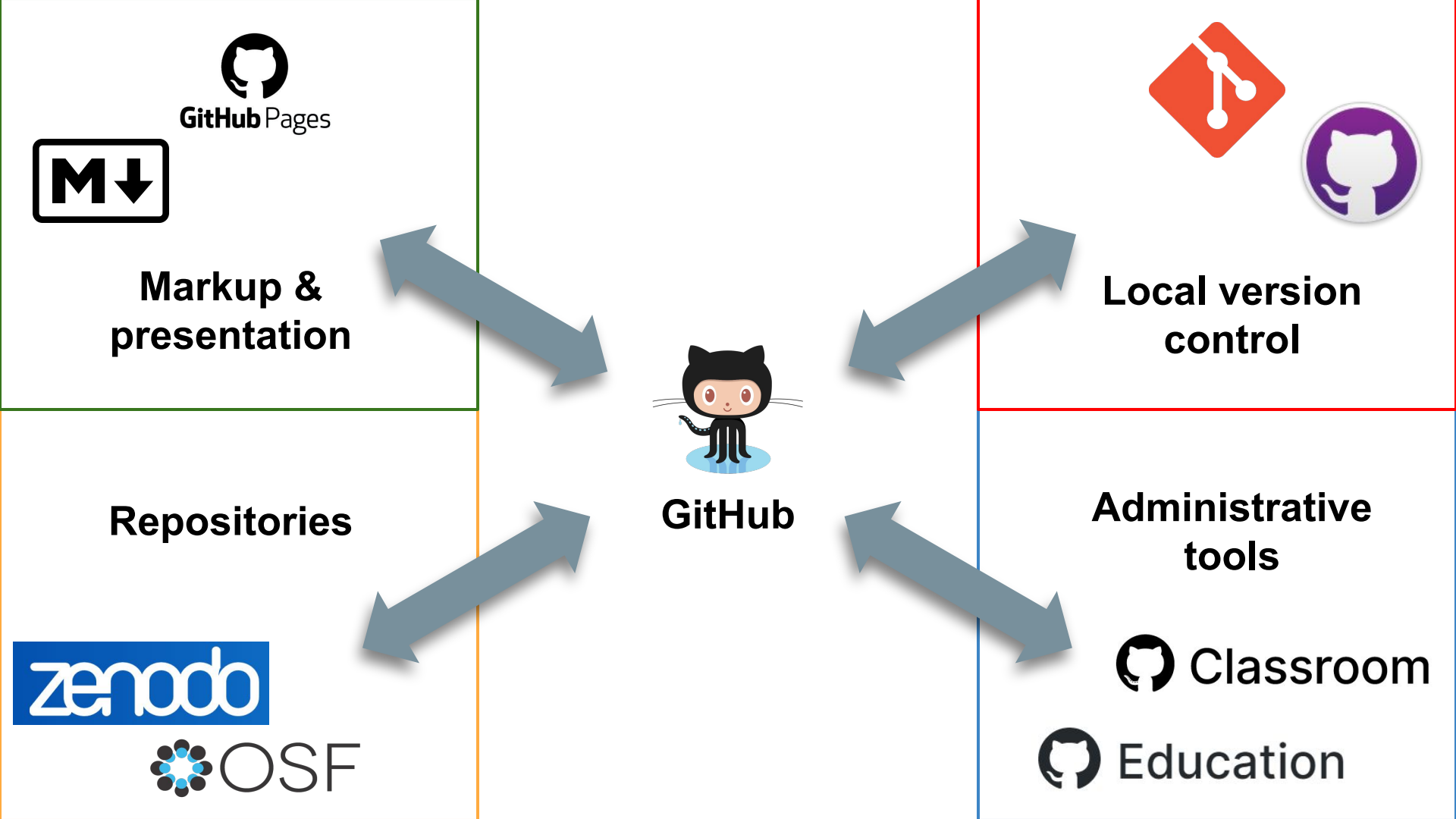
0

I should probably be giving this webinar!

47

In the past, I have you used GitHub to ...




GitHub Pages
Markup & presentation
Local version control
Repositories
GitHub
Administrative tools
zenodo
OSF
Classroom
Education

Outline

Version control systems - types and value

Basic workflows in git and GitHub

Managing collaboration, access, & sharing

Sharing results: Markup and presentation

Administrative tools

Version control systems

(and why you might need them)

Local version control

my-research/
  readme.txt
  ↳ data/
    ↳ trial1results.csv
  ↳ scripts/
    ↳ t1analysis.py

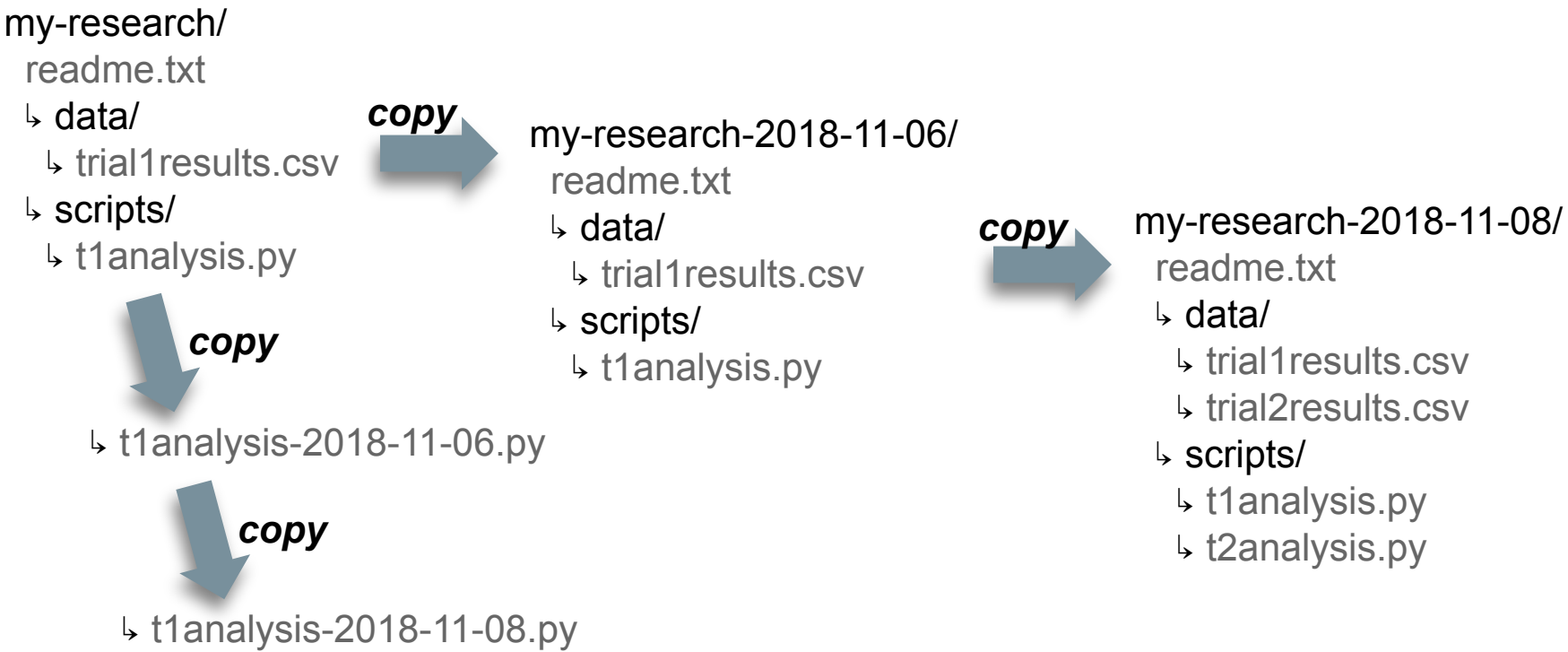
Local version control (the hard way!)

my-research/
- readme.txt
- ↳ data/
  - ↳ trial1results.csv
- ↳ scripts/
  - ↳ t1analysis.py

↳ t1analysis-2018-11-06.py

↳ t1analysis-2018-11-08.py

Local version control (the hard way!)



Local version control

You can track versions manually!
BUT, it's prone to errors:

- Writing to the wrong file/folder
- Overwritten files
- Misnamed (or poorly named) files
- "I just keep forgetting to do it"
- "Which old version is the correct one?"

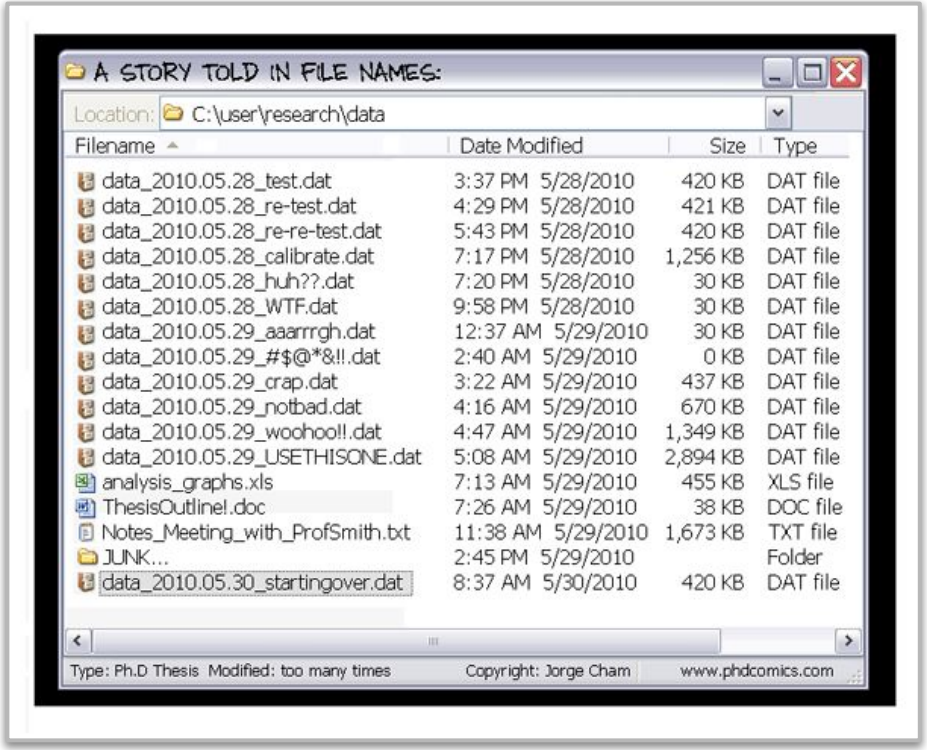


Image credit: PhDComics | Image © jorge cham

Local version control

Database system records changes to files and folders over time

Benefits: Can be mostly automated; consistent and dependable; traceability

Challenges: Not conducive to collaboration; local system failure could lead to data loss

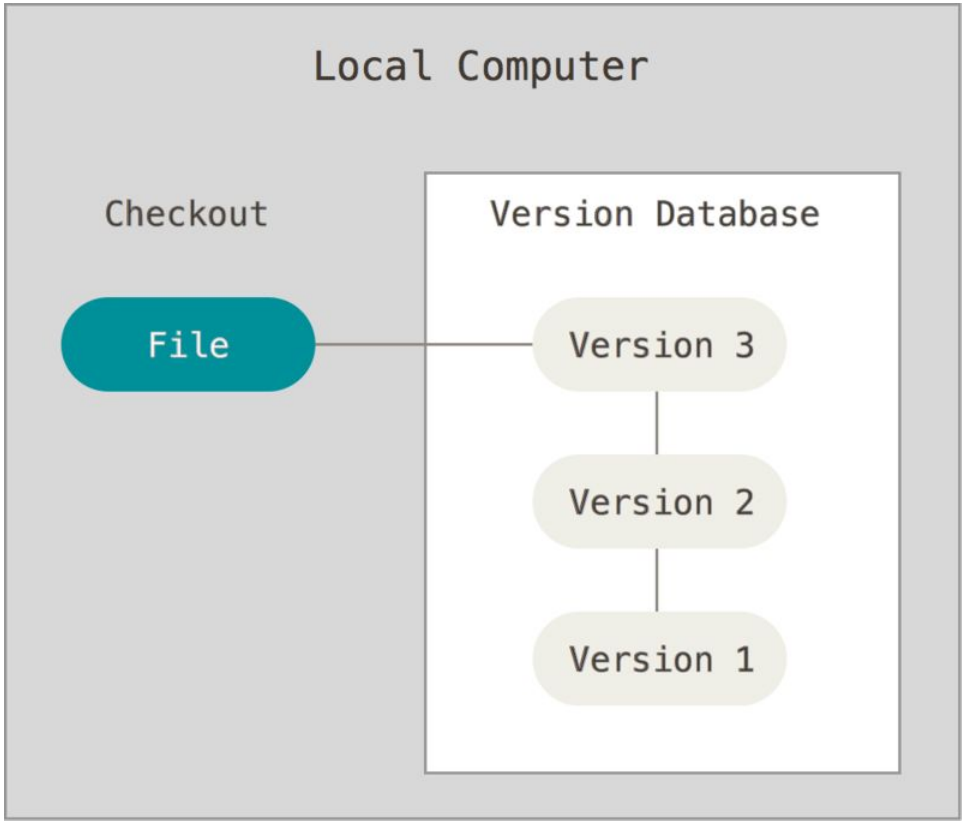


Image credit: Pro Git

Centralized version control

Central (remote) database records changes from multiple local users

Users ‘check out’ a version they are working on.

Benefits: Allows for collaboration & granular permissions

Challenges: Can get ‘locked out’ or lose access altogether during outages

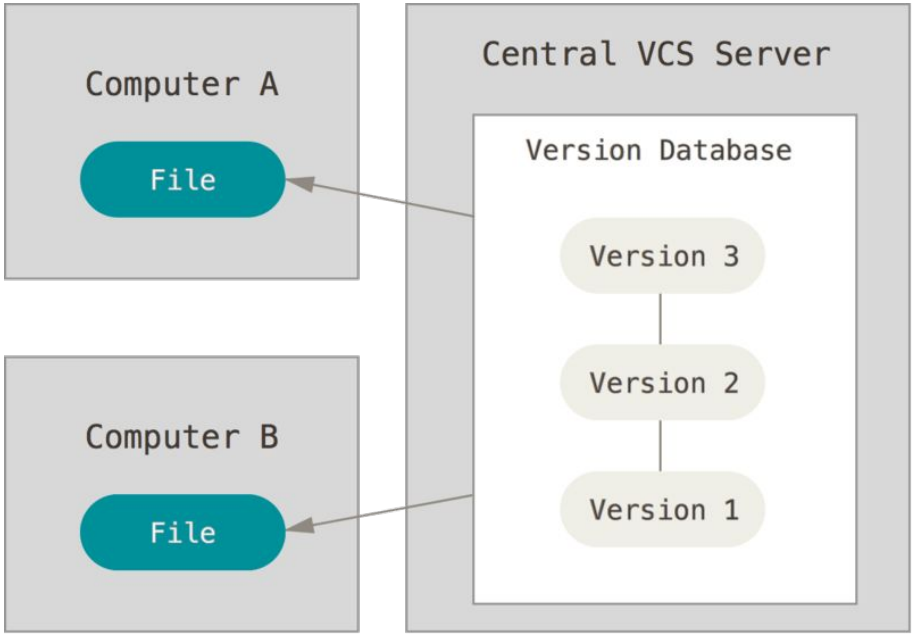


Image credit: Pro Git

Distributed version control

Clients (users) ***clone*** the entire repository locally

Clients work locally; ***push*** changes to the server

Changes managed and ***merged*** at the server

Clients ***pull*** new changes

Benefits:

- Collaboration & concurrent development
- Granular permission
- No single point of failure

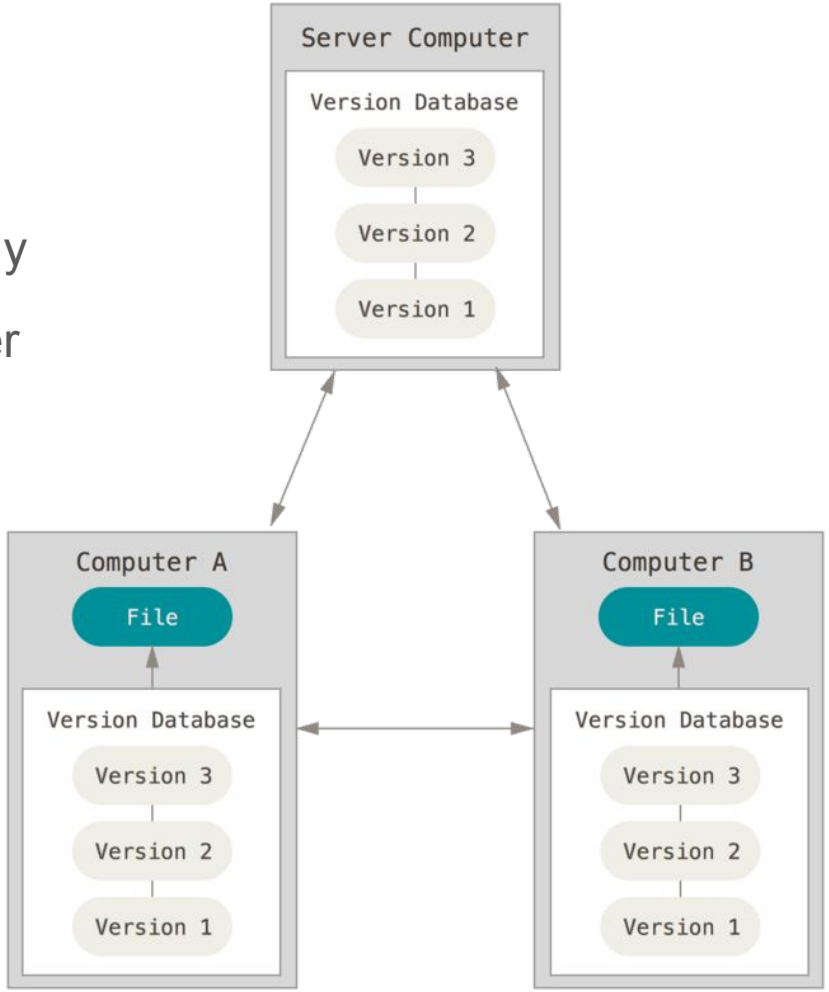


Image credit: Pro Git

Why use distributed version control?

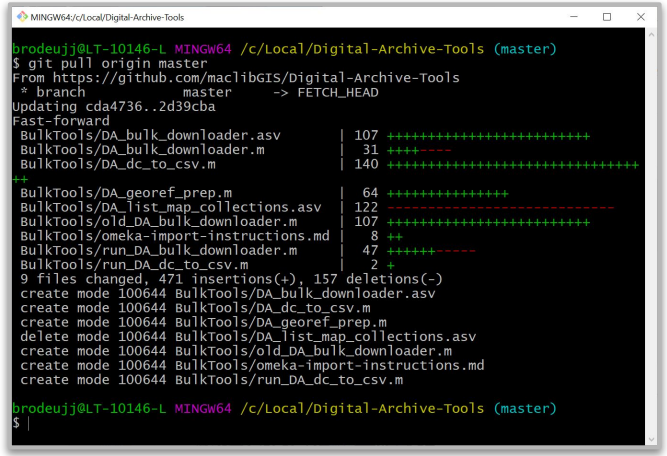
Distributed version control software allows you (and your collaborators) to:

- Track, compare, and revert changes (more quickly and granularly)
- Enable and manage collaborative development
- Deal with challenges of scale (# files, # changes, # collaborators)
- Share materials (openly or controlled); allow collaboration and reuse
- Backup your work to an external repository

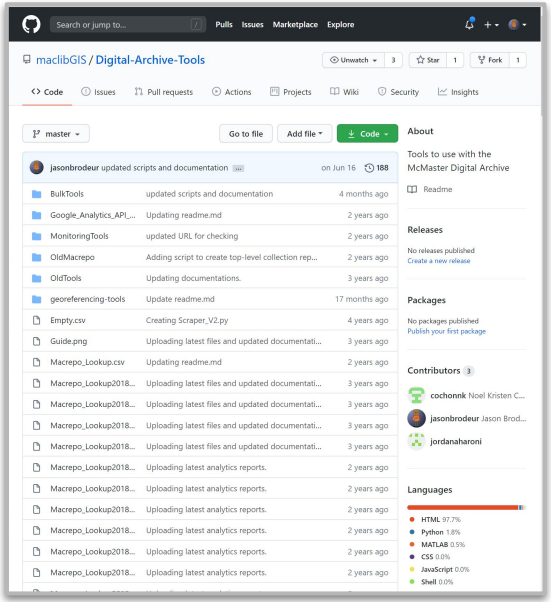
Use it to manage:

- **Software / code** — Openrefine: github.com/OpenRefine/OpenRefine
- **Datasets** — OpenIndexMaps: github.com/OpenIndexMaps
- **Documentation** — DCN data curation primers: github.com/DataCurationNetwork/data-primers
- **Books** — Git from the Bottom Up: github.com/jwiegley/git-from-the-bottom-up
- **Websites** — UBC Research Commons' intro to git: ubc-library-rc.github.io/intro-git/

git is a free and open source ***distributed version control system*** to handle everything from small to very large projects with speed and efficiency.



GitHub is a web-based hosting service for version control using git. It offers all of the distributed version control and source code management functionality of git as well as additional features.



Basic workflows

in git and GitHub

1. ***Initialize*** or ***clone*** a ***repository*** (in git)

A ***repository (repo)*** is a set of files/directories managed with a VCS

Initialize git to create a new local repo in a selected directory (with or without files)

```
$ cd C:/Local/my-repo
$ git init
```

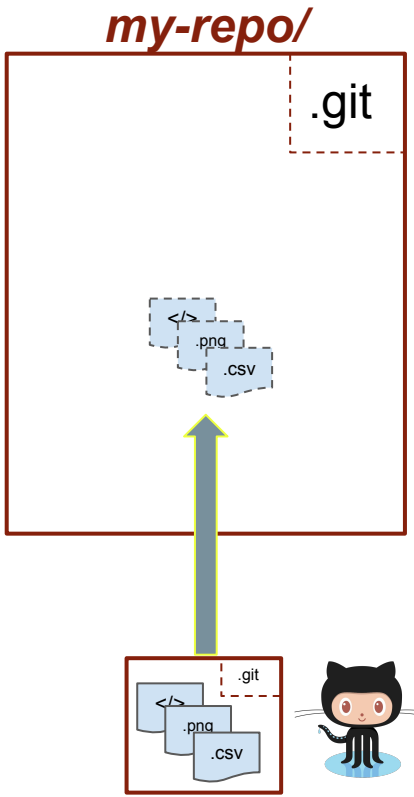
OR

Clone an existing repo (e.g from GitHub) to your local system

```
$ git clone https://github.com/username/my-repo.git
```

Tip: Create a readme.md and LICENSE file in the top directory

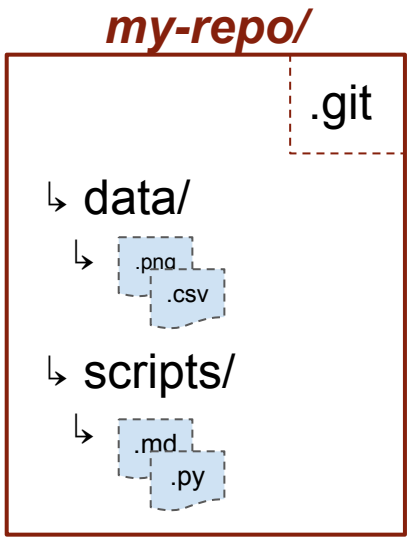
➢ These become your repo's readme file and license



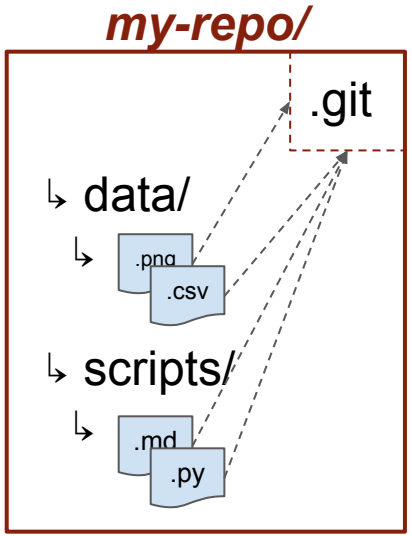
2. Do your work

Create and edit files and folders

NOTE: Changes aren't tracked until you take a snapshot of them.



3. ***Add*** or ***update*** files (in git)



Add: tell git to begin keeping track of a file and its versions

```
$ git add README.md
$ git add --all
$ git add *.py
```

Update: tell git to take note of the changes that has been made to a file (staging)

```
$ git add -u
```

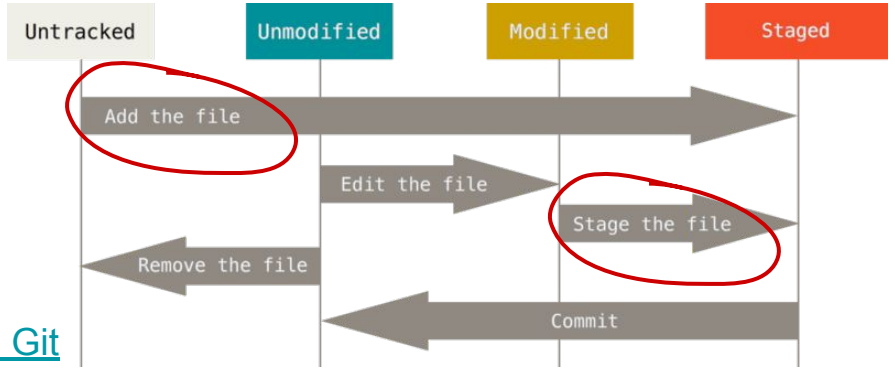
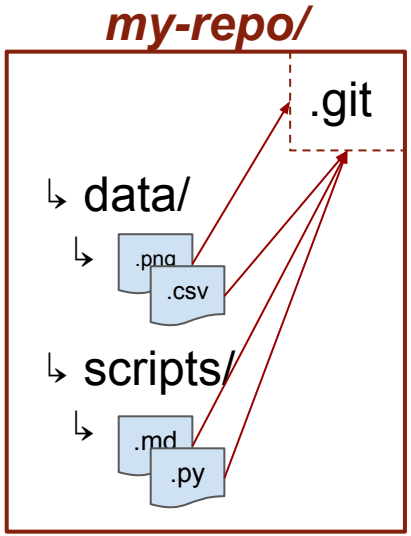


Image credit: Pro Git

4. ***Commit*** changes (in git)



Commit: tell git to take a snapshot of all staged (changed) files (while keeping old snapshots):

```
$ git commit -m "Title" -m “Description......"
```

- Add a short comment and a longer description

Add and ***commit*** at once with:

```
$ git commit -a -m "Title"
```

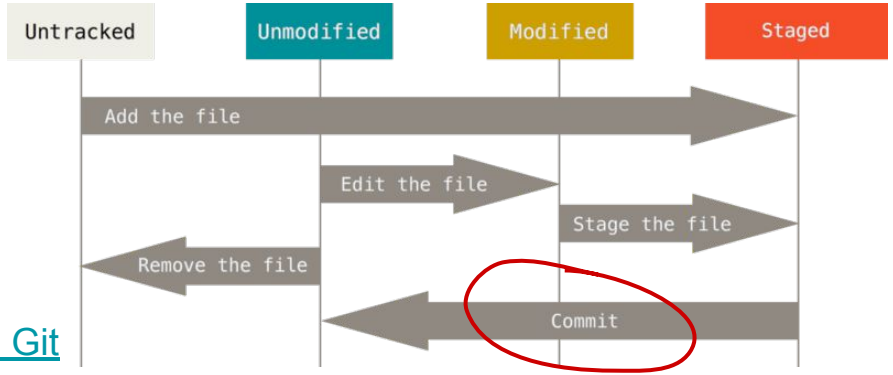


Image credit: Pro Git

OK

What Now?

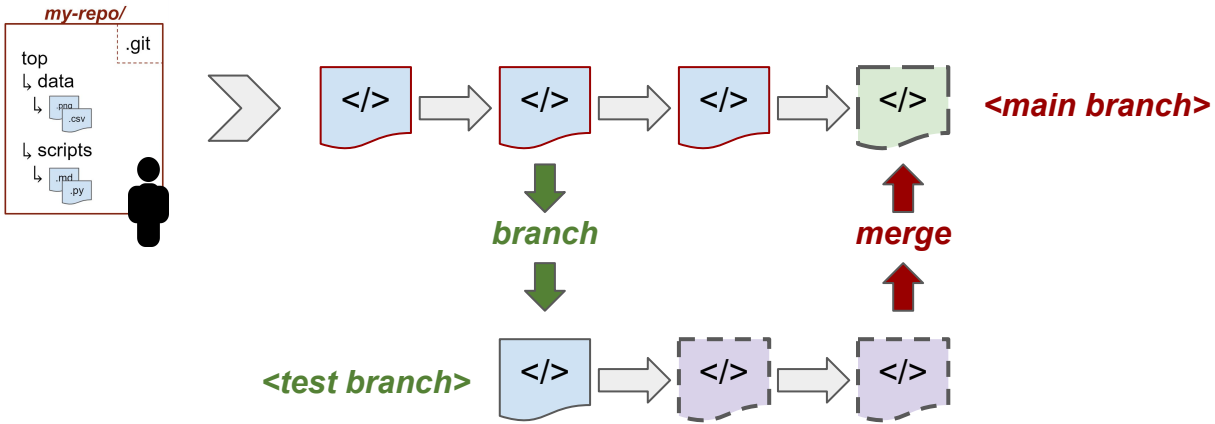
5a. Continue to work, add + commit

5b. Make a new ***branch*** to allow separate development

Clones the ***main/master branch*** (but tracked in the same git)

Allows separate development without breaking what's in place

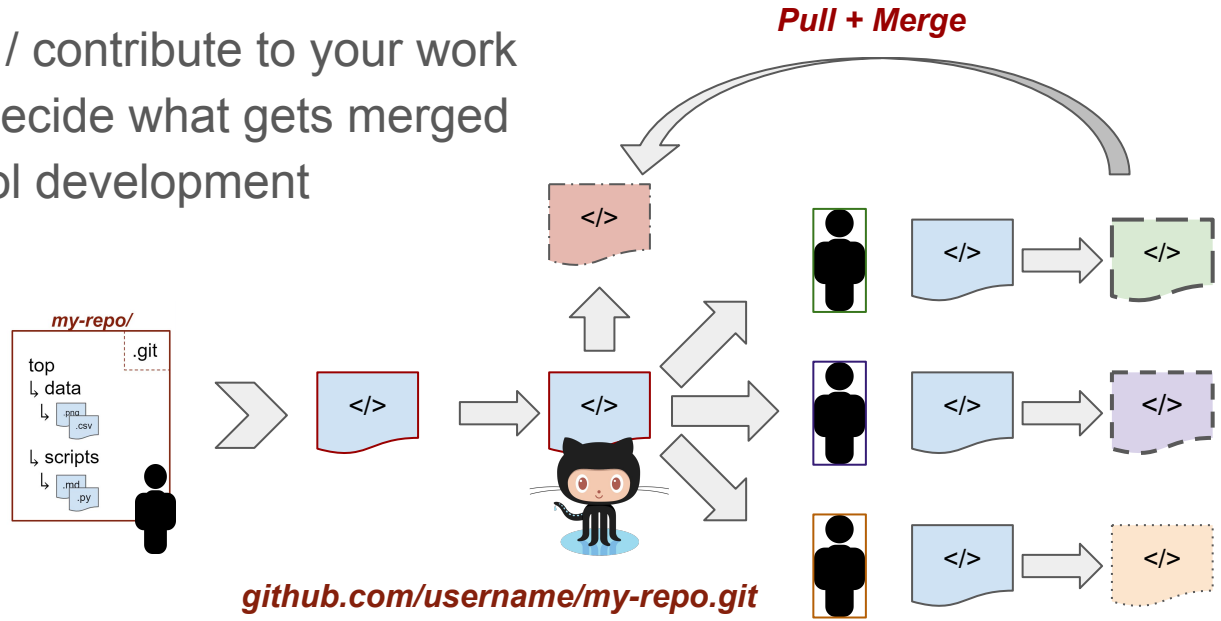
New branch can be later ***merged*** into the ***main/master*** one



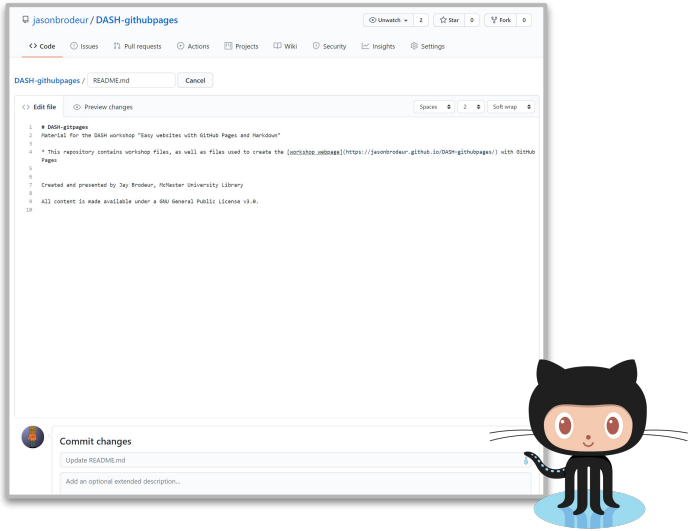
5c. ***Push*** changes to a remote repository (e.g. GitHub)

Push: send modified files (and git database and associated metadata) to a remote repository (e.g. GitHub, or another hosted repository)

- Disseminate & share
- Enable others to modify / contribute to your work
- Manage contributions; decide what gets merged
- Merge changes & control development



In GitHub

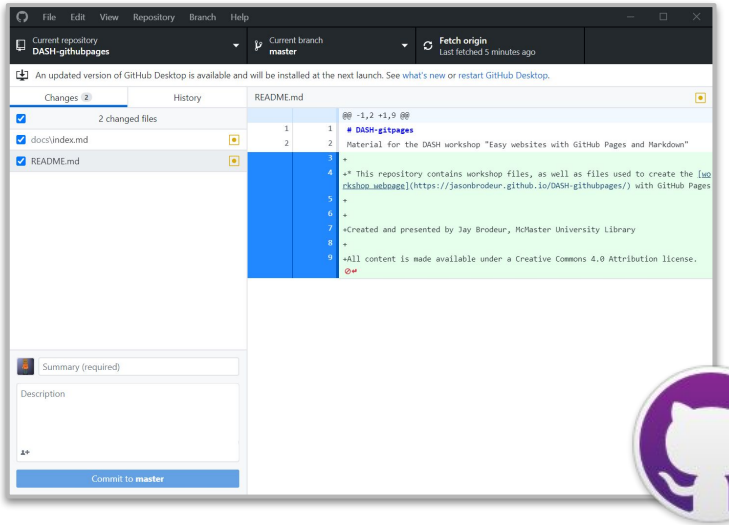


Use the web interface to:

create / clone repository (with readme)

- upload, edit etc.
  - add + commit
    - branch + merge

In GitHub Desktop

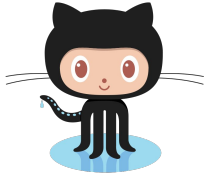


GitHub Desktop is a desktop application for local version control and interaction with GitHub using a GUI.

Managing collaboration, access, & sharing

with GitHub

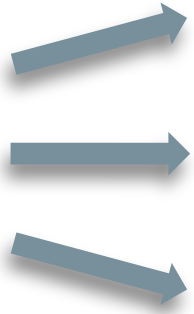
In GitHub



Use the web interface to:

↦ create / clone a repository

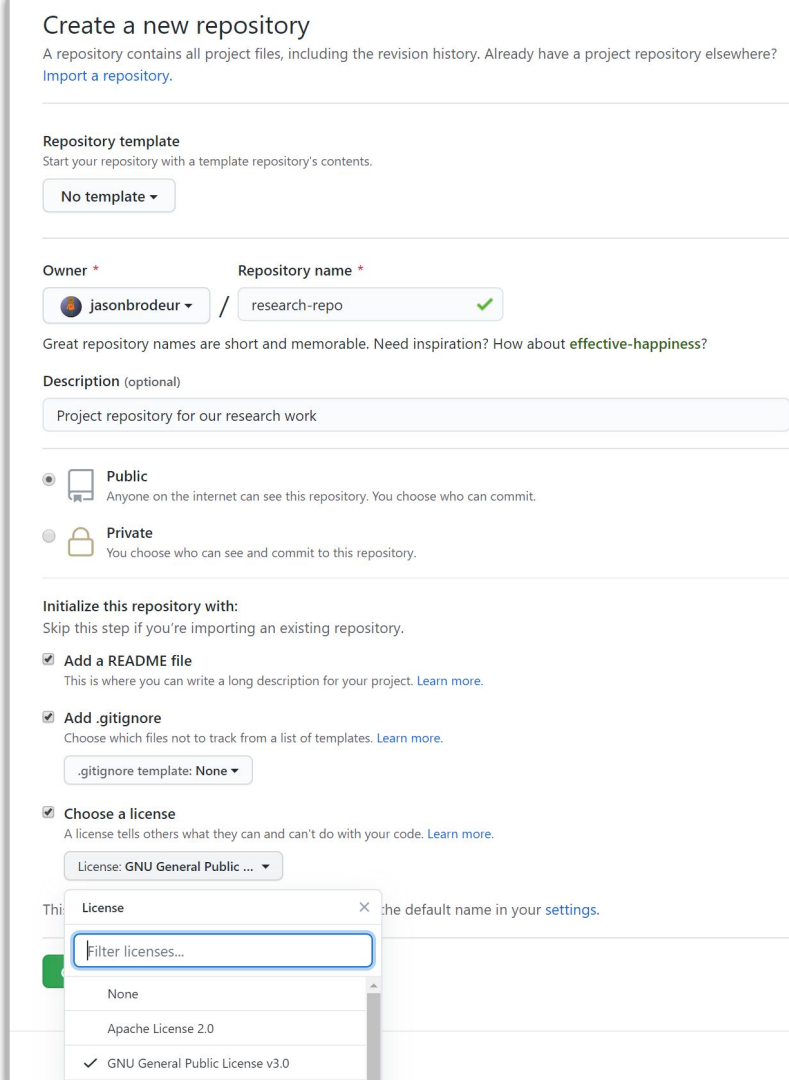
- Create name and description
- Set visibility
- Add README, .gitignore, license



🔁 upload, edit etc.

🔁 add + commit

🔁 branch + merge

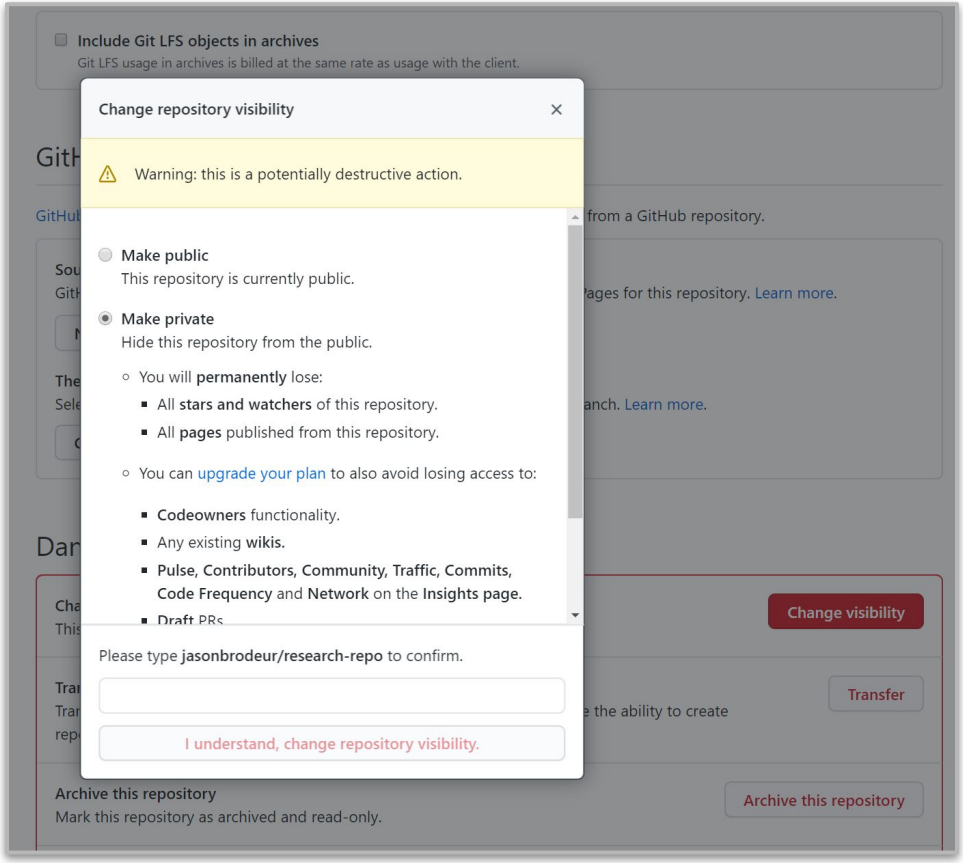


Managing repository visibility

Set at creation or anytime in >Settings

Options:

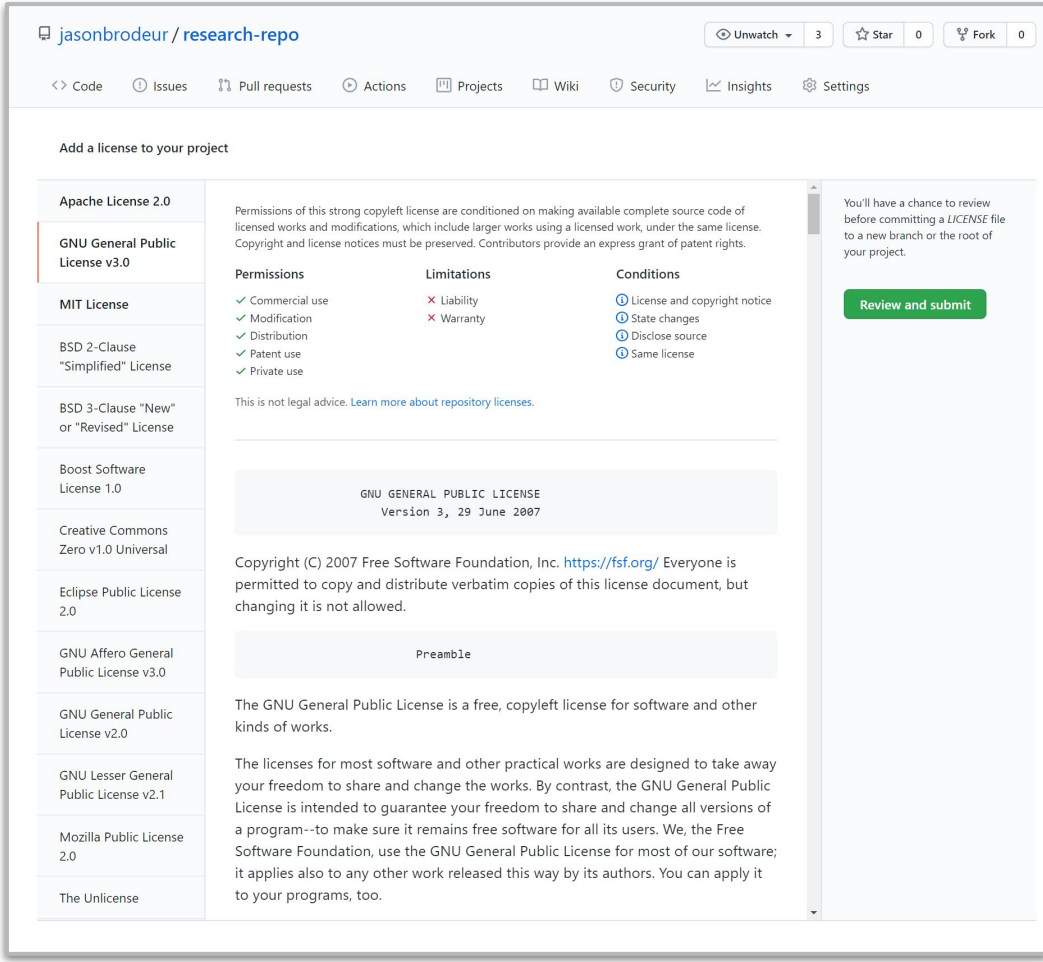
- Public to everyone
- Private to collaborators / teams
  - Added private features with upgrade to paid account / organization



Adding a license

Created in the LICENSE file in the top-level of the repository

Built-in license selector (or add your own)



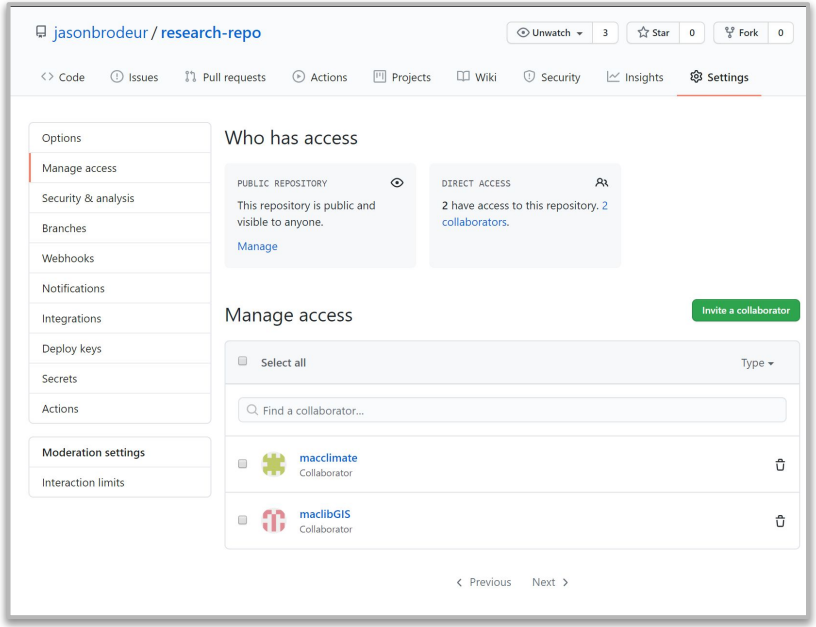
Managing collaborators on a personal repository

Collaborators can be added to both public and private repositories

For ***personal repositories*** (owned by a user), collaborators have only one set of privileges

- Can push, pull (read), and fork
- Manage pull requests, wikis, releases, etc.

More granular permissions are available for repositories owned by ***organizations***

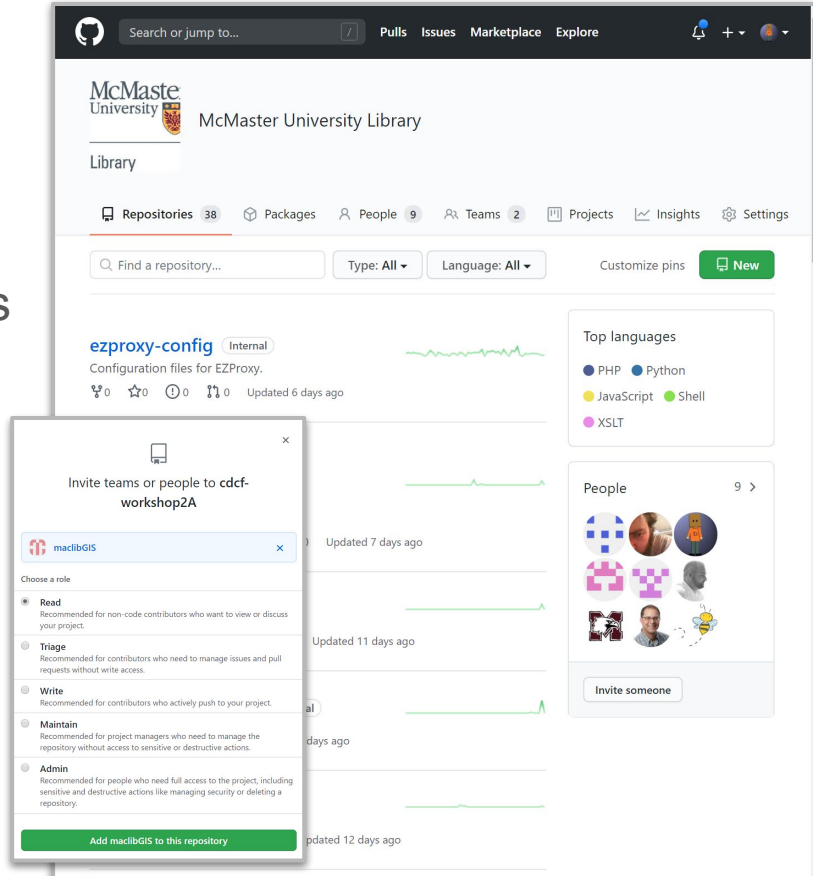


Managing access with ***organizations*** and ***teams***

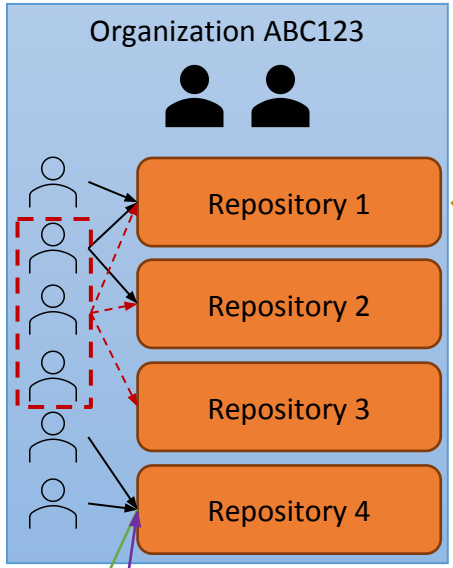
Organizations are shared accounts for projects

Benefits:

- Shared ownership of (unlimited) repositories
- Top-level management of repositories
- Unlimited membership
- Range of roles and permissions
- Nested ***teams*** with cascading access
- Two-factor authentication
- Are free to create



Managing access with ***organizations*** and ***teams***



Granular permissions for *members*, *teams*, public (view), and outside collaborators,

Organization administrators can:

- Add users to the organization; manage members' roles
- Manage access using nested teams with cascading access permissions
- View members' two-factor authentication (2FA) status and require 2FA

Members of the public can view repositories made public by organization administrators

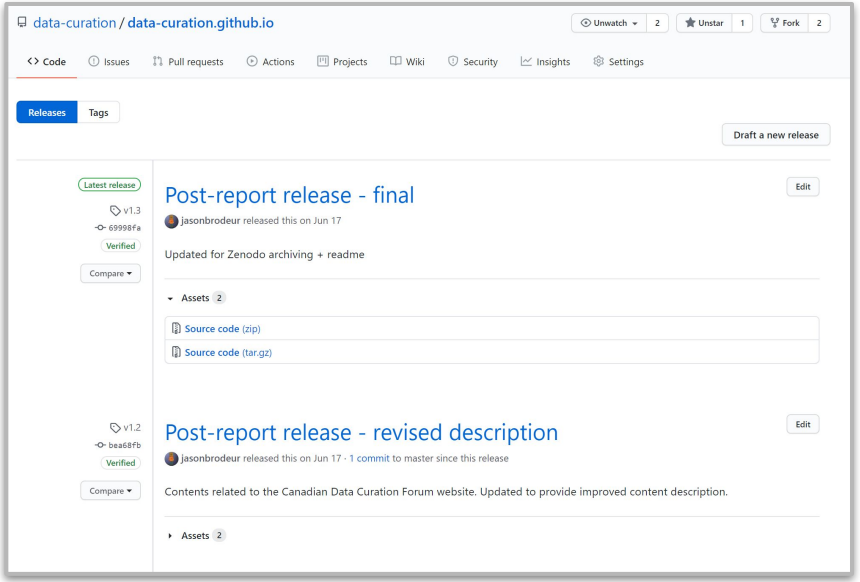
Outside collaborators can be added to repositories in an organization (without being added to the organization itself)

Packaging and ***releasing*** repositories

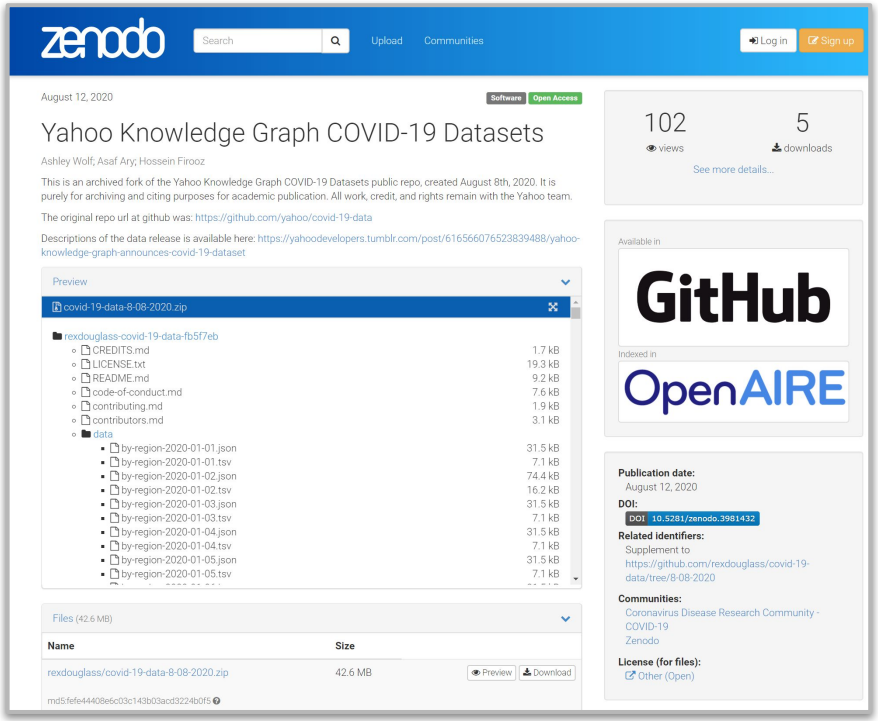
A release is a tagged snapshot of a repository for deploying a discrete version to broader audiences.

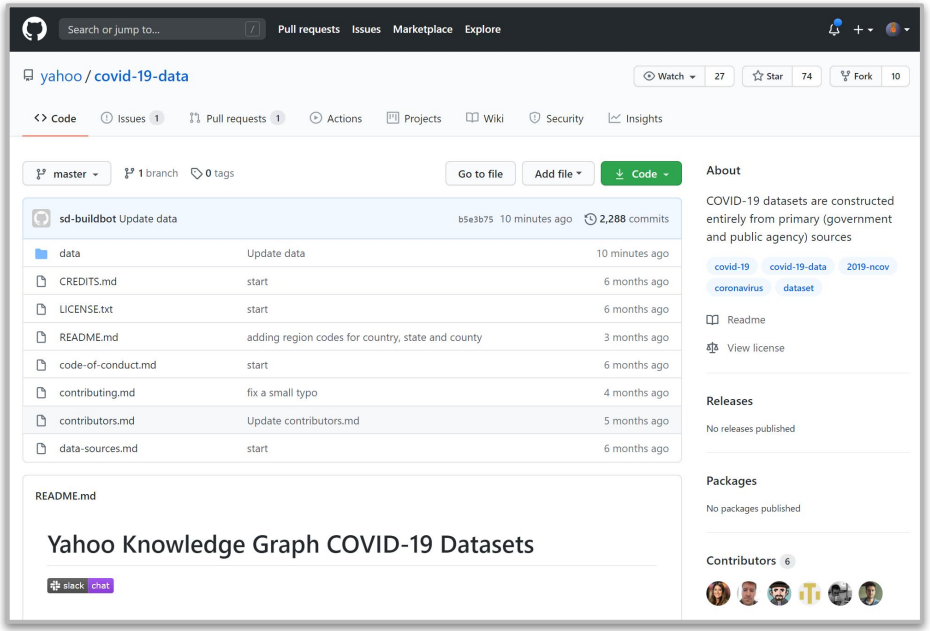
Releases are used when:

- Deploying software packages
- Packaging as supplemental materials (e.g. supporting a publication)
- Archiving in data / code repositories



An Example - Archiving in Zenodo





Ashley Wolf, Asaf Ary, & Hossein Firooz. (2020, August 12). Yahoo Knowledge Graph COVID-19 Datasets (Version 8-08-2020). Zenodo. http://doi.org/10.5281/zenodo.3981432

Main repo: https://github.com/yahoo/covid-19-data
Release (forked): https://github.com/rexdouglass/covid-19-data/tree/8-08-2020

Communication & collaboration tools - Pull requests

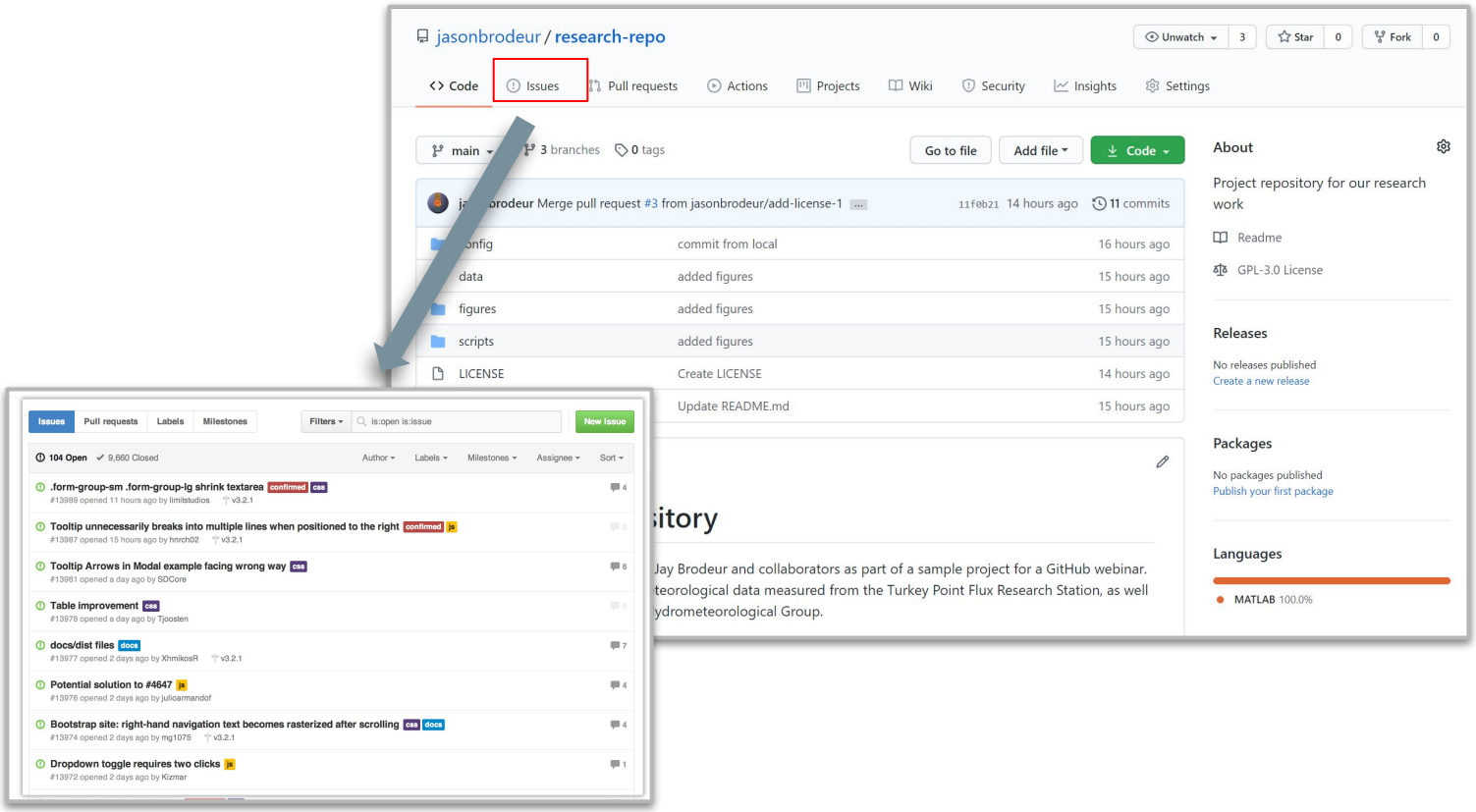
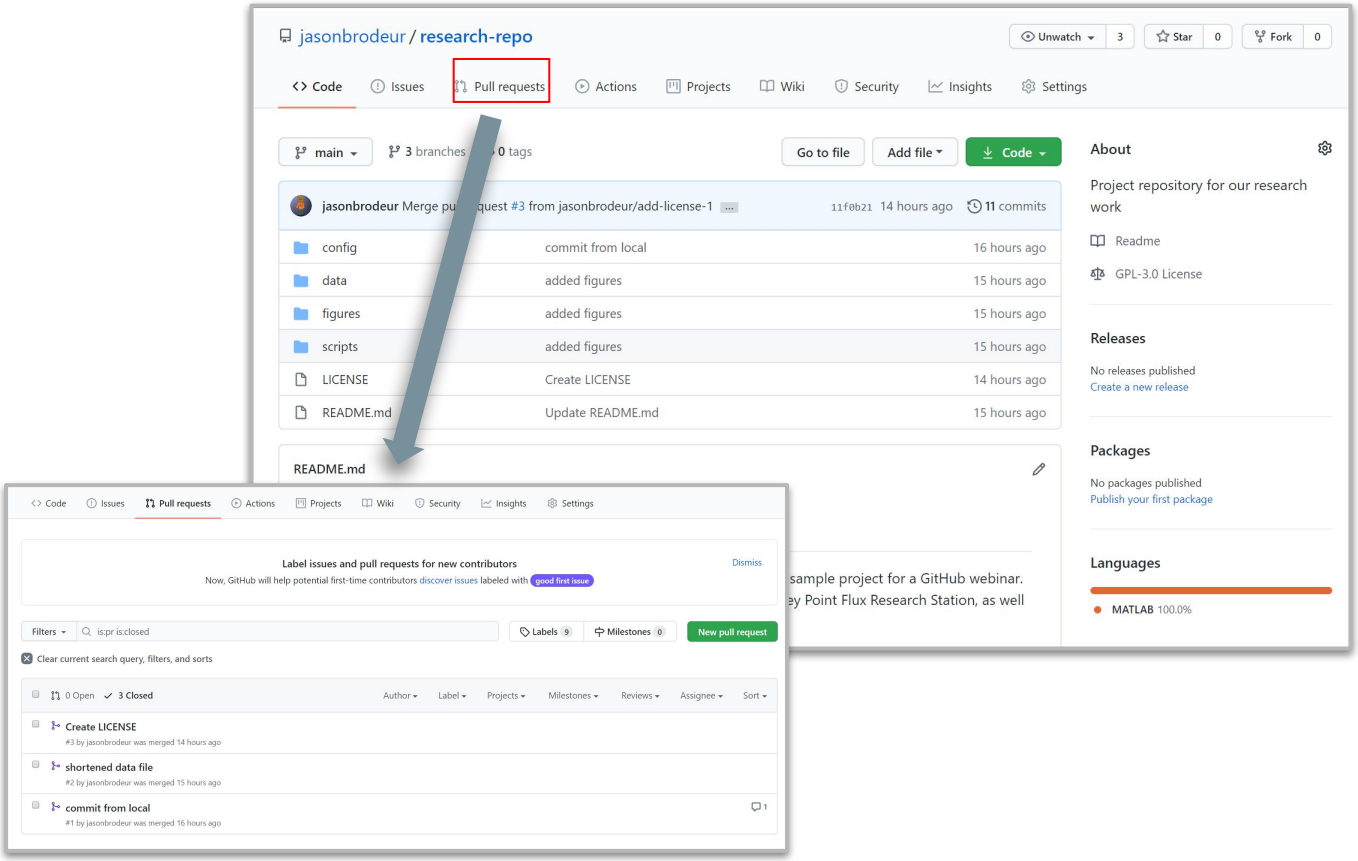


Image credit: [GitHub guides]

Communication & collaboration tools - Pull requests



Communication & collaboration tools - Project Boards

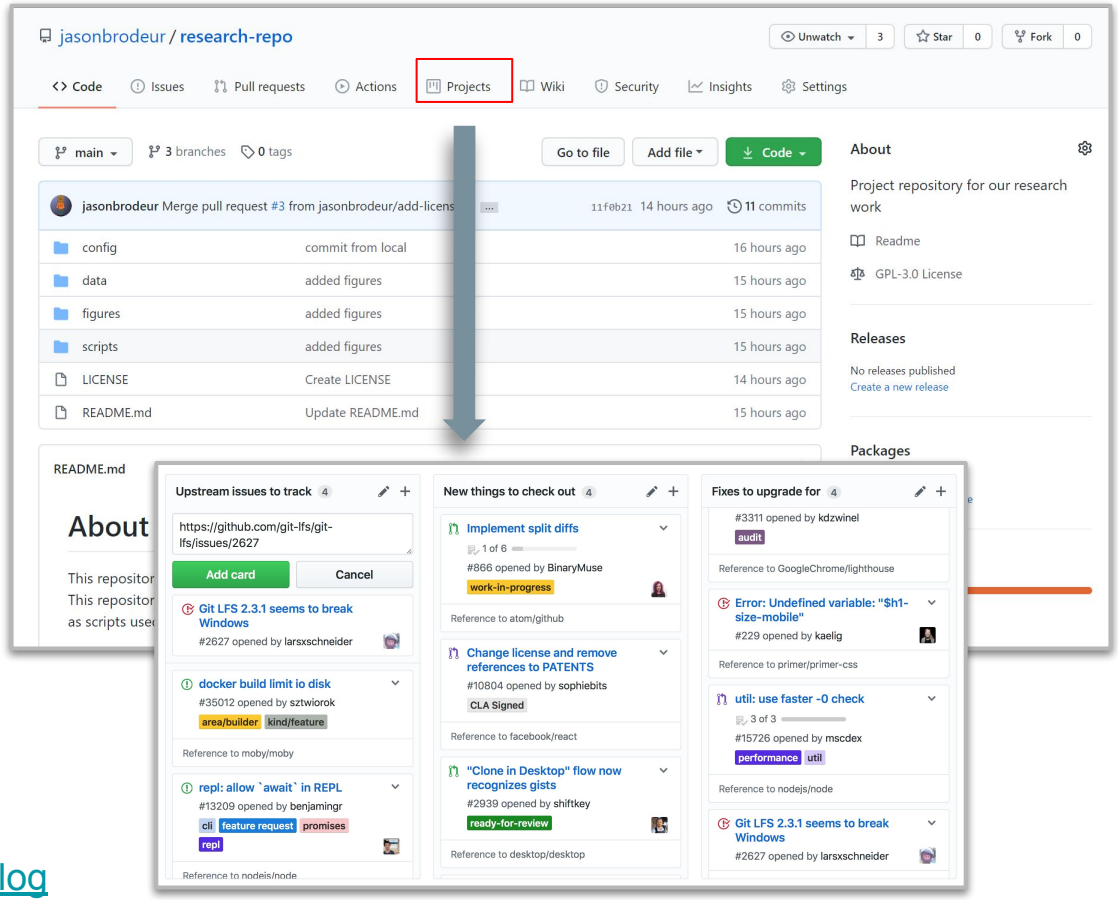


Image credit: [GitHub blog]

Communication & collaboration tools - Wikis

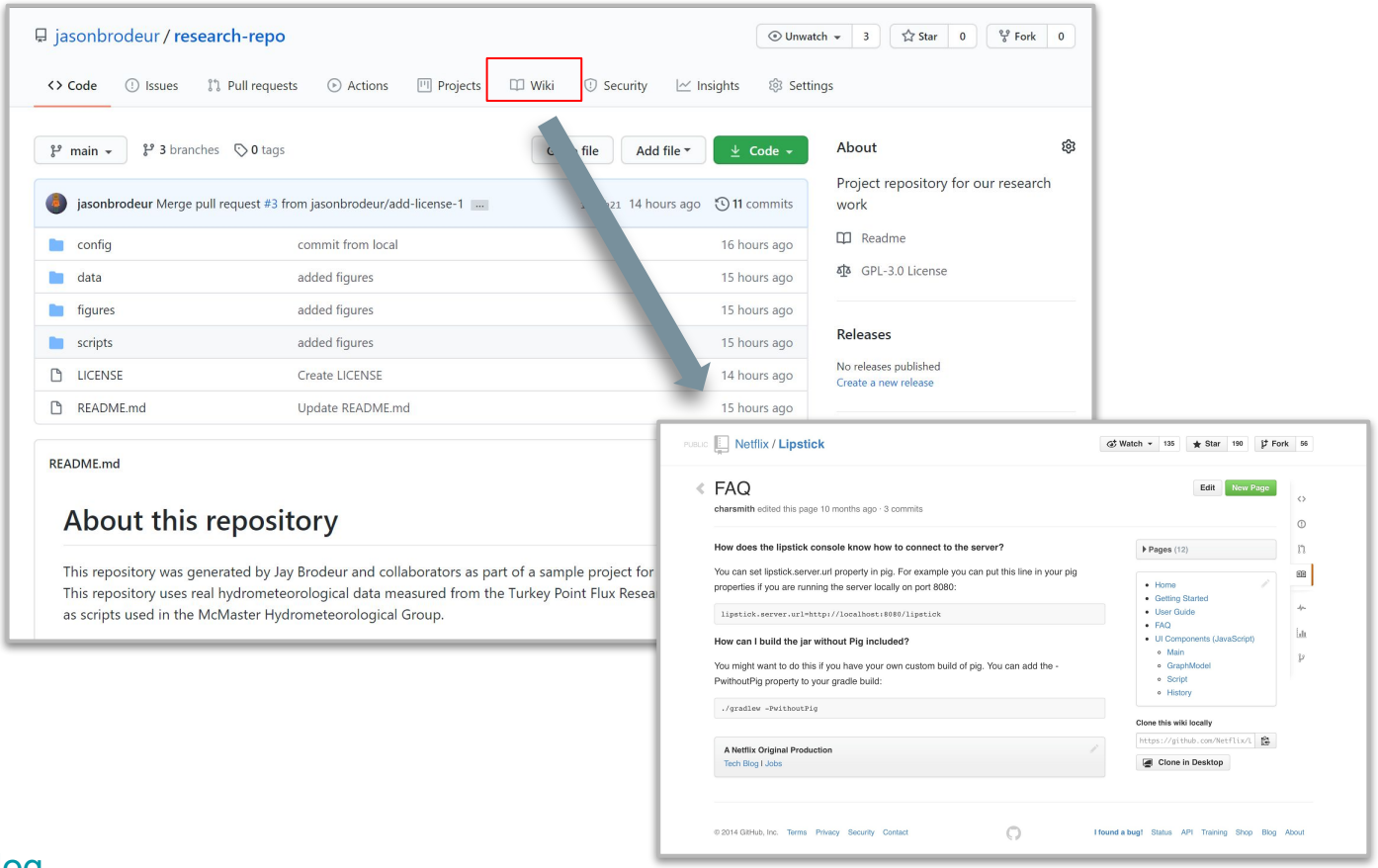
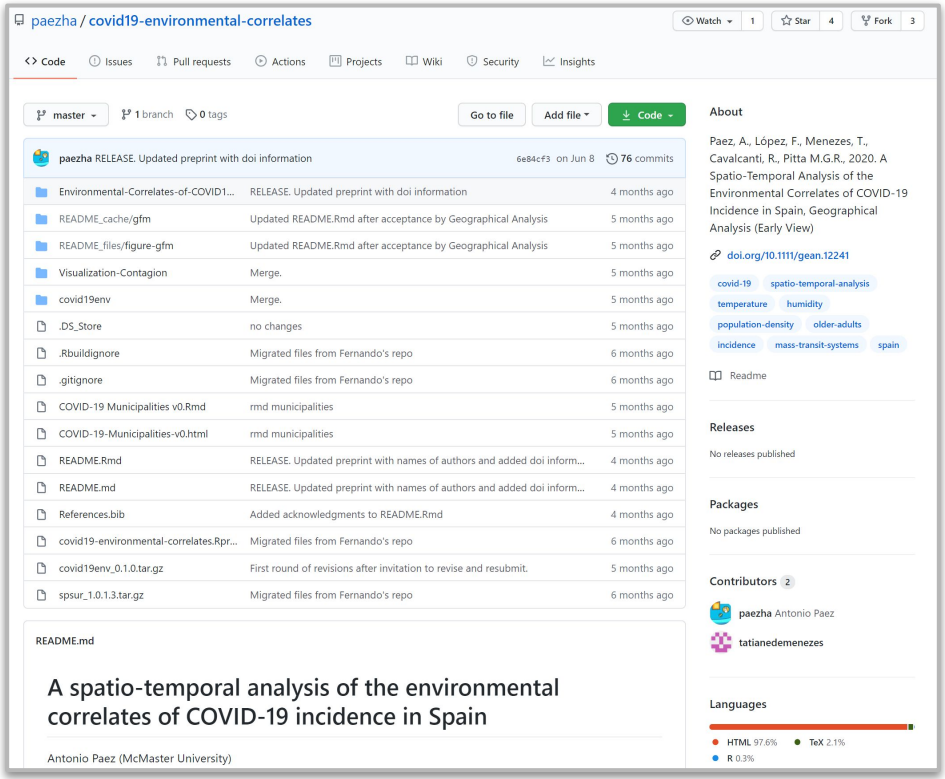


Image credit: [GitHub blog]

Markup and presentation

GitHub as the medium

Example: GitHub repository as a preprint



Paez, A., López, F., Menezes, T., Cavalcanti, R., Pitta M.G.R., 2020. A Spatio-Temporal Analysis of the Environmental Correlates of COVID-19 Incidence in Spain, Geographical Analysis

https://github.com/paezha/covid19-environmental-correlates

Markdown in GitHub

- A very lightweight markup language used by GitHub (and Reddit, and Trello)
- Improves formatting while leaving the plain document readable.
- Mostly just regular text with a few non-alphabetic characters thrown in

```
1  # This is a heading
2  I am making **these words** bold
3
4  Here is a list
5  * Of things
6  * And stuff
```

Markdown

This is a heading

I am making **these words** bold

Here is a list

- Of things
- And stuff

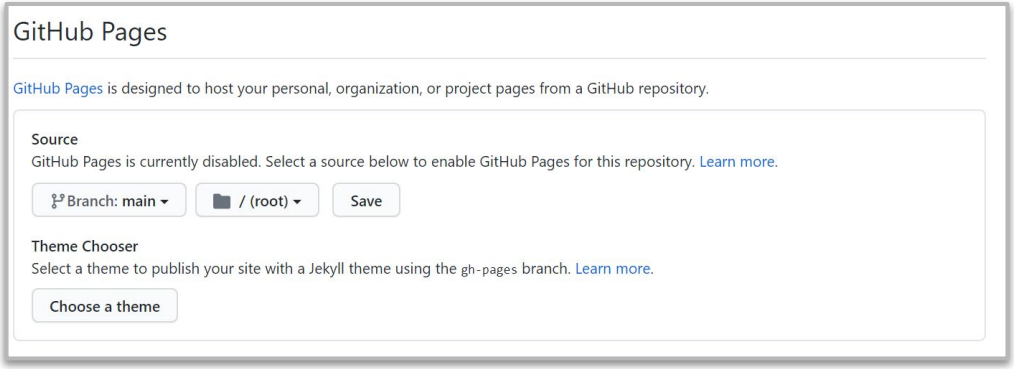
Rendered text

Learn more: https://guides.github.com/features/mastering-markdown/

GitHub Pages

"**GitHub Pages** is a static site hosting service that takes HTML, CSS, and JavaScript files straight from a repository on GitHub, optionally runs the files through a build process, and publishes a website."

GitHub pages also allows you to create webpages from markdown files, using a built-in software called jekyll.



Administrative Tools

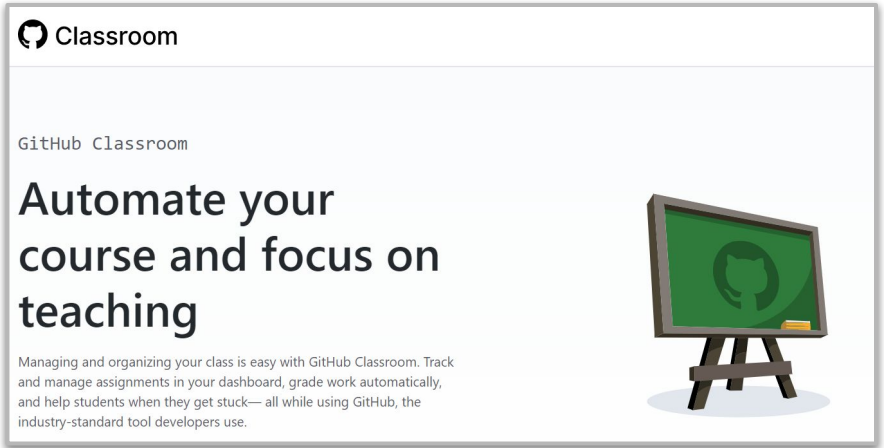
To support research & teaching

GitHub Classroom

Classroom

Tools to use GitHub for course management

- Manage students in an organization
- Create assignment repositories from templates
- Granular access management of submitted materials
- Automated management & grading

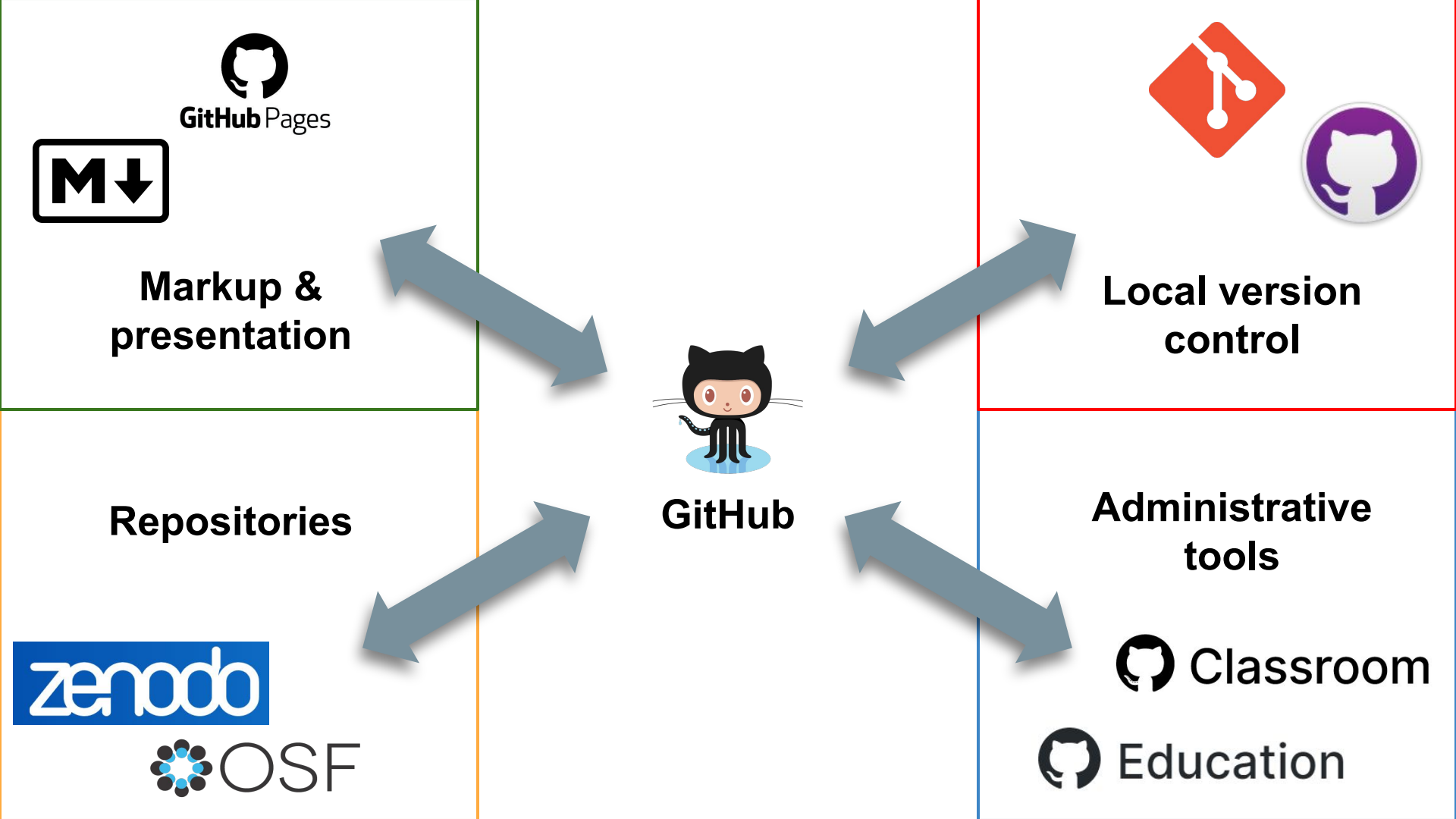


GitHub Campus Program

Provides Institutional-level access to **GitHub Enterprise Cloud**, which:

- Helps institutions manage collaboration and access (including SAML single sign on and 2FA)
- Allows unlimited organizations
- Access to **GitHub Enterprise Support**
- Offers premium features (such as continuous integration)
- Provides administrators a single point of visibility and management.

education.github.com/schools


GitHub Pages
Markup & presentation
Local version control
Repositories
GitHub
Administrative tools
zenodo
OSF
Classroom
Education

And again …

One more question

I would be interested in learning more about...

McMaster University LIBRARY

How to cherry-pick select elements only from pull requests.

GitHub classroom

Actual implementation and training across a real research group for collaboration.

git on its own

command line vs. desktop github

Needed to be hands on

version control with GitHub

how to start using it ? for someone working in the domain of info literacy how do I use it?

moving repositories to an organization

Best practices for sharing/pulling/forking/pushing/etc.

More examples of team collaboration and the logistics eg. merge requests, plus best practices?

maybe how linking content to dissertations thesis or master

more depth each area to support actual implementation to support my research work for HQP

version control and developing software

Sample practical examples of pull, merge, branch, etc.

Data-driven model

Hands on would be great!

githob

I would be interested in learning more about...

McMaster University LIBRARY

A practical look at collaboration

Hands on would be good - specific topics - markdown, or gitpages etc.

Integration with Jira and pull requests

"Case studies" - How academic research groups are currently using GitHub, and in particular what did GitHub replace for them? e.g. "we were using dropbox and just hack and bashing", etc. Also how the wiki feature works!

Seconding more depth in each area. Mini-tutorials!

upvoting case studies suggestion!

Hand on using github

Learn more

The Git Pro book: https://git-scm.com/book/en/v2

Introduction to GitHub: https://lab.github.com/githubtraining/introduction-to-github

GitHub Guides: https://guides.github.com/

UBC Library Research Commons - Intro to git and GitHub: https://ubc-library-rc.github.io/intro-git/

Getting started with GitHub Pages: https://guides.github.com/features/pages/

GitHub Classroom: https://classroom.github.com/

GitHub Campus Program: https://education.github.com/schools