

git and GitHub!

Hopefully cutting through the
fogginess

Daniel Anderson

Week 1, Class 2

Agenda

- *git* vs GitHub vs GitKraken
- Two ways to create a repo
- Adding collaborators
- Branching
- Pull requests
- Forking
- Stashing
- General recommendations

Setup

I'm assuming you're all setup. If not, please do the following

- Download and install *git*
- Register for a GitHub account
- Download and install GitKraken

git vs GitHub

git and GitHub are somewhat analogous to R and RStudio

- R is the "engine", RStudio is the IDE
- *git* is the "engine", GitHub is the web-based hosting service

It's somewhat easy to conflate these things, but it's important to keep them separate... Why?

The background is a solid black field. It is decorated with several abstract, colorful elements: thin lines in blue, red, yellow, and purple that form various geometric shapes like circles, squares, and zig-zags. Some of these lines end in small open circles of the same color. Scattered throughout the background are small white plus signs (+).

GitKraken

A visual interface for interacting with *git*

Why GitKraken

You could argue it would be better to focus on the command line interface.

My experience

Most people have an easier time grasping the *concepts* of *git* (like branching) when there's a visual interface

GitKraken allows you to do 95%+ of the work you need to do without moving to the CLI

Why GitKraken

If you decide to go with CLI eventually, that's totally fine. You'll probably need to for some operations. But as Jenny Bryan notes

■ No one is giving out Git Nerd merit badges

So use the tools that make you most effective

Big picture

Why do we use *git*?

We use *git* to:

- Contain version control and track changes to documents over time
- Increase transparency and reproducibility in process
- Collaborate with others efficiently and effectively
- Share our work with others

Warning

This will be painful at times – you've probably already had painful experiences. I believe the payoff is worth the effort.

The vocabulary is probably half the battle



GitHub

Creating repositories: Starting with
an empty project

GitHub first

If you are starting a new project and you're going to use version control, I suggest going GitHub first

- Create a new empty repo on Github
- Clone the repo locally
- Start your project (i.e., RStudio Projects)
- Commit your initial changes
- Push the changes to the remote



GitHub

Creating repositories: Starting with
an existing project

Project first

If you already have an existing project, you need to:

- Initialize the repo locally
- Create an empty repo on GitHub
- Connect your local repo to the remote

Vocabulary pop quiz

What do the following mean?

- Stage
- Commit
- Push
- Pull
- Clone
- Branch
- Fork

Merge conflicts

- Merge conflicts only occur if you try to edit *the same line* of an out-of-date file
- Let's create and resolve one with the README file.

Branching

Super powerful approach to collaborating with *git*. Essentially create a copy of the repo at a given point.

- Create and *checkout* a new branch.
- Make changes and commits *on the branch*, which frees you up to play and experiment without fear of "breaking" the main branch
- Push the branch to the remote
- When you're ready, submit a *pull request* to merge the branch with the main branch
- Note – any changes on the branch *will not be visible* if you don't have that branch checked out

Pull requests

Submit a PR for a branch when you're ready

Consider tagging others to review the changes before they are merged in

Link the PR to any relevant Issues.

Consider adding labels (e.g., bug fix, enhancement)

Forking

Similar process to branching, with a few major differences:

- You don't have to be (and shouldn't be) a direct collaborator
- You copy the entire repo *to your account*
 - You can then make whatever changes you want and push the changes to your fork
- If you want your changes implemented, you *must* submit a pull request (you can't contribute to the repo directly unless you're a collaborator) and *compare across forks*

Stashing

Last topic for the day

When working across multiple branches, you might find yourself occasionally making edits on the wrong branch

You can't commit those changes, but you don't want to lose them either

Stashing allows you to basically create a temporary branch, store the changes there, then "pop" them on the correct branch

The screenshot displays the GitHub Desktop application interface. At the top, the repository is identified as 'example-existing-project-sdsr' on the 'main' branch. The interface is divided into several panels:

- Left Sidebar:** Contains a file explorer showing 'LOCAL' and 'REMOTE' files, and a list of issue trackers (GitHub, GitHub Enterprise, GitKraken Boards, GitLab).
- Main Panel:** Shows a commit graph for the 'main' branch. The graph illustrates a sequence of commits, including 'Initial commit', 'project setup & initial exploration', 'ignored the .DS_Store file', 'Update README.md', 'Updated README locally', 'Fixed merge conflict - took remote version', 'styled plot', and 'created report template'. A 'Merge pull request #1 from datalorax/exa...' is also visible.
- Right Sidebar:** Displays 'Unstaged Files (2)' with files 'report/example-rmd.html' and 'report/example-rmd.Rmd'. Below this, the 'Staged Files (0)' section is empty.

example-existing-project-sdsr

repository
example-existing-project-s... > branch
report

Undo Redo Pull Push Branch Stash Pop Boards Timelines

Viewing 3/3 Show All
Filter (% + Option + f)

LOCAL 2/2
main
report

REMOTE 1/1
example-repo-sdsr2
main

STASHES 1
on: main

PULL REQUESTS 0

ISSUES

Select an issue tracker for this repo
GitHub

BRANCH / TAG

GRAPH

COMMIT MESSAGE

WIP on main
Merge pull request #1 from datalorax/exa...
created report template
styled plot
Fixed merge conflict - took remote version
Updated README locally
Update README.md
ignored the .DS_Store file
project setup & initial exploration
Initial commit

Pop Stash

commit: 886bdb

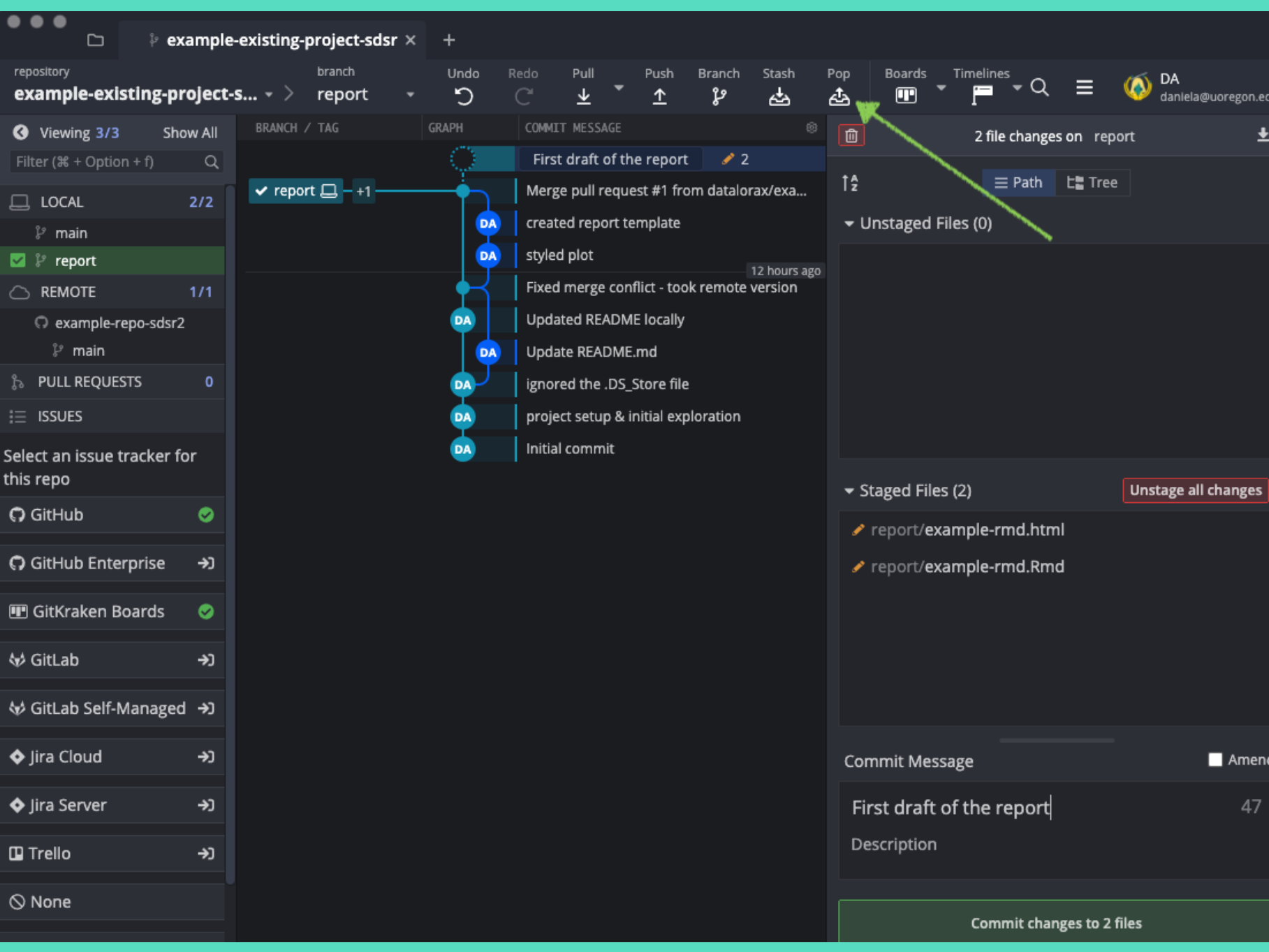
Merge pull request #1 from datalorax/exa...
Example branch

DA Daniel Anderson
authored 12/22/2020 @ 12:02 PM

G GitHub
committed 12/22/2020 @ 12:02 PM

2 modified + 2 added

plots/ggplot-example.pdf
+ report/example-rmd.html
+ report/example-rmd.Rmd
scripts/exploration.R



General advice

- Create a repo
- Add collaborators
- Always pull before starting new work
- Always commit all changes before finishing your work
 - If you don't do this, you might end up with automatic stashing
- File Issues to track work that should be completed
 - Include assignments for individuals
 - Tag issues to help with organization

General advice

- Use branches to create new changes
 - A good rule of thumb – one branch for each issue
- Submit pull requests when the work on a given branch is complete, and link it to the corresponding issue
 - Tag collaborators to review the pull request
 - Use GitHub's review resources to comment on individual lines of code, as needed
 - Use the PR to have conversations about the changes and any revisions needed

General advice

- Merge pull requests and delete the remote branch
 - Move back to your local, checkout the main branch, pull the merged changes, delete the local branch

Next time

Introduction to visualizations