

# gesis

Leibniz Institute  
for the Social Sciences



## Tools and Workflows for Reproducible Research in the Quantitative Social Sciences

### Using Git and GitHub

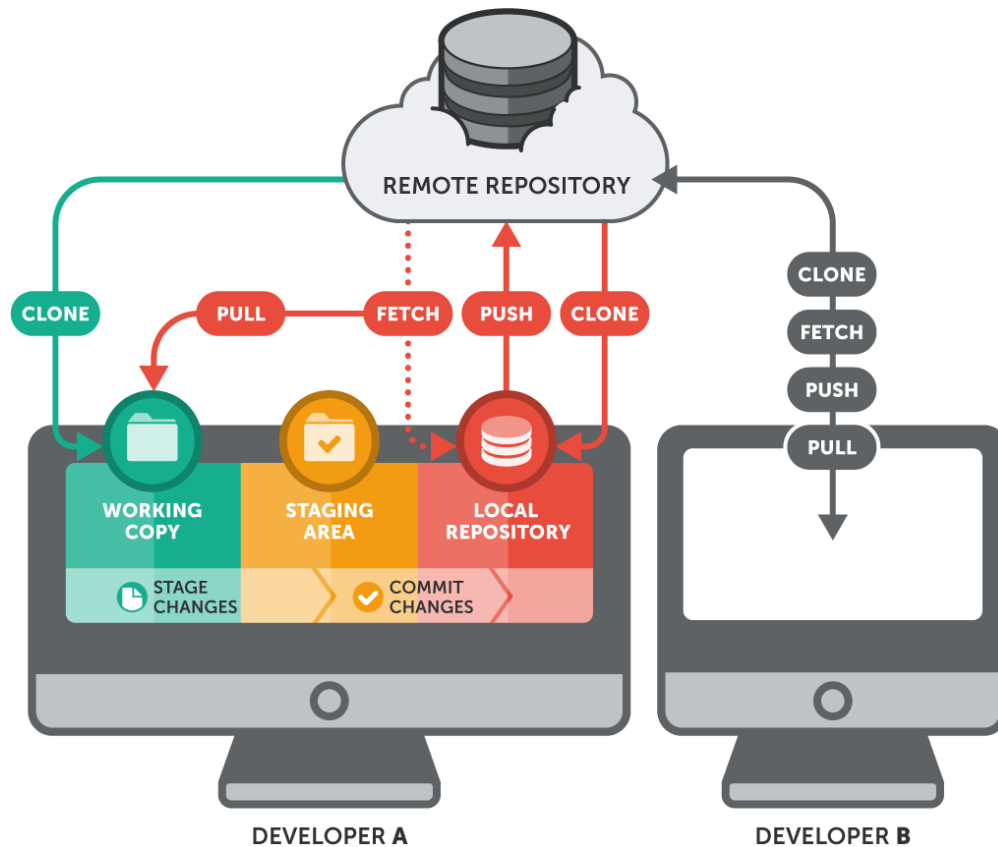
*Bernd Weiß*

*2022-11-18*

# 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/repo 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

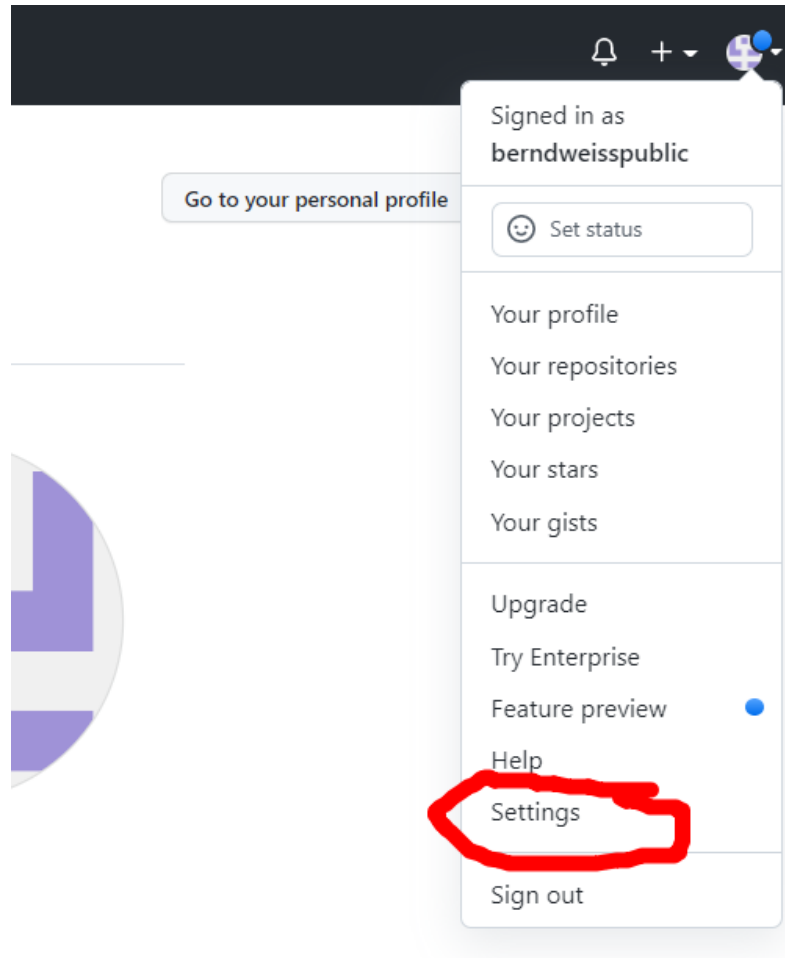
- GitHub (which will be using)
- GitLab
- ...

# GitHub: Using personal access tokens

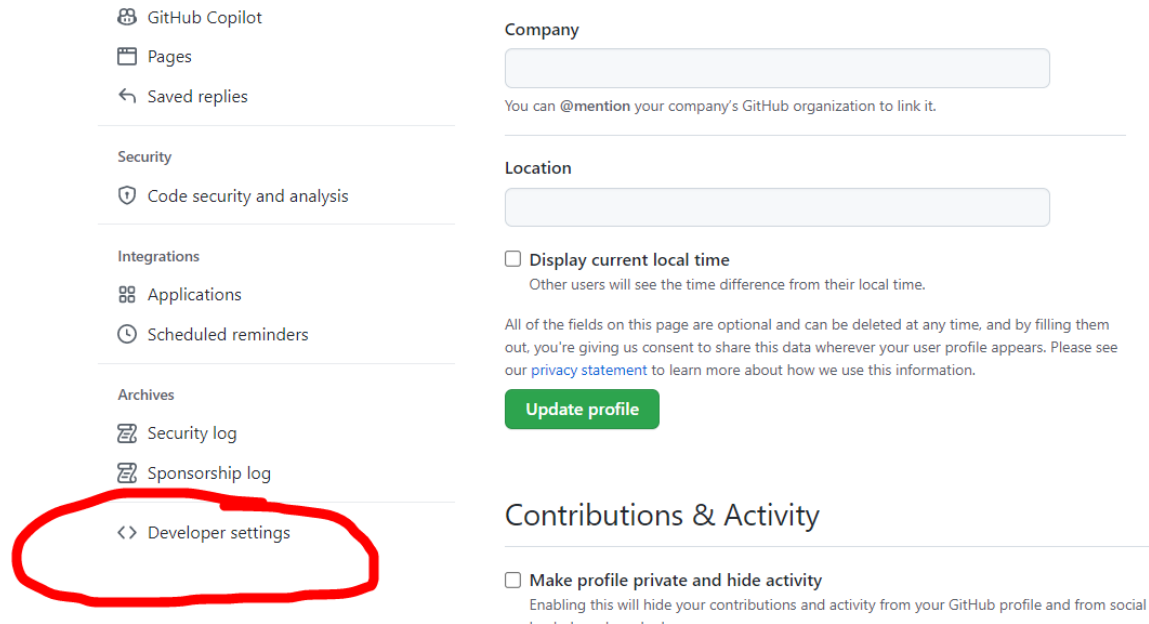
- These days, authentication via personal access tokens (PAT) (and [https](#)) seems the way to go when using GitHub
- In the following, I will illustrate the process using multiple screenshots
- Note that my explanation does not include any R/RStudio-related processes. Johannes will talk about these 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:



The screenshot shows the GitHub Developer Settings page. On the left sidebar, the 'Developer settings' option is highlighted with a red circle. The main content area shows the 'Company' and 'Location' fields, both of which are empty. Below these fields, there is a checkbox for 'Display current local time' which is unchecked. A green 'Update profile' button is visible. The 'Contributions & Activity' section is partially visible at the bottom, showing a checkbox for 'Make profile private and hide activity' which is also unchecked.

**Developer settings**

GitHub Copilot

Pages

Saved replies

Security

Code security and analysis

Integrations

Applications

Scheduled reminders

Archives

Security log

Sponsorship log

**Company**

You can @mention your company's GitHub organization to link it.

**Location**

☐ **Display current local time**  
Other users will see the time difference from their local time.

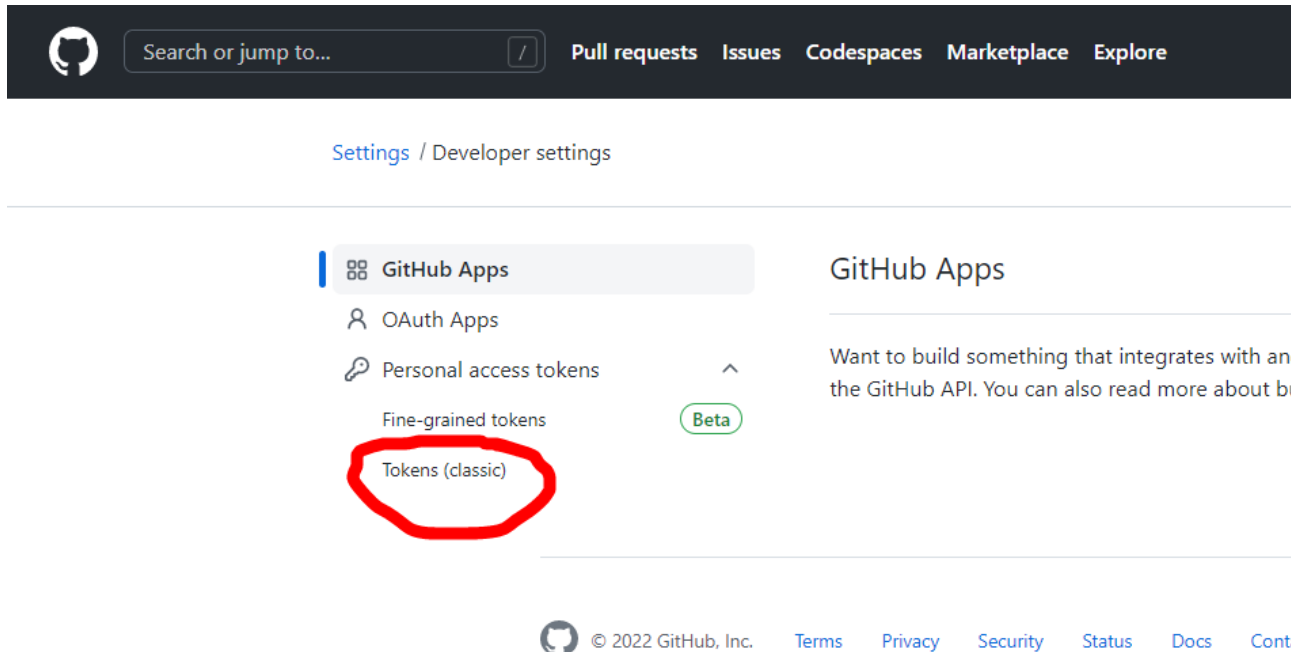
All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our [privacy statement](#) to learn more about how we use this information.

**Update profile**

**Contributions & Activity**

☐ **Make profile private and hide activity**  
Enabling this will hide your contributions and activity from your GitHub profile and from social

Then, choose Tokens (classic):



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

### Personal access tokens (classic)

Generate new token ▼

Need an API token for scripts or testing? [Generate a personal access token](#) for

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used to [authenticate to the API over Basic Authentication](#).

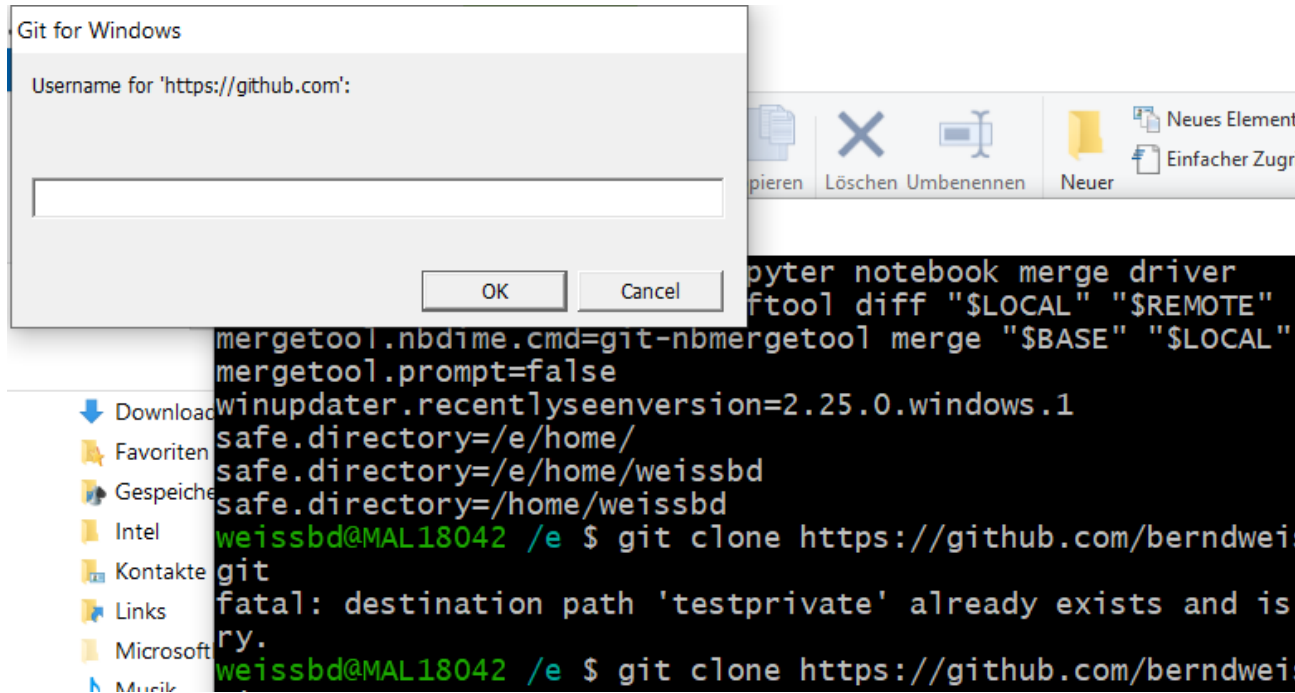
Generate new token Beta

Fine-grained, repo-scoped

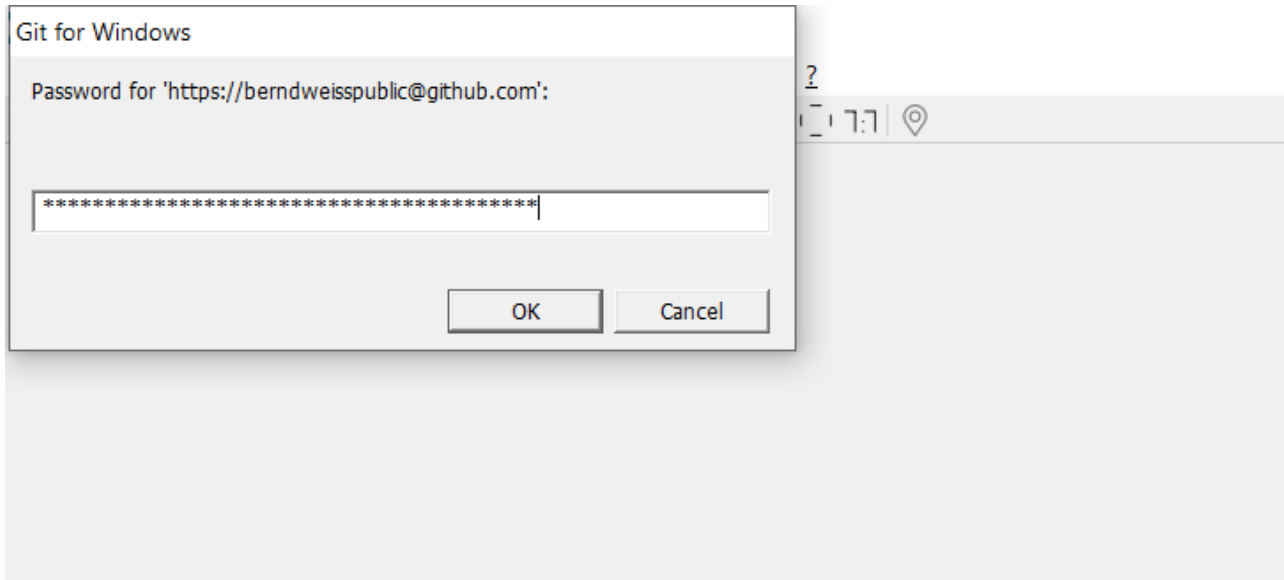
Generate new token (classic)

For general use

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
AAAAB3NzaC1yc2EAAAABIwAAAQEAY0Q9RT6Tkfgkd02NspzdVJE5CZ03yYAhVwLGo
CrI3E9/Ix0MAySunXExjhsQi2XkhPBjLOEahYuuLaAWHuBc7apUPRNSBy+mdUHnH3
0BdTQijQ6vj3RL99H04yrZnipILkS5ufw/+hpbXX0zS0qTvyGtL9ygm3eA2HDSQtz
2ptFq8an0DDJKrgTbNLb/YZ9KDIcpd0/Sfk4LtvaGF3tIFlyE+pogNmN4eWiYg9Xv
25BhVVxWMHAdRFLedastW04SedriEHZQYaNgxVNTufqolJ0nbg4R//fVDxjR2SbzV
AHLZ+eVPUx+vzcPVMP9wYPcni9YLisRy+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.

# 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>
- Use your `.gitignore` file to exclude sensitive files/folders
- Please enable [two-factor authentication on GitHub](#)
- In case you have accidentally include sensitive information, check out this GitHub website on [Removing sensitive data from a repository](#)

# Working with remote repositories

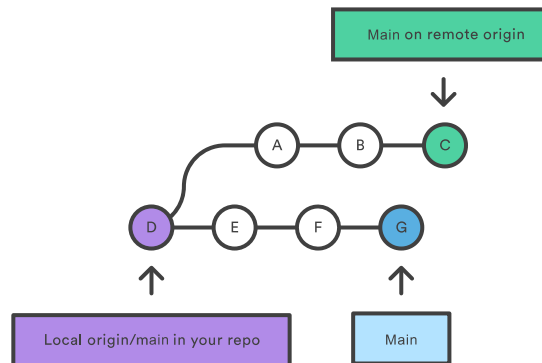
# Adding a remote repository via `git`

## `remote add`

- Can be accomplished using the git command `git remote add <name> <url>`. The usual name for <name> is `origin`, however, feel free to choose another name (when you clone a repo, then this has already happened).
- **SSH:** The <url> for this repository looks like `git@git.gesis.org:weissbd/ps2017-xx-intro2git.git`; another example is this one: `git@github.com:berndweiss/ps2017-11_porto-campbell-ma-workshop.git`. The url can be found in the respective github/gitlab repository.
- **HTTPS:** `git remote add origin https://github.com/berndweiss/lala.git`

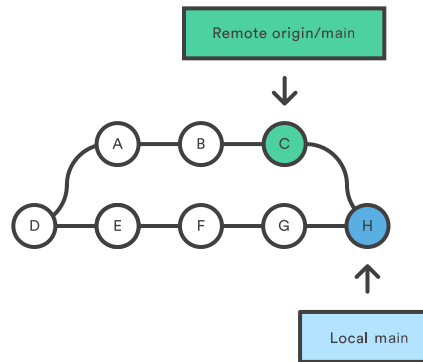
# git pull

- Given that you have already established a link to an remote server (such as GitHub, e.g., via `git remote add` or `git clone`), updates can be downloaded via the `git pull <name remote server> <branch>` command
- Most often, this is `git pull origin main`



(Source: <https://www.atlassian.com/git/tutorials/syncing/git-pull>)

- In the background, `git pull` combines two steps, `git fetch` and `git merge`



(Source: <https://www.atlassian.com/git/tutorials/syncing/git-pull>)

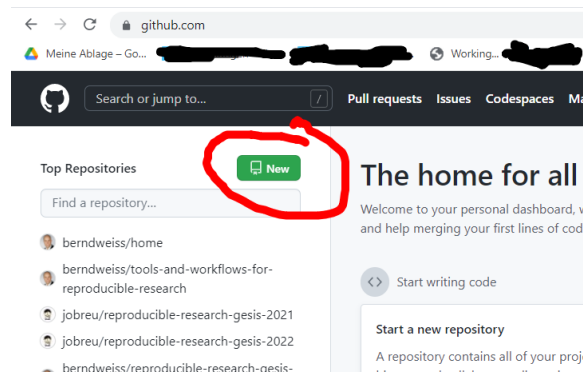


# git push

- Again, given that you have already established a link to an remote server (such as GitHub, e.g., via `git remote add` or `git clone`), updates can be uploaded via the `git push <name remote server> <branch>` command
- Most often, this is `git push origin main`
- More information can be found here:  
<https://www.atlassian.com/git/tutorials/syncing/git-push>

# 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"
- HTTPS:
  - GitHub offers some assistance in terms of providing the full `git remote add` command; copy this line, which should look like

```
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)

- 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`)
- 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`

- You can delete a GitHub repository via the tab "Settings" -> "Options", then scroll down -> "Danger Zone" -> "Delete this repository"

# Inviting collaborators

# Adding collaborators to your GitHub repo

- Adding collaborators works only for GitHub repositories that you own
- GitHub provides a lot of collaboration features
  - Edit files in browser
  - Change highlighting and commenting
  - Interactive revise-and-resubmit workflow
  - Issue tracker (to-do list and discussion)

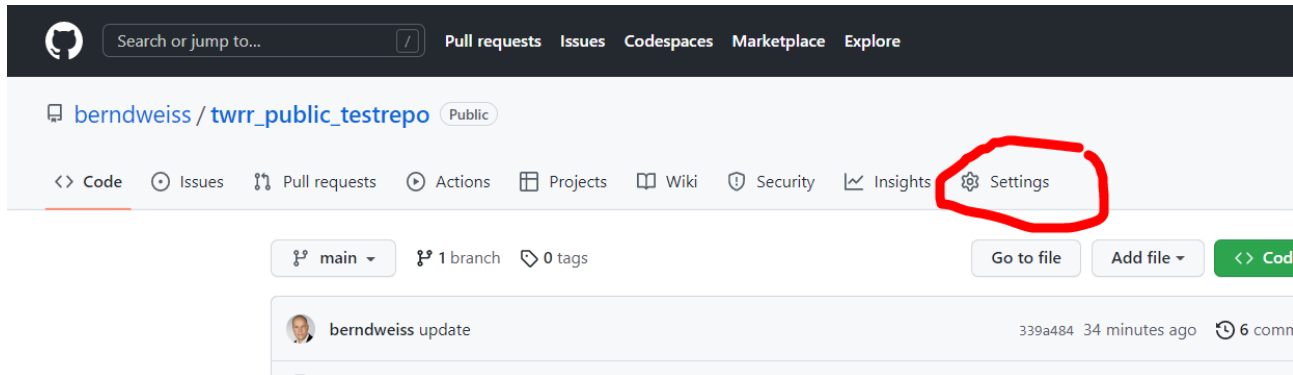
(Source: [http://frederikaust.com/reproducible-research-practices-workshop/slides/6\\_github\\_collaboration.html#5](http://frederikaust.com/reproducible-research-practices-workshop/slides/6_github_collaboration.html#5))



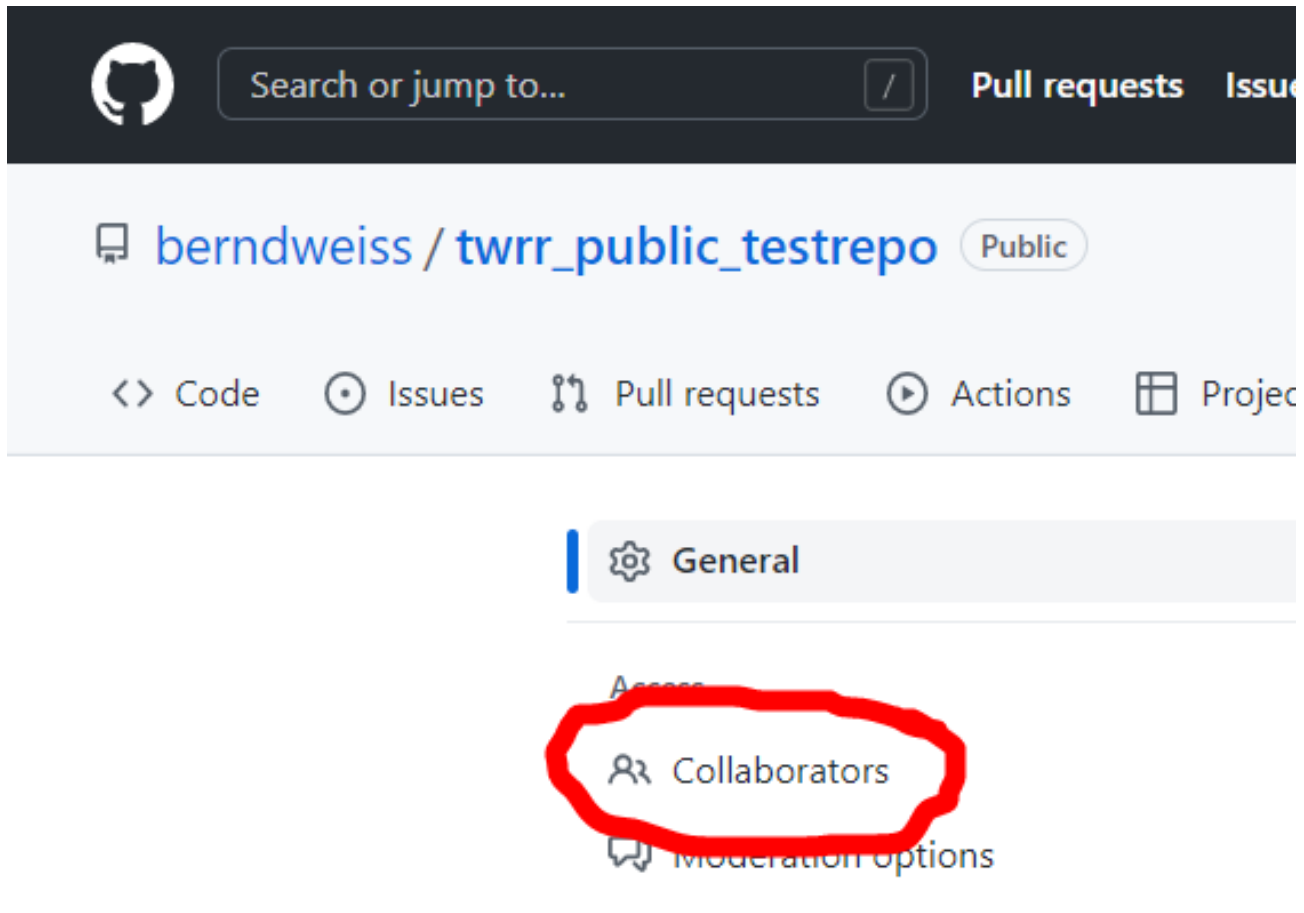
# Workflows for collaboration

- Adding changes to a repo without prior review
  - Push directly to main branch on GitHub
  - You need be an invited collaborator
- Suggest changes with review (pull request)
  - Create a new branch ("parallel universe" of repository)
  - You can be an invited collaborator or a complete stranger
- Edits can be made directly on GitHub or locally on your computer

(Source: [http://frederikaust.com/reproducible-research-practices-workshop/slides/6\\_github\\_collaboration.html#8](http://frederikaust.com/reproducible-research-practices-workshop/slides/6_github_collaboration.html#8))



The screenshot shows the GitHub interface for the repository 'berndweiss / twrr\_public\_testrepo'. The repository is marked as 'Public'. The navigation bar includes links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Settings' link, which includes a gear icon, is circled in red. Below the navigation bar, there are buttons for 'Go to file', 'Add file', and a green 'Code' button. A commit history section shows a recent update by 'berndweiss' with the message 'update', commit hash '339a484', and '34 minutes ago'.



The screenshot shows the GitHub interface for the repository 'berndweiss / twrr\_public\_testrepo'. The repository is marked as 'Public'. The navigation bar includes links for 'Code', 'Issues', 'Pull requests', 'Actions', and 'Projects'. Below the navigation bar, the 'General' tab is selected, showing options for 'Access', 'Collaborators', and 'Moderation options'. The 'Collaborators' option is highlighted with a red hand-drawn circle.

Search or jump to... / Pull requests Issues

berndweiss / twrr\_public\_testrepo Public

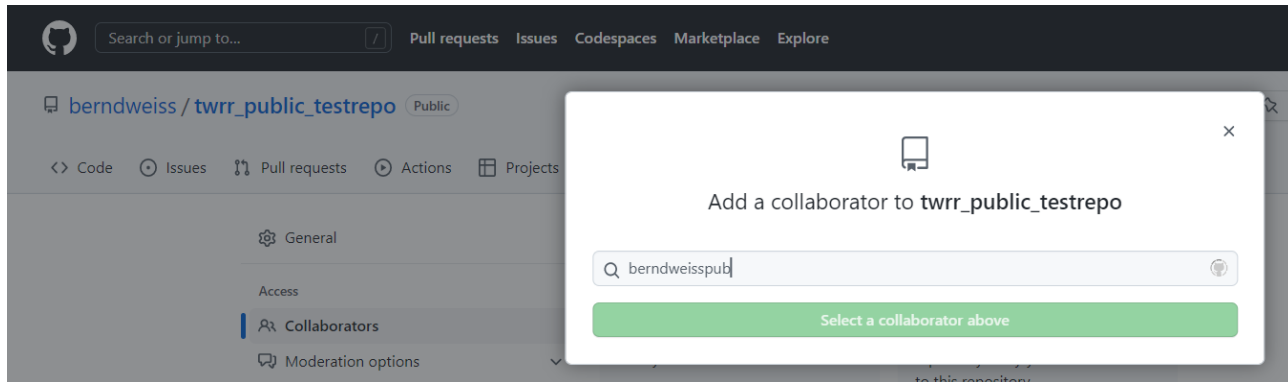
<> Code Issues Pull requests Actions Projects

General

Access

Collaborators

Moderation options




# Merge conflicts

# Merge conflicts


- Merge conflicts occur when there are two competing changes that affect the same file *and* the same lines in that same file; or if one person decided to delete it while the other person decided to modify it
- Git will inform you about a merge conflict and will indicate the two competing changes in a file




(source: <https://www.git-tower.com/learn/git/ebook/en/command-line/advanced-topics/merge-conflicts>)


berndweiss / twrr\_public\_testrepo
Public

<> Code
Issues
Pull requests
Actions
Projects
Wiki
Security

main
twrr\_public\_testrepo / test.R


berndweisspublic i want to draw 100000 observations


2 contributors



3 lines (3 sloc) | 33 Bytes

```

1 x <- rnorm(100000)
2 mean(x)
3 sd(x)

```

```
nothing to commit, working tree clean
weissbd@MAL18042 /e/tmp/twrr_public_testrepo (main) $ git add . -A
weissbd@MAL18042 /e/tmp/twrr_public_testrepo (main) $ git commit -m "10 observations are enough"
[main fe79328] 10 observations are enough
1 file changed, 1 insertion(+), 1 deletion(-)
weissbd@MAL18042 /e/tmp/twrr_public_testrepo (main) $ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 671 bytes | 27.00 KiB/s, done.
From https://github.com/berndweiss/twrr_public_testrepo
* branch          main      -> FETCH_HEAD
   339a484..047e3f7 main      -> origin/main
Auto-merging test.R
CONFLICT (content): Merge conflict in test.R
Automatic merge failed; fix conflicts and then commit the result.
weissbd@MAL18042 /e/tmp/twrr_public_testrepo (main) $ |
```



# An example of a merge conflict

- This is how a merge conflict looks like in the file `test.R`:

```
<<<<<<< HEAD
x <- rnorm(10)
=====
x <- rnorm(100000)
>>>>>>> 047e3f7a00a5541622e5a40dc342df3af0591838
mean(x)
sd(x)
```

- A merge conflict only affects the developer who is causing a merge conflict
- You have to resolve the merge conflict by editing the respective file(s) (then add and commit the changes) (remove the `<<<<<<<`, `=====`, `>>>>>>>` and save the file)

```

/usr/bin/bash --login -i
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 671 bytes | 27.00 KiB/s, done.
From https://github.com/berndweiss/twrr_public_testrepo
 * branch          main      -> FETCH_HEAD
   339a484..047e3f7  main      -> origin/main
Auto-merging test.R
CONFLICT (content): Merge conflict in test.R
Automatic merge failed; fix conflicts and then commit the result.
weissbd@MAL18042 /e/tmp/twrr_public_testrepo (main) $ git add . -A
weissbd@MAL18042 /e/tmp/twrr_public_testrepo (main) $ git commit -m "resolve merge conflict; i really think that 10 observations are enough"
[main ba346f7] resolve merge conflict; i really think that 10 observations are enough
weissbd@MAL18042 /e/tmp/twrr_public_testrepo (main) $ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 458 bytes | 458.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/berndweiss/twrr_public_testrepo.git
   047e3f7..ba346f7  main -> main
weissbd@MAL18042 /e/tmp/twrr_public_testrepo (main) $

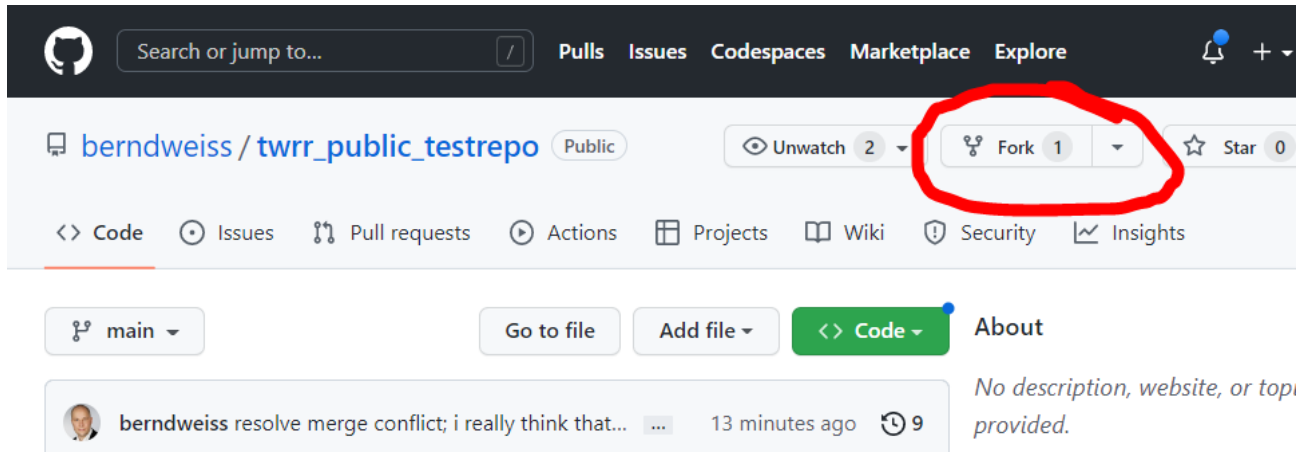
```

# Forking

# Forking

- Forking refers to the process of creating a personal copy of someone else's project
- Forking works only for public repositories (or, you have been invited to a private repository)
- In order to contribute to another (public) repository via *pull requests*, you first need to fork the respective repository

(Source: <https://docs.github.com/en/get-started/quickstart/contributing-to-projects>)



The screenshot shows the GitHub interface for the repository `berndweiss / twrr_public_testrepo`. The repository is public. The top navigation bar includes links for Pulls, Issues, Codespaces, Marketplace, and Explore. Below the repository name, there are buttons for Unwatch (2), Fork (1), and Star (0). The 'Fork' button and its count are circled in red. Below this, there are tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. The 'Code' tab is selected. Below the tabs, there are buttons for 'main' branch, 'Go to file', 'Add file', and 'Code'. The 'About' section is visible on the right, stating 'No description, website, or topics provided.' Below the repository name, there is a commit by `berndweiss` with the message 'resolve merge conflict; i really think that...' and a timestamp of '13 minutes ago'.

# Pull requests

# Pull requests on GitHub

- Pull request only work in the context of a web platform such as GitHub or GitLab
- It is a (polite/only) way to contribute to another person's GitHub repository
  - If you are a collaborator, then it is a polite way to contribute
  - If you are not a collaborator, then it is the only way to contribute
- Note, the following example is based on
  - two GitHub users: `berndweiss` and `berndweisspublic`
  - both users are not collaborating
  - on each slide I will indicate which GitHub user is currently involved

- The current status of the repo (and the file `test.R`) from the perspective of GitHub user `berndweiss`



berndweiss / twrr\_public\_testrepo Public

Code Issues Pull requests 1 Actions Projects Wiki

main twrr\_public\_testrepo / test.R

berndweiss 10 observations are enough

2 contributors

3 lines (3 sloc) | 29 Bytes

```

1 x <- rnorm(10)
2 mean(x)
3 sd(x)

```




- GitHub user `berndweisspublic` insists of 100000 observations and modified the file accordingly

The screenshot shows the GitHub interface for the repository `berndweiss / twrr_public_testrepo`. At the top, there are buttons for 'Unwatch' (2), 'Fork' (1), and 'Star' (0). Below this is a navigation bar with links to 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', and 'Insights'. A light blue warning box states: "You're making changes in a project you don't have write access to. Submitting a change will write it to a new branch in your fork `berndweisspublic/twrr_public_testrepo`, so you can send a pull request." Below the warning, the file path `twrr_public_testrepo / test.R` is shown, along with the branch `berndweiss:main` and a 'Cancel changes' button. The file editor shows the following R code:

```
1 x <- rnorm(100000)
2 mean(x)
3 sd(x)
4
```

On the right side of the editor, there are settings for 'Spaces' (set to 2) and 'No wrap'. A green circular icon with a 'G' is visible in the bottom right corner of the editor area.

Since GitHub user `berndweisspublic` does not own the repository, he cannot commit the changes but *proposes* changes



**Propose changes**

add n = 100000 observations

Add an optional extended description...

**Propose changes** **Cancel**

The next step is that `berndweisspublic` creates a *pull request*, i.e., asking user `berndweiss` to accept his changes

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base repository: berndweiss/twrr\_public\_testrepo
base: main
head repository: berndweisspublic/twrr\_public\_t...
compare: patch-1

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)
Create pull request

1 commit
1 file changed
1 contributor

Commits on Nov 18, 2022

add n = 100000 observations  
berndweisspublic committed 23 seconds ago
Verified
cb5e704

Showing 1 changed file with 1 addition and 1 deletion.
Split Unified

test.R



```

...
@@ -1,3 +1,3 @@
1 - x <- rnorm(10)
1 + x <- rnorm(100000)
2   mean(x)
3   sd(x)



```

GitHub user `berndweiss` is informed about a pull request; he can accept the pull request (merge) or close the pull request, i.e., deny it



add n = 100000 observations #3

 `berndweisspublic` wants to merge 1 commit into `berndweiss:main` from `berndweisspublic:patch-1` 



Conversation 0 Commits 1 Checks 0 Files changed 1


 `berndweisspublic` commented 1 minute ago Contributor 


No description provided.

  add n = 100000 observations Verified `cb5e704`


Add more commits by pushing to the `patch-1` branch on `berndweisspublic/twrr_public_testrepo`.

  **Require approval from specific reviewers before merging**  
Branch protection rules ensure specific people approve pull requests before they're merged. Add rule ×


 **Continuous integration has not been set up**  
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.


 **This branch has no conflicts with the base branch**  
Merging can be performed automatically.

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

 Write Preview H B I ≡ <> ↶ ↷ ↻ @ 📎 ↶

Leave a comment

Attach files by dragging & dropping, selecting or pasting them. 

 Close pull request Comment

# 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 (A and B) 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, i.e., A clones B's repo and B clones A's repo

- Task 2: Fight!
  - Everyone in the group (again, let's use members A and B): On your computer, user A modifies the **first line** of the text file in your partner's repo (B). Make sure that your changes are different (delete a word in the first line, add a word etc.)
  - The repo's owner (B) also makes changes using the GitHub file editor; these changes should not be identical to user's A 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.
  - Now switch sides, i.e., B edits the file locally, and A makes changes on GitHub



- Task 3: Forking
  - Fork my public test repo:  
[https://github.com/berndweiss/twrr\\_public\\_testrepo](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`)