Version control for researchers with Git and GitHub

Karin Knudson

karin.knudson@tufts.edu

Why use Git and GitHub?

- Sleep better at night with version control and remote repositories

- Collaborate smoothly with teammates

- Promote and maintain quality in your code

- Increase the impact of your research

- Develop your career

- Contribute in the open source community

After this workshop you should be able to…

- Explain how version control, Git, and GitHub can help you and your team

- Create a new project that is tracked with Git, or add Git to an existing project

- Use a simple workflow with Git and GitHub that is useful for small or individual projects

- Use a more complex workflow with branches and pull requests

- Contribute to some else's open source project on GitHub

- Know where to go to learn more

Git vs. GitHub

- Git = software for version control

Will learn to use basic Git commands: `init, remote, fetch, merge, status, add, commit, merge, push, fetch, checkout`

- GitHub = a repository hosting service with a graphical interface and additional tools for collaboration and more

Will learn to put repositories on GitHub, collaborate when others' have GitHub repositories, use GitHub pull requests.

Other options: e.g. BitBucket

Set up a new repository with Git and GitHub

- Initialize with `git init`
- Connect your repository to a remote GitHub repository with GitHub's interface and `git remote add`
- Copy the content you created to your remote repository (hosted on GitHub) with `git push`

local repository

GitHub

remote repository
(often called `origin`)
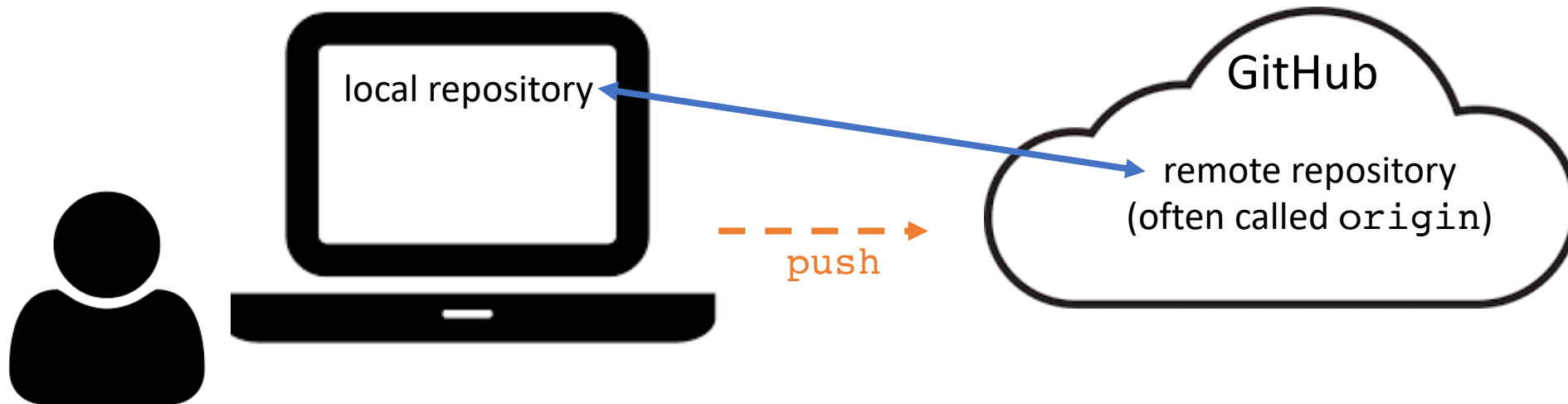
push
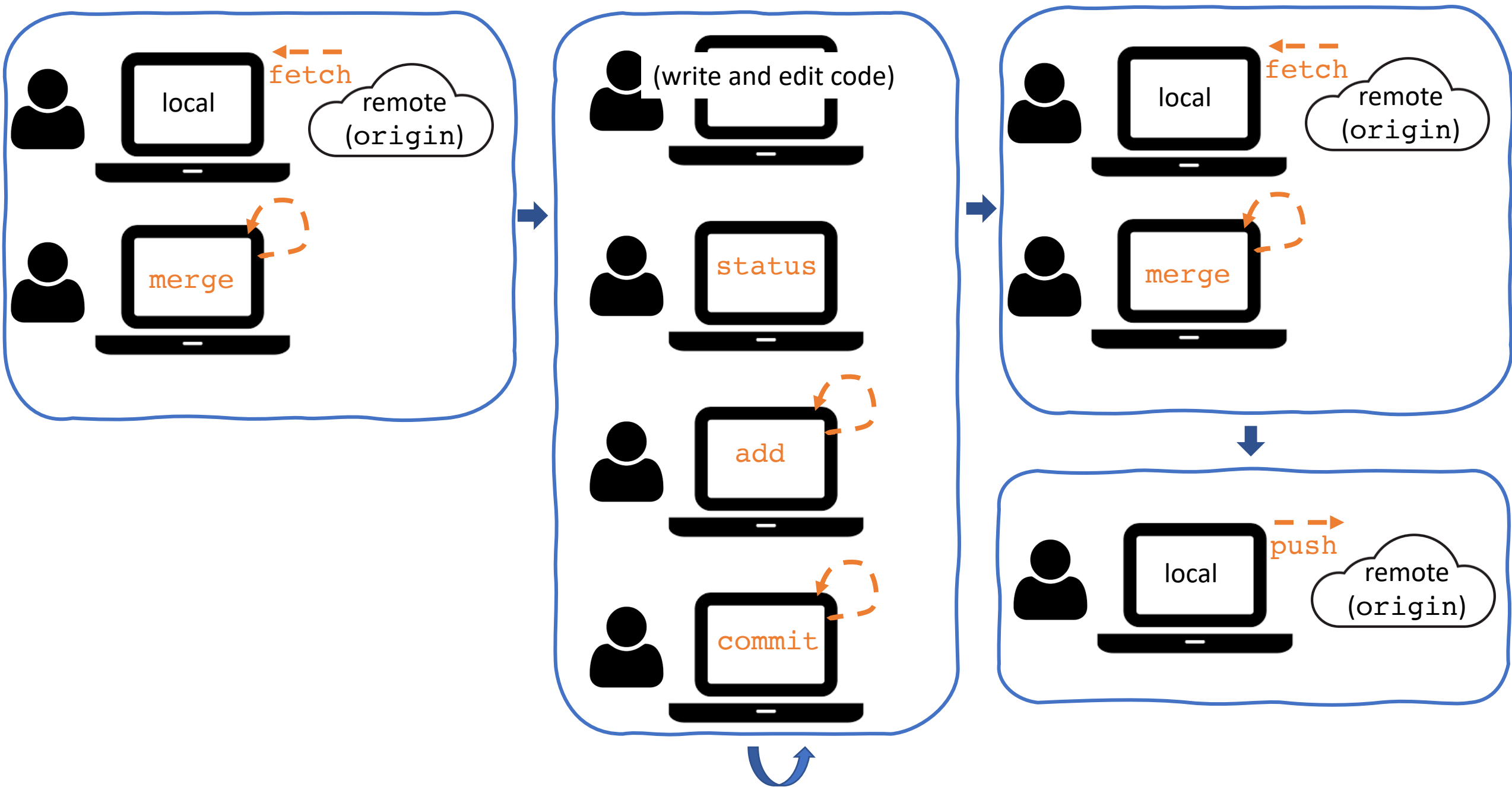
Follow the numbered instructions in parts I and II

https://xkcd.com/1656/

git + command + flags/arguments

git push -u origin master

git fetch origin

git merge origin master

git add hello.py

Follow the instructions in part III to practice the workflow now

https://xkcd.com/1296/

Merge conflicts

- If you and a collaborator are simultaneously changing different parts of the code and merging, no problem!

- If you change a line of code, and in the meantime some one has made a different change to **the same line** and pushed those changes, you can have a merge conflict.

Use Git branches and GitHub pull requests
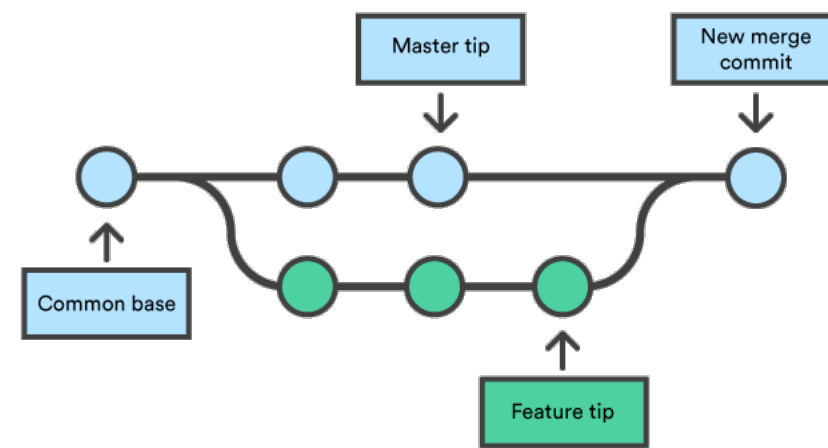
- Instead of making all changes to master branch, create different branches for different features you develop.

- Make branches locally, and then create and connect them to corresponding remote branches.

- Once your feature-specific branch is where you want it to be, *then* merge the changes on this branch back into the master branch of the remote repository.

- Use GitHub's **pull requests** to get collaborator's consent and input before merging the code on feature branch into the master branch of the remote repository.

- Move between branches with the `checkout` command
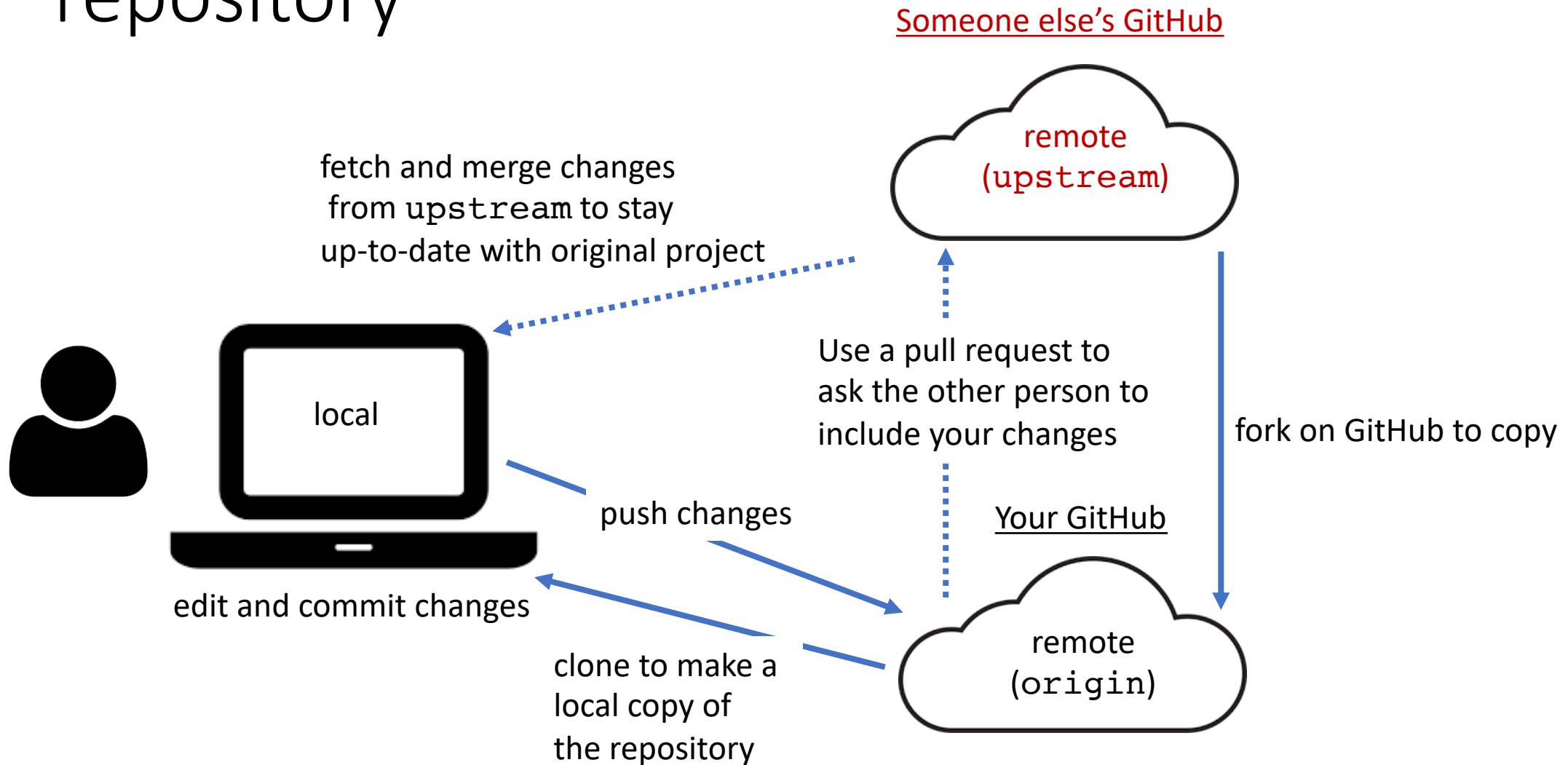
   (caution: `git checkout <filename>` is dangerous).

- Move between branches with the `checkout` command (CAUTION: `git checkout <filename>` is dangerous).

Follow the instructions in part V to practice the workflow now

Using GitHub forks to build on someone else's repository

Someone else's GitHub

remote
(upstream)

fetch and merge changes
from `upstream` to stay
up-to-date with original project

local

Use a pull request to
ask the other person to
include your changes

fork on GitHub to copy

push changes

Your GitHub

edit and commit changes

clone to make a
local copy of
the repository

remote
(`origin`)

"Homework"

- Contribute to https://github.com/karink520/TuftsGitHubSampleToUpdate using the process above as outlined in part VI.

Some other important topics

- Stashing changes
- Undoing changes and reverting
- See what has changed with `git diff`
- Ignoring files you don't want to track
- Use ssh to connect to GitHub
- GitHub actions (e.g. to automatically run tests or other checks)

(see part VII)

Resources (part VIII)

- Searches and StackOverflow

- DangitGit!? https://dangitgit.com/

- GitHub Guides https://guides.github.com/introduction/git-handbook/

- Browser game for learning about Git branching https://learngitbranching.js.org/

- "A minimal tutorial": https://kbroman.org/github_tutorial

- Atlassian tutorials https://www.atlassian.com/git/tutorials

  and Git "cheat-sheet" https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet)

- MIT CSAIL's "Missing Semester" lesson on Git:

https://missing.csail.mit.edu/2020/version-control/

- Renaming the default branch: https://dev.to/rhymu8354/git-renaming-the-master-branch-137b

Office hours:

Questions??