gesis

Leibniz Institute
for the Social Sciences

Tools and Workflows for Reproducible Research in the Quantitative Social Sciences

Using Git and GitHub

Bernd Weiß

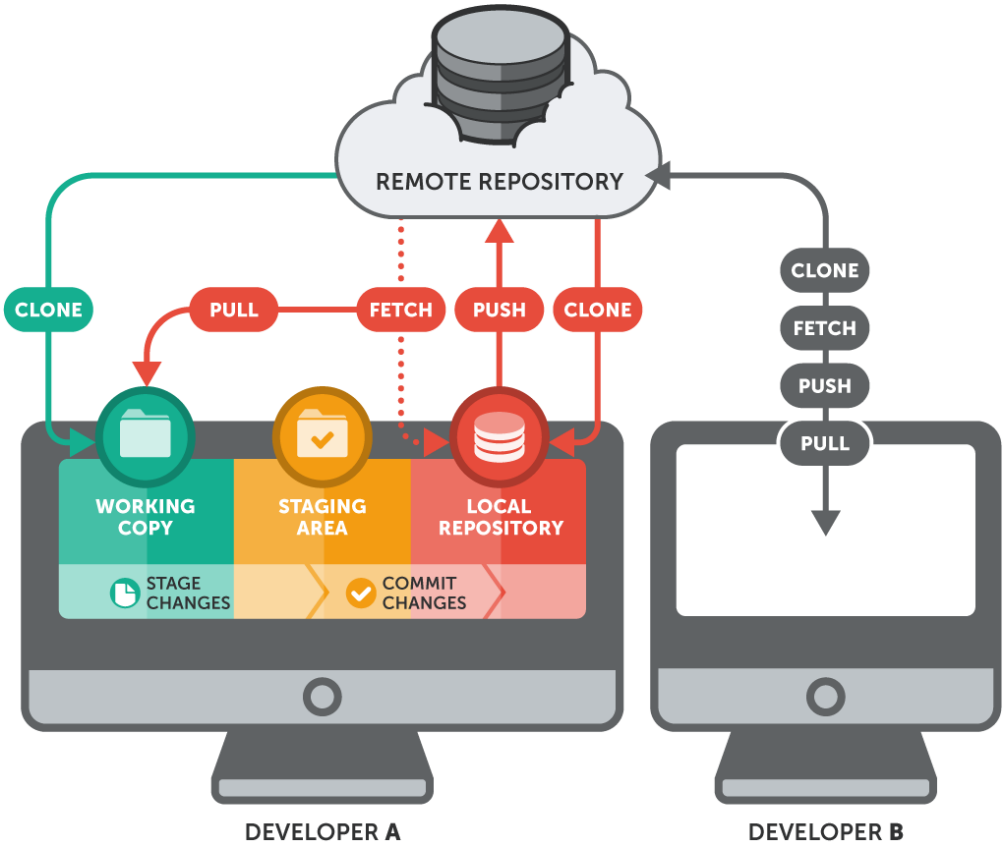
2022-11-18

Leibniz Association

Overview

Recap

- Basic introduction to Git
- Git workflow
- Important Git concepts/commands (`git init`, `git add`, `git commit`, `git log`, `git show`, `git clone`, `git checkout`, `git branch`, ...)
- Also, the following slides are heavily influenced by another course on "Reproducible research workflows for psychologists" by Frederik Aust and Johannes Breuer, especially the part on "Collaborate with Git & GitHub"



(Source: https://www.git-tower.com/learn/git/ebook/en/desktop-gui/remote-repositories/introduction)

Please, let me in (authentication)

Local or remote & HTTPS or SSH?

- A Git project is stored in a repository, which can be local or remote
- When using Git to access a remote repository (for backup or collaborative work) on a remote server, you need to authenticate yourself to the server
- There are two ways of authentication: HTTPS or SSH

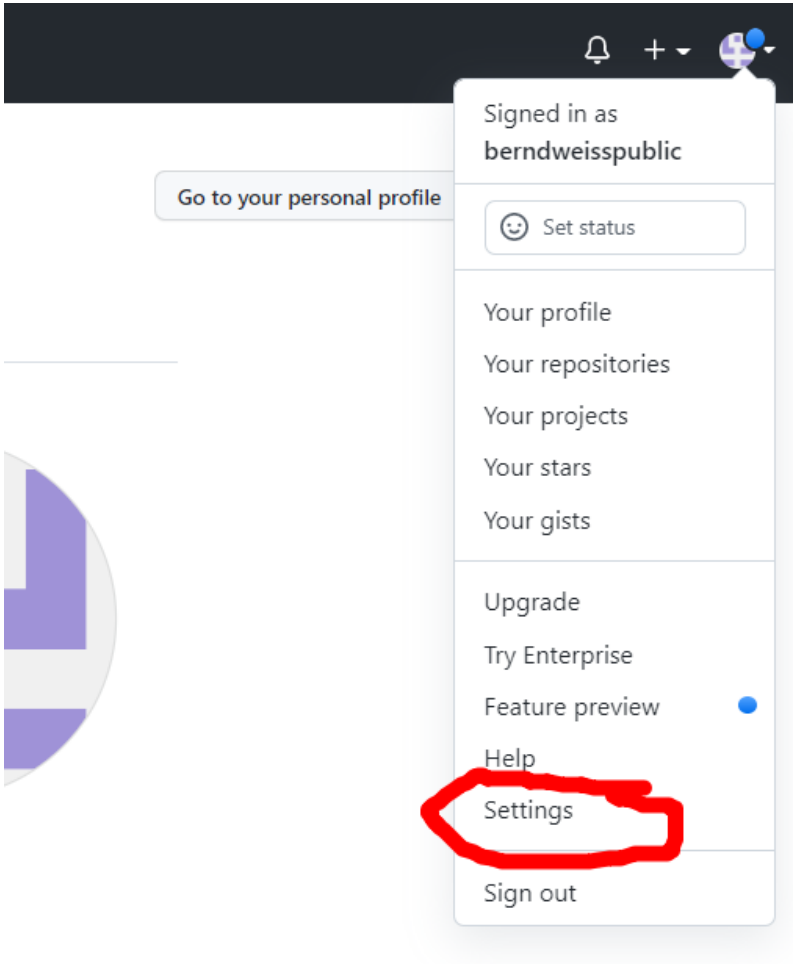
Choose a remote server

- GitHub (which will be using)
- GitLab
- ...

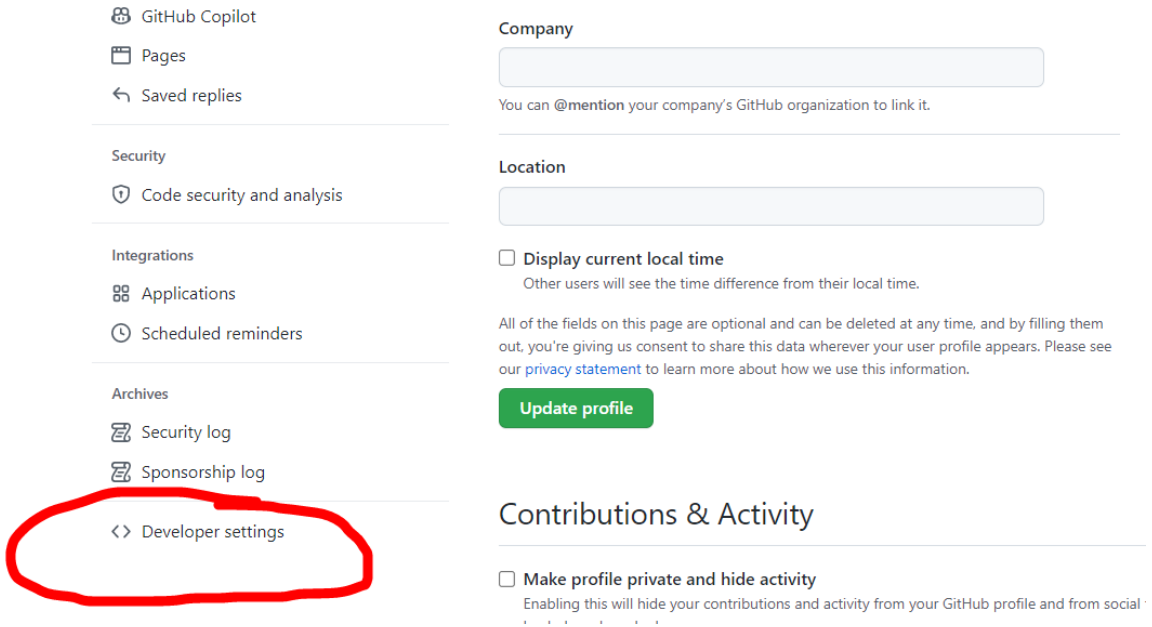
GitHub: Using personal access tokens

- These days, authentication via personal access tokens (PAT) (and https) seems the way to go when using GitHub
- In the following, I will illustrate the process using multiple screenshots
- Note that my explanation does not include any R/RStudio-related processes. Johannes will talk about these things in more detail
- Finally, I will focus on MS Windows. Arnim can help with macOS/Linux.

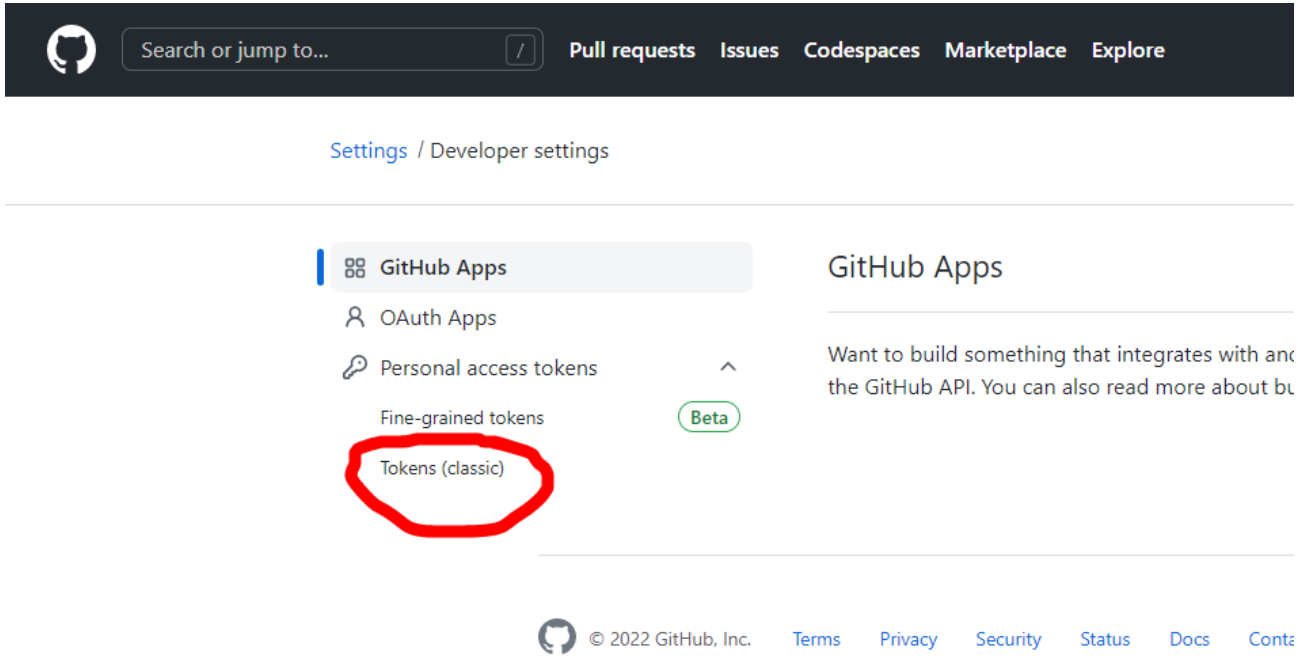
In GitHub, go to the `Settings` website:



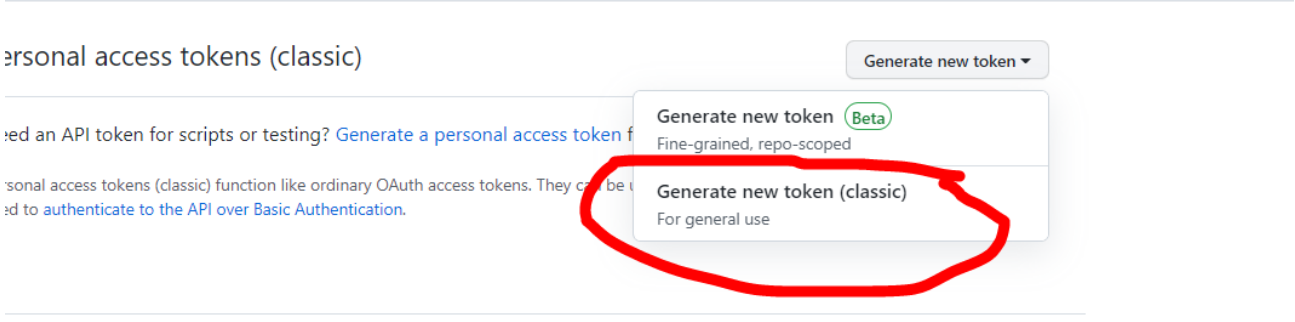
Next, go to the `Developer Settings` entry:



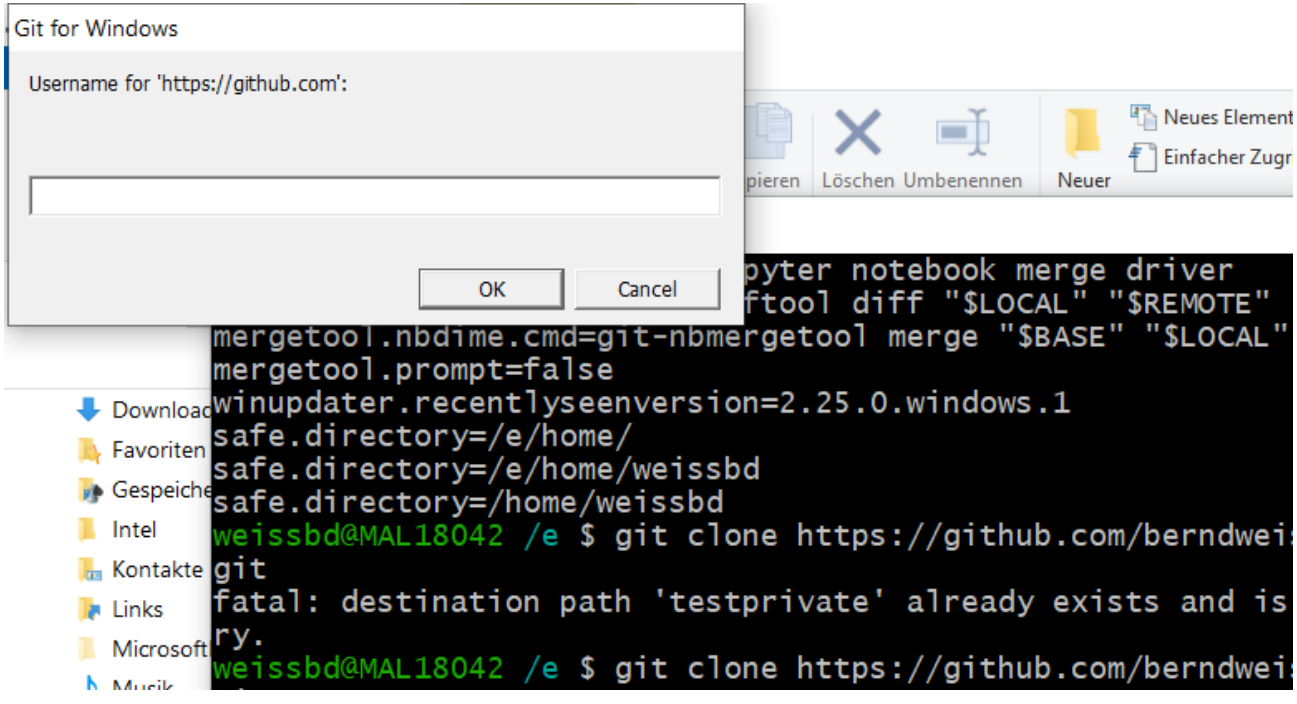
Then, choose `Tokens (classic)`:



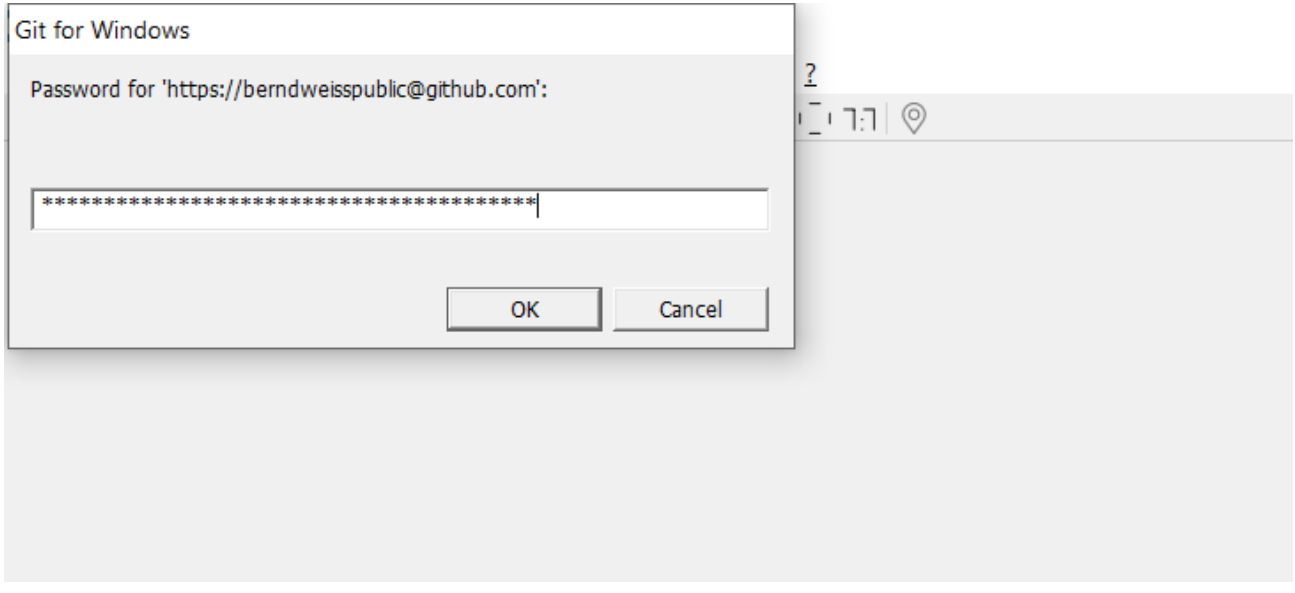
And, generate a new token; important, save this Token (e.g., in your password manager):



When you are now cloning a new repository (or pushing/pulling for the first time), you will be asked once to enter your username...



... and your Token (even though it says "Password"):



If, for whatever reason, you decide to reset/remove your credentials, you can do so using the Windows Credentials Manager (in German: "Anmeldeinformationsverwaltung")

Setting up SSH

- SSH is a network protocol that comes in handy, when you work with remote repositories and when you do not want to type-in your password every time you pull (fetch) or push (send) from a remote repository. You still need to authenticate yourself, though.
- To work with Git on you local computer, you do not need SSH (= Secure Shell).
- Authentication in SSH (which is also the name of the program) works by using a private and a public key (usually the public key has the file extension `.pub`, e.g., my public key is `id_rsa.pub`). When you start working with SSH for the very first time, you have to create both keys.

- The private key remains on your local computer and you have to make sure that it is safe -- it is a simple text file and it is your password now, and everyone who has your private key can access your files. Again, everyone who has your private key has your password!

This is what my *public key* looks like:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAyOQ9RT6TkfgkdO2NspzdVJE5CZ03yYAhVwLGo
CrI3E9/Ix0MAySunXExjhsQi2XkhPBjLOEahYuuLaAWHuBc7apUPRNSBy+mdUHnH3
0BdTQijQ6vj3RL99HO4yrZnipIlkS5ufw/+hpbXXOzSOqTvyGtL9ygm3eA2HDSQtz
2ptFq8anODJDKrgTbNLb/YZ9KDIcpdO/Sfk4LtvaGF3tIFlyE+pogNmN4eWiYg9Xv
25BhVVxWMHadRFLeDastWO4SedriEHzQYaNgxVNTufqolJ0nbg4R//fVDxjR2SbzV
AHLZ+eVPUx+vzcPVMP9wYPcnii9YLiSRy+hlUAOR/kXeQ== berndweiss
```

The *public key* (not the private key!) has to be stored at the GitHub/GitLab/... website. Now, everyone who has your public key can encrypt files (that are sent to you via the internet) but only you (or anyone else who has your private key) can decrypt the files. And, for that reasons you do not have to login everytime you push/pull files from the remote repository.

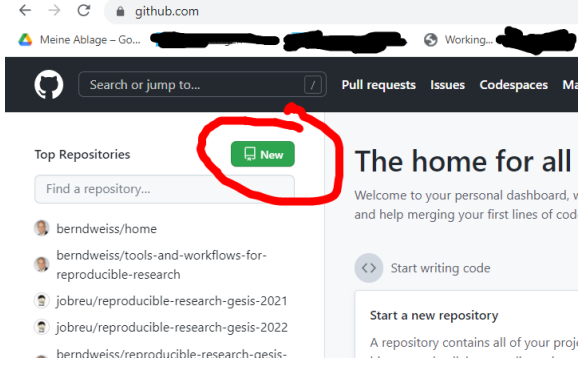
- How to setup SSH on you computer is explained on this website: https://docs.gitlab.com/ce/ssh/README.html ("Generate an SSH key pair")
- The most important point is that `ssh` is able to find your key pair, i.e., it needs to be located in your HOME folder

If everything works well, you should receive the following friendly welcome message after typing in the command `ssh -T git@github.com`:

> Hi berndweiss! You've successfully authenticated, but GitHub does not provide shell access.

Exercise

- In a previous exercise, you have created your own repository (let's call it `your-new-repo`)
- Now, go to your GitHub account and create a new repository on GitHub
- Startpage -> tab "Repositories" -> green button "New"



- Enter a new "Repository name"
- Make it "Private" (unless you have something important to share)
- Do **not** check any of the "Initialize this repository with" boxes
- Hit the green "Create repository" button

- Choose SSH or HTTPS as protocol ("Quick setup — if you've done this kind of thing before")
- Look for "...or push an existing repository from the command line"
- SSH:
  - Copy the line `git remote add origin git@github.com:berndweiss/your-repo-name.git`
  - Execute the command `git remote add origin git@github.com:berndweiss/your-repo-name.git` in the Git Bash in your local repository (`your-new-repo`)

- HTTPS:
  - Copy the line `git remote add origin https://github.com/berndweiss/your-repo-name.git`
  - Execute the command `git remote add origin https://github.com/berndweiss/your-repo-name.git` in the Git Bash in your local repository (`your-new-repo`)
- Make some changes (edit a text file, create a new file etc.). Then `add` and `commit` these changes locally
- Make sure that `git status` shows a clean repository
- Now you can run your first `git push origin main`
- Reload the GitHub page via F5; you now should see the content of your local repo `your-new-repo`

Security

- Be extremely careful including sensitive information (e.g., personal data, passwords, access tokens) into a (public) GitHub repository. There are people out there who search for these things... see also https://docs.github.com/en/code-security/secret-scanning
- Please enable two-factor authentication on GitHub
- xxxxx how to remove entries ...github remove

Working with remote respositories

git pull

git push

Inviting collaborators

class: center, middle

Merge conflicts

Merge conflicts

- When mult

https://www.git-tower.com/learn/git/ebook/en/command-line/advanced-topics/merge-conflicts

class: center, middle

Forking

Pull requests

Now, work together!

Exercises

- Since this exercise is about collaboration, you need at least another person who is willing to collaborate with you. Go to our Ilias repo, there is a file called `wsrr_groups-github.txt` which assigns participants to groups.
- We have setup a Breakout room for each group.

- Task 1: Start the collaboration
  - Every group member should create a new public repo on GitHub, the repo should include at least one text file
  - The other group member(s) should be invited to this GitHub repo
  - Clone the repo on your local computer

- Task 2: Fight!
  - Everyone in the group: On your computer, modify the **first line** of the text file in your partner's repo. Make sure that your changes are different (delete a word in the first line, add a word etc.)
  - Use the GitHub file editor to also make make changes that should not be identical to your local changes
  - `add` and `commit` your changes on your computer and then try to `pull` the modified repo from GitHub.
  - You will hopefully experience a merge conflict. Deal with it.

- Task 3: Forking
  - Fork my public test repo: https://github.com/berndweiss/twrr_public_testrepo
  - Fork your partner's public test repo

- Task 4. Pull requests
  - Go to you partner's repo, fork it, introduce some changes, and then make a pull request
  - Accept (or close, i.e., decline) your partner's pull request (you can also check out the options under `Files changed` -> `Review changes`)

https://www.git-tower.com/learn/git/ebook/en/command-line/advanced-topics/merge-conflicts