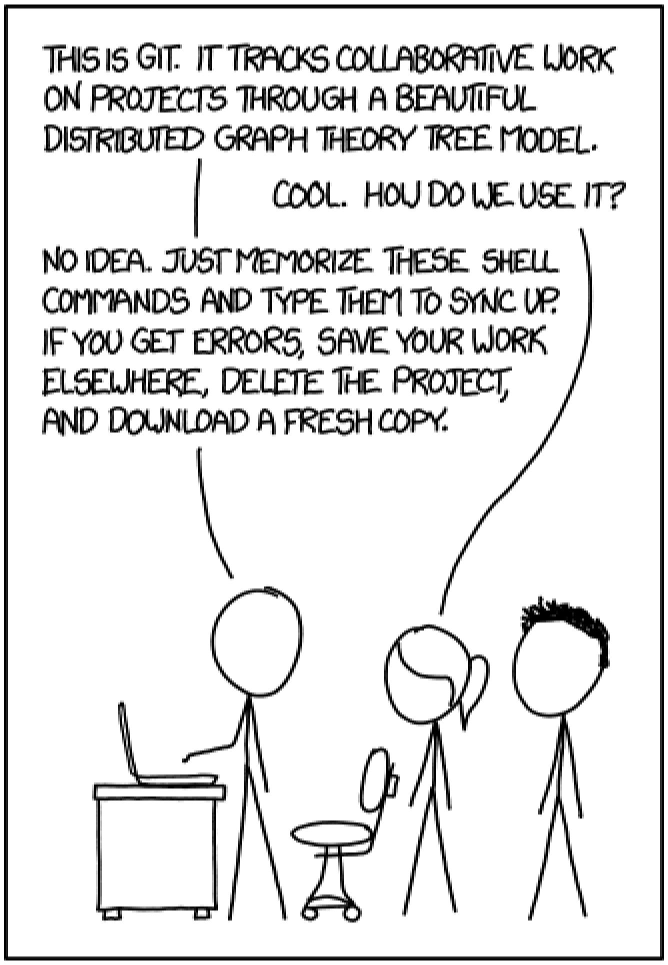
Tools: Git and GitHub


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.

https://xkcd.com/1597/

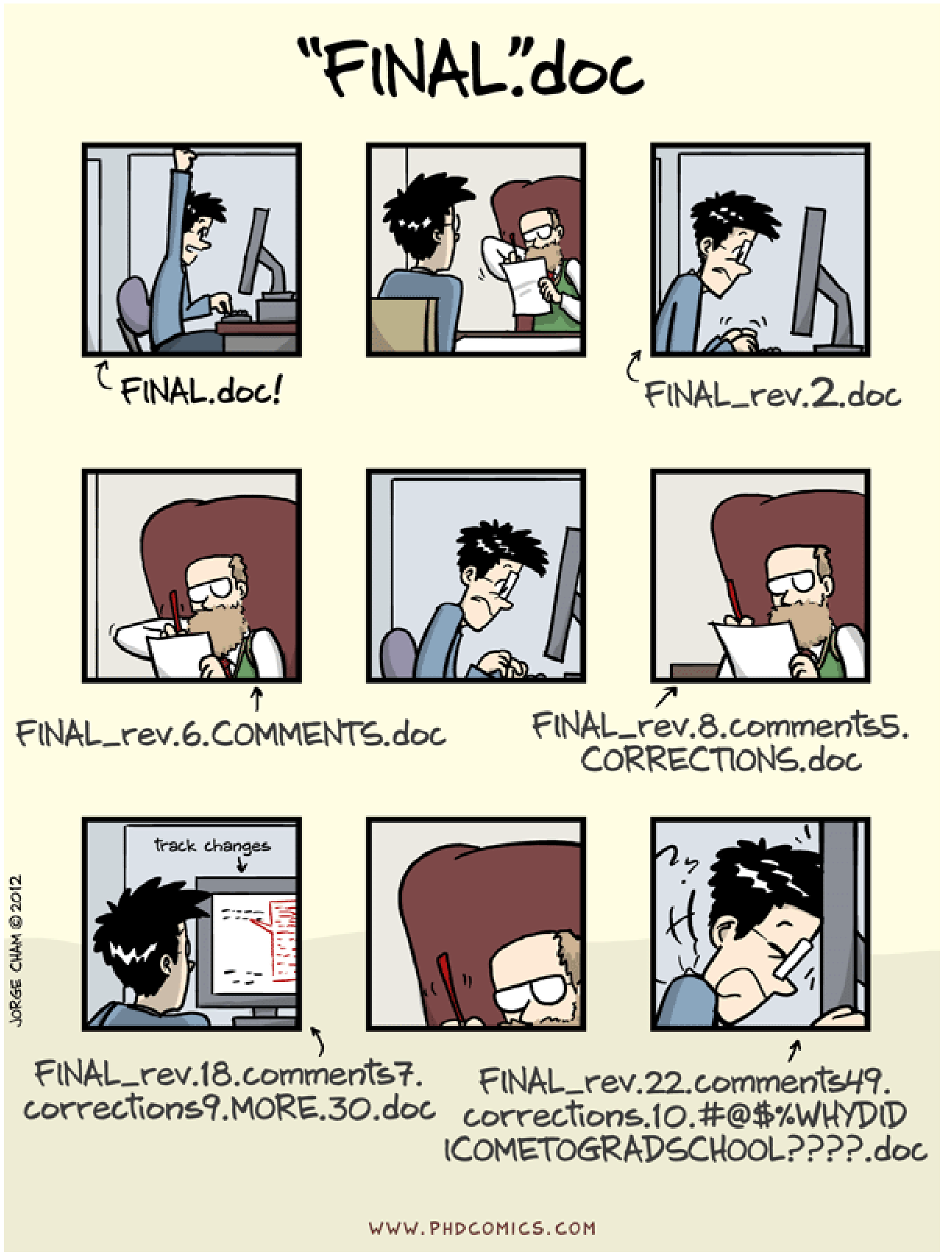
Note: Thanks to http://happygitwithr.com for inspiration and material.

0.1 Definition/background

Git is a version control system that was created to help developers manage collaborative software projects. Git tracks the evolution of a set of files, called a **repository** or **repo**.

This helps us

- *merge* conflicts that arrise from collaboration
- *rollback* to previous versions of files as necessary
- store *master* versions of files, no more `paper_final_final_I_really_mean_it.docx`


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDIDICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

(http://phdcomics.com/comics/archive.php?comicid=1531)

0.2 Terminology

- **Repository:** The most basic element of git, imagine as a project's folder. A repository contains all of the project files, and stores each file's revision history. Can be either public or private.
- **Clone:** A copy of a repository that lives on your computer instead of on a website's server somewhere, or the act of making that copy.
- **Pull:** When you are fetching in changes and merging them.
- **Commit:** An individual change to a file (or set of files). Every time you save it creates a unique ID that allows you to keep record of what changes were made when and by who.
- **Push:** Sending your committed changes to a remote repository such as GitHub.com.
- **Fork:** A fork is a personal copy of another user's repository that lives on your account. Forks allow you to freely make changes to a project without affecting the original.
- **Pull Request:** Proposed changes to a repository submitted by a user and accepted or rejected by a repository's collaborators.
- **Issue:** Issues are suggested improvements, tasks or questions related to the repository.
- **Remote:** This is the version of something that is hosted on a server, most likely GitHub.com. It can be connected to local clones so that changes can be synced.

From https://help.github.com/articles/github-glossary/.

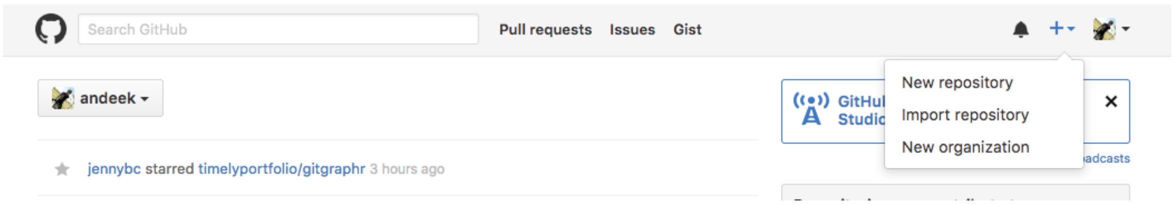
0.3 GitHub

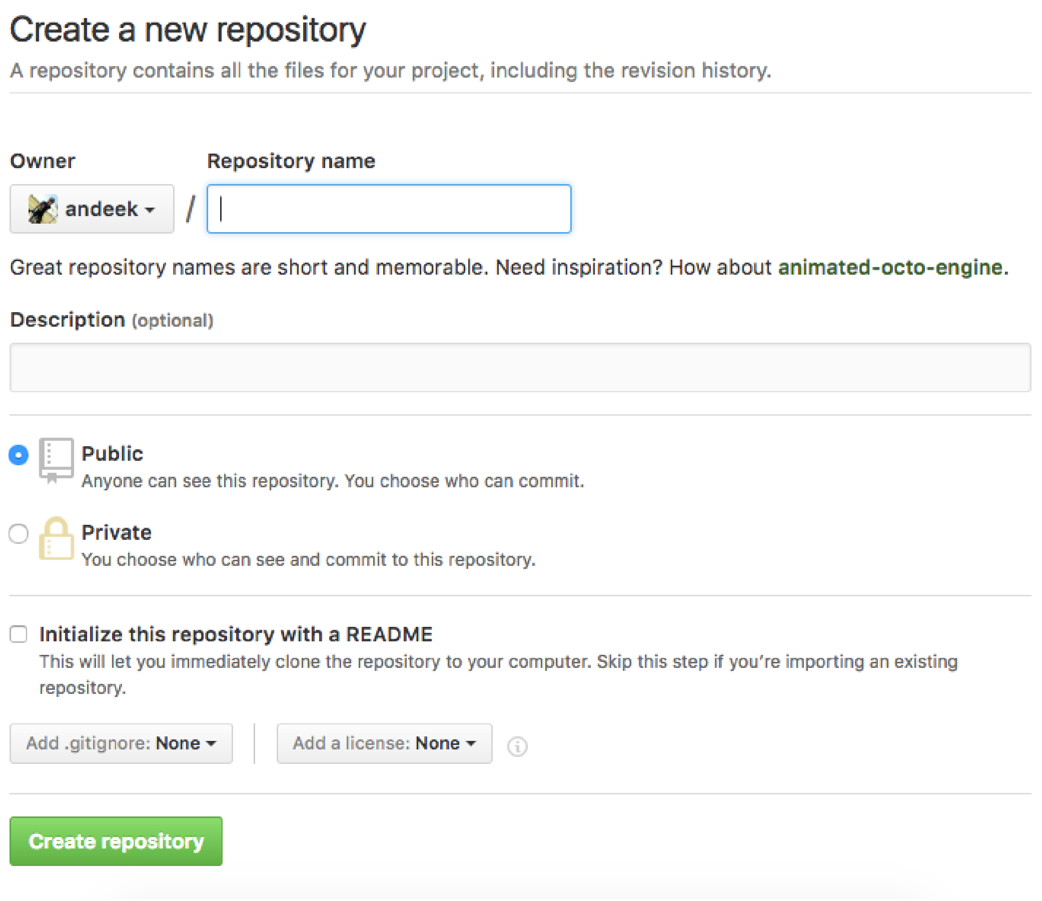
There are many hosting services for remote repositories (GitHub, Bitbucket, GitLab, etc.). We will use GitHub in this class, but the ideas carry over to the other services.

By default, all materials on GitHub are **public**. This is *good* because you are getting your work out there and contributing to the open source community!

If you need **private** repos, checkout GitHub for Education (https://education.github.com/) - free private repos for students/postdocs/professors.

0.4 Creating Repos

1. 

2. 

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: andeek / Repository name:

Great repository names are short and memorable. Need inspiration? How about **animated-octo-engine**.

Description (optional)

[✓ Public] Anyone can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

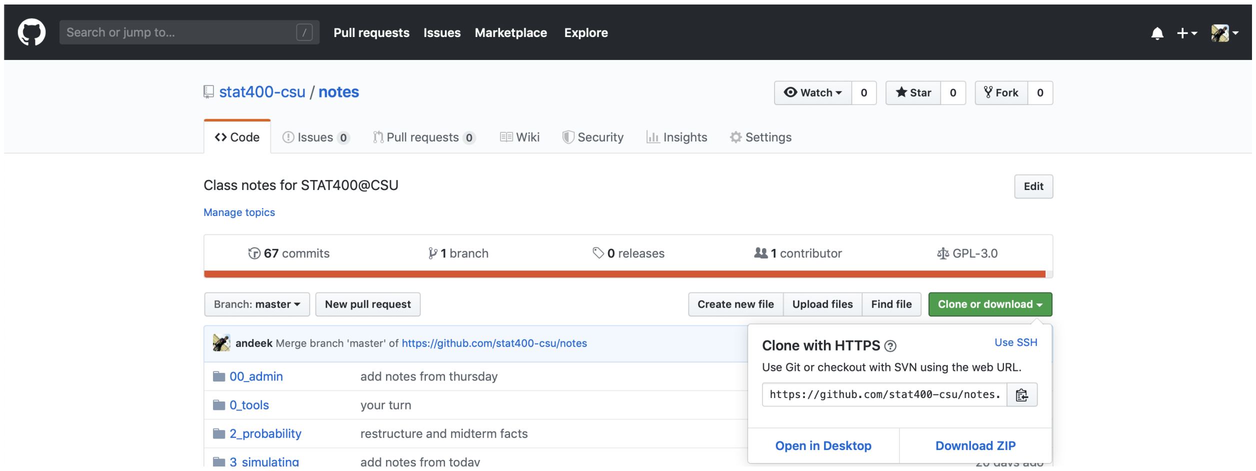
Create repository

Initialize readme (yes), .gitignore (R usually), license (e.g. GPL 3)

Your Turn

1. Create a GitHub account
   - github.com (http://github.com)
   - Consider your username, want to be identifiable, professional and probably include your actual name. Do you have other handles? Twitter?
   - Don’t worry about paying for a plan now, stick with the free one.
2. Create a `hello-world` repo

Cloning a repo –



stat400-csu / notes

Class notes for STAT400@CSU

67 commits | 1 branch | 0 releases | 1 contributor | GPL-3.0

andeek Merge branch 'master' of https://github.com/stat400-csu/notes

00_admin	add notes from thursday
0_tools	your turn
2_probability	restructure and midterm facts
3_simulating	add notes from today

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

```
https://github.com/stat400-csu/notes.
```

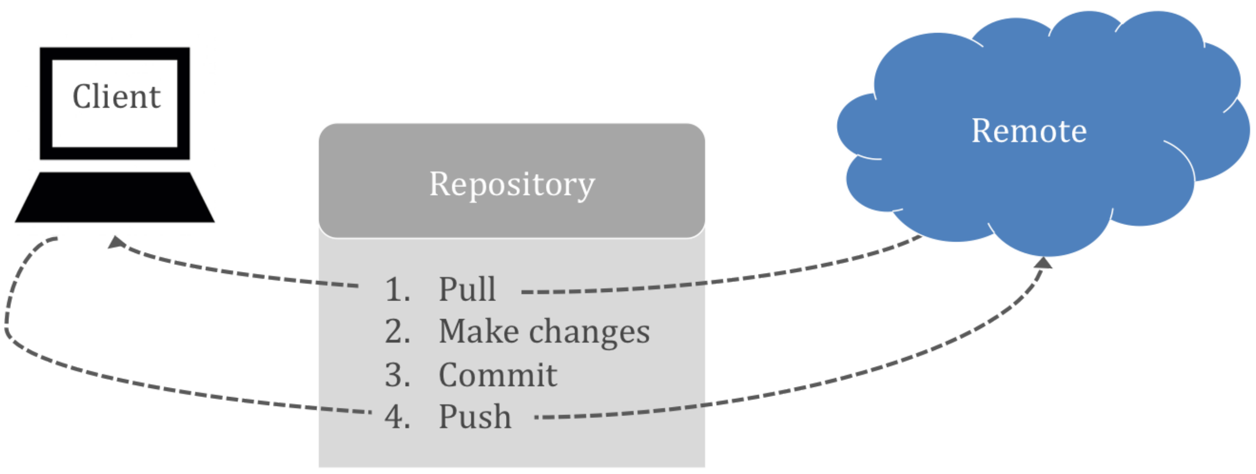
From scratch:

1. Create the repo on the GitHub website
2. Clone the repo
3. Start working
4. Add files, commit, push, etc.

From existing folder:

1. Create the repo on the GitHub website
2. Clone the repo
3. Copy existing work into local folder
4. Add files, commit, push, etc.

0.5 Pushing and pulling, a tug of war



Important: remember to pull before you start working to get the most up to date changes from your collaborators (or your past self) before making local changes!

0.6 When should I commit?

Think of commits as a checkpoint in a video game. This is a point in time when you want to save your status so that you can come back to it later if need be.

> Commits are like voting. I like to do it early and often.

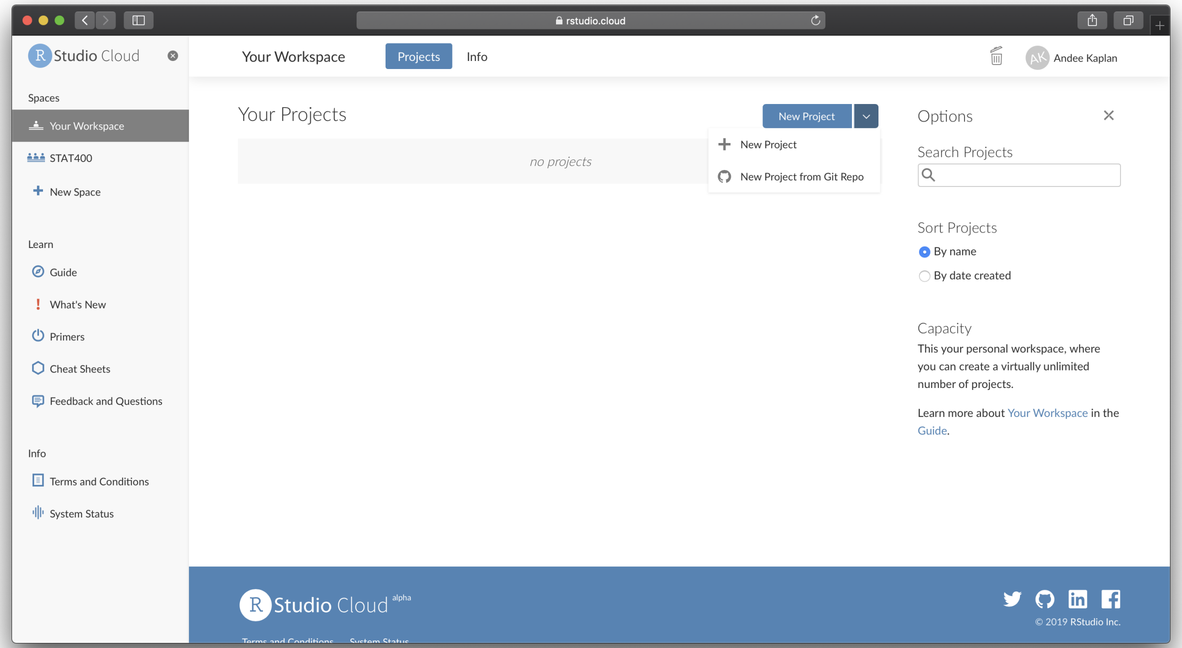
- *Me, right now*

0.7 Blow it up

Sometimes your local repo gets borked. That's OK. There are ways that we can work really hard and fix them, but sometimes you just want to stash your files somewhere and re-clone from your centralized repository.

0.8 Git with RStudio

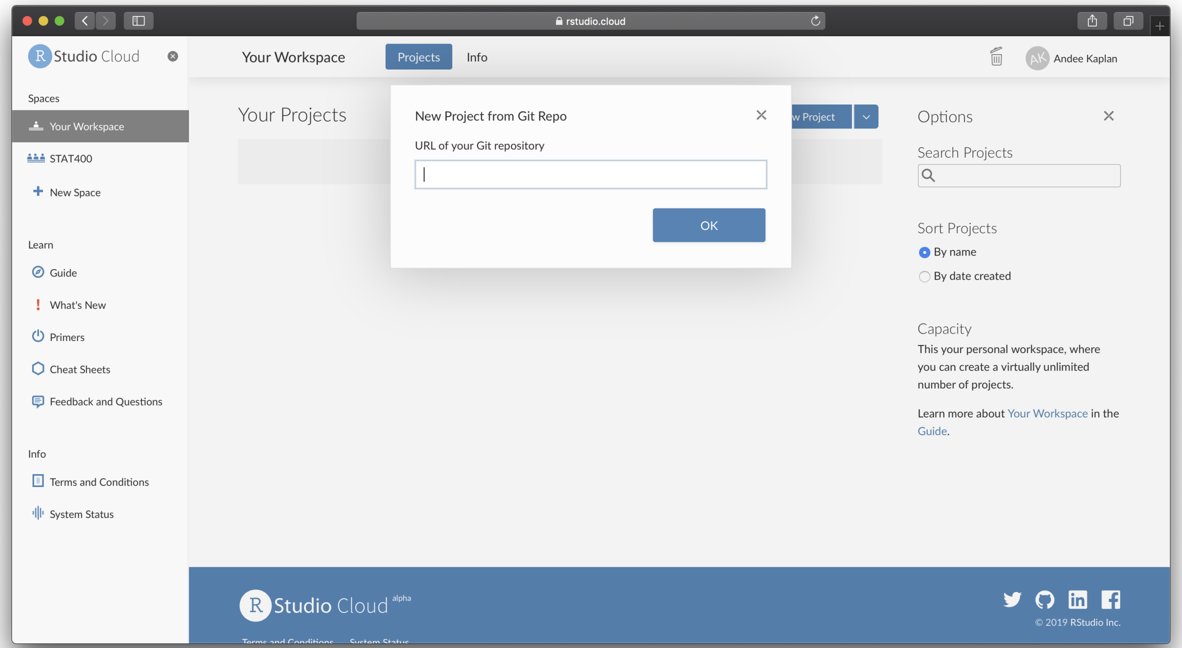
Rstudio.cloud allows you to make projects based on GitHub repos.



Sort Projects: [✓ By name] By date created

This your personal workspace, where you can create a virtually unlimited number of projects.

Learn more about Your Workspace in the Guide.

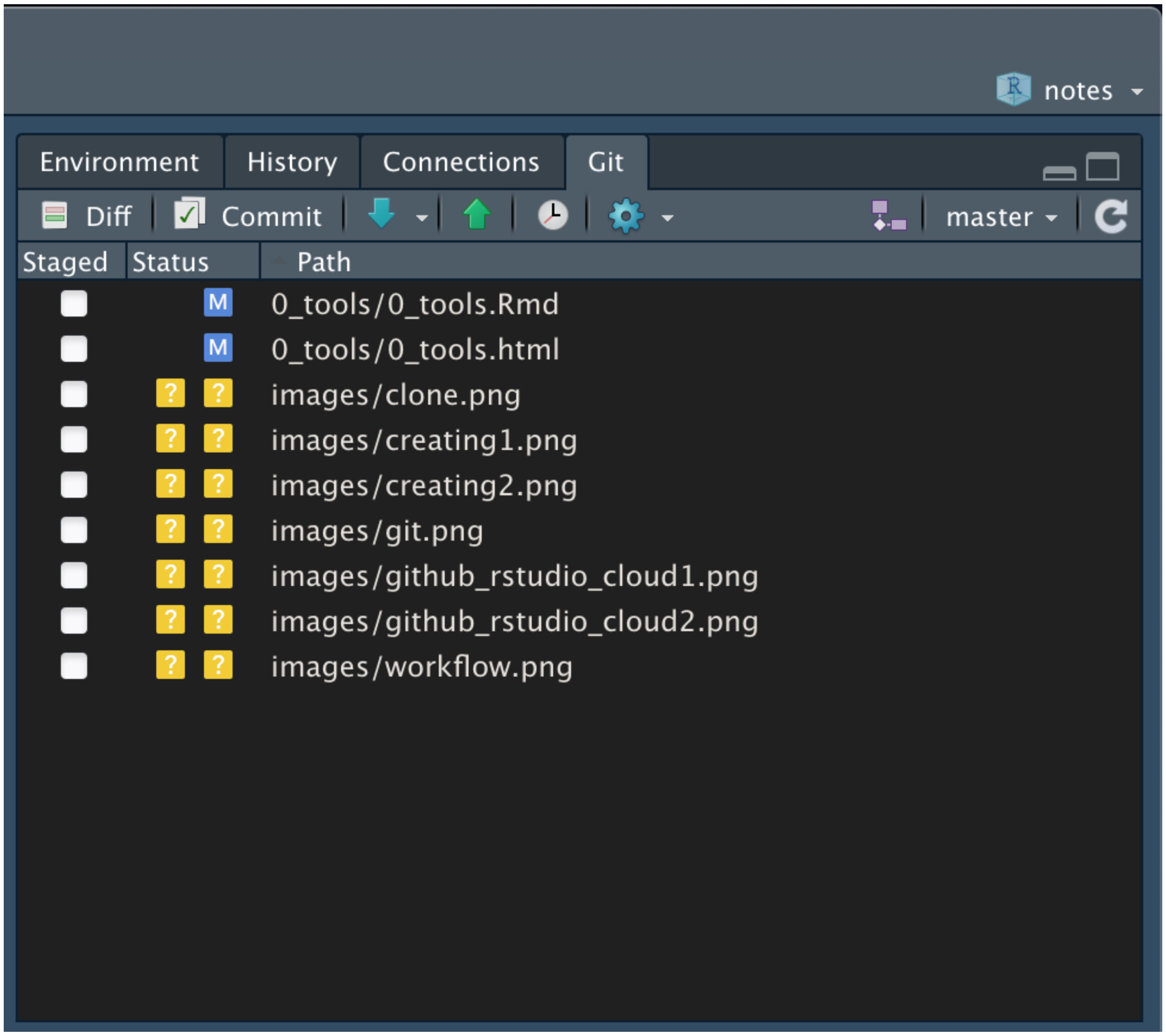


Sort Projects: [✓ By name] By date created

This your personal workspace, where you can create a virtually unlimited number of projects.

Learn more about Your Workspace in the Guide.

Local RStudio works much the same way, with the ability to push/pull from a local project to a GitHub repo.



Staged	Status	Path
☐	M	0_tools/0_tools.Rmd
☐	M	0_tools/0_tools.html
☐	? ?	images/clone.png
☐	? ?	images/creating1.png
☐	? ?	images/creating2.png
☐	? ?	images/git.png
☐	? ?	images/github_rstudio_cloud1.png
☐	? ?	images/github_rstudio_cloud2.png
☐	? ?	images/workflow.png

By letting us

1. Select files to commit.
2. Commit
3. Push/Pull

0.9 Collaboration

In this class, we will have a collaborative project. It will be hosted on GitHub and part of your grade will be how well you collaborate through the use of GitHub. For this, we will need to have project repos that everyone can push to!

GitHub Repo site (https://github.com/username/reponame) > Settings > Collaborators & Teams > Collaborators > Add collaborator

Your collaborators will have a lot of power in your repo, so choose wisely! They can change anything in the repo and add their own files. They can also delete your files! The only thing they can't do is delete the repo, add collaborators, or change other settings for the repo.

0.10 Installation help

We are not covering installation on your personal computers for this class. If you would like to work through it on your own, here is an *excellent* guide:
http://happygitwithr.com/installation-pain.html

Feel free to come to office hours or setup individual time with us if you need help.

Your Turn

1. Edit the README file in your `hello-world` repo
2. Commit and push changes
3. Check out your commit history!
4. Add me (`andeek`) as a collaborator to your hello-world repo