

Git and GitHub

Workflow for cloning a remote repository to your local computer

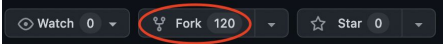

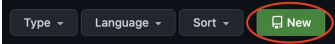
Overview

1. Get a remote repository: Create, Fork or Collaborate
2. Get remote repository onto your local machine
 - 2.1. Copy the SSH link to clone the repository
 - 2.2. Go to your terminal on Mac to clone the repository
3. Work Locally
 - 3.1. Create a new branch (if necessary)
 - 3.2. Work locally in your cloned repository
4. Synchronize with remote repository
 - 4.1. Push changes from your local repository to GitHub
 - 4.2. Open pull request and merge changes (if necessary)

(This is one way to work with Git and GitHub, but by no means the only one.)

1. Get a remote GitHub repository

You first need a remote repository on GitHub, before you can clone - aka copy - it to your local computer. There are three ways to get a repo on GitHub:

FORK a repository 	This is usually done with neue fische repositories , and in general every time you do not have write access to a remote repository.
GET INVITED / ADDED 	Get invited (added) to a repository, where you are allowed to make changes, e.g. your protocol repository , and whenever you'll work on a project with other people.
CREATE a repository 	Create a new repository in your own account (under 'Repositories') to work on your own - or invite other people to collaborate.

2. Get remote repository onto your local machine

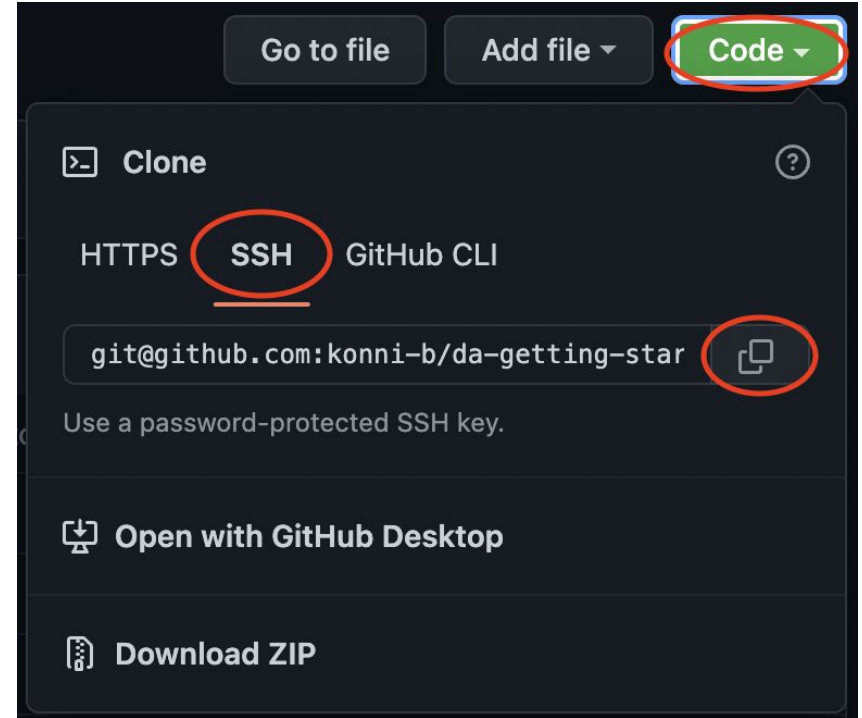
Now that you have a repository on GitHub, the next step is to get the remote repository to your local computer.

Since you enabled the use of the SSH key during the Set up day, you will use this option to clone the remote repository to your local machine.




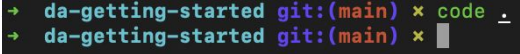
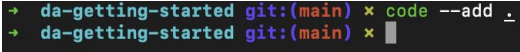
For more information about SSH, please follow this link: [About SSH](#)

2.1. Copy the SSH link to clone the repo

1. Go on GitHub to the repository you want to clone
2. Click on the green 'Code' button
3. Select SSH
4. Copy the link via the overlapping-squares-button



2.2. Go to your terminal on Mac to clone the repo

STEP	WHERE	WHAT
1	parent directory	Change into the directory you would like to use, in this example the neufische directory: <code>cd neufische</code> 
2	neufische directory	type and paste link: <code>git clone <link></code> 
3	neufische directory	change into repository directory: <code>cd <repo name></code> 
4	cloned repository	open repository in e.g. VSCode (optional) <ul style="list-style-type: none">• new window: <code>code .</code>• existing window: <code>code --add .</code>  

3. Work locally

Now that you have your repository on your local computer, you can finally start working locally in it.

If you collaborate with other people on this repository, then you will want to create a new branch to work in as shown on the next slide.

Most of the times, you will work in VSCode during the bootcamp. But of course, there are other options to work locally with the content of a repository.

3.1. Create a new branch (if necessary)

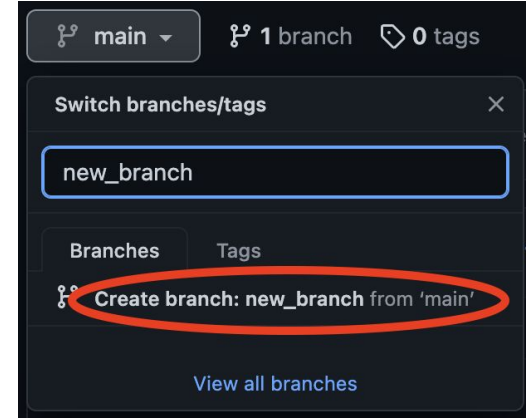
Do you need someone to review your changes?

→ Here is how you create a new branch on GitHub:

1. Go on GitHub to the remote repository, click on main, type in a branch name and click on “Create branch: branch name from ‘main’”.
2. Go to the terminal on your computer (e.g. in VSCode), `cd` into the repository (if you’re not already in it) and use `git pull` to get the newly created branch.
3. Switch to the new branch in the terminal, using `git switch <branch name>`
4. Start working. 😊

Alternatively, you can create a new branch locally:

`git switch -c <branch name>`



```
(base) → da-getting-started git:(main) × git pull
From github.com:konni-b/da-getting-started
* [new branch]      new_branch -> origin/new_branch
Already up to date.
(base) → da-getting-started git:(main) ×
```

```
((base) → da-getting-started git:(main) × git switch new_branch
Branch 'new_branch' set up to track remote branch 'new_branch' from 'origin'.
Switched to a new branch 'new_branch'
(base) → da-getting-started git:(new_branch) ×
```

```
((base) → da-getting-started git:(main) × git switch new_branch
Switched to branch 'new_branch'
Your branch is up to date with 'origin/new_branch'.
(base) → da-getting-started git:(new_branch) ×
```


3.2. Work locally in your cloned repo (via e.g. VSCode)

You are now ready to work with your repository locally, for example in VSCode. You can make changes to existing files, or create new ones.

Before you can push changes to the remote repository, remember to save them first!

Again, do you need someone to review these changes?

→ Then use your newly created branch to push them.

You work in your own repository and no one needs to review your changes?

→ You can just push in main.

4. Synchronize with remote repository

So, you made some changes to your local repository, and now you want to have these changes in the remote repository on GitHub, too.

Some tips:

- Make sure that you saved the changes that you want to commit!
- Use *git status* to see what is going on in your repository.
- Don't wait until you're - fully - finished working locally, you can add and commit in between (more on this on the next slide).
- General sequence of git actions: **add** → **commit** → **push**

4.1 Push changes from your local repository to GitHub

The next slides show you how to add, commit and push local changes to GitHub.

Note that you can add and commit changes repeatedly, before you finally push them once to GitHub!

The Benefit of making smaller commits:

The commit acts like a checkpoint, because whenever you commit something, it is like a point in time you can go back to. So, whenever you think that this might be a change or a version of a file you might want to be able to go back to later, add and commit it.

With the command `git log` you can take a look at previous commits.

4.1. a) Check the status of your local repo

In terminal type:

git status

You get a lot of useful information like which files are untracked (i.e. new), modified or staged; you can see on which branch your on, and so on.

```
[(base) → da-getting-started git:(new_branch) × git status
On branch new_branch
Your branch is up to date with 'origin/new_branch'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  pairings/speed_dating.py
  pairings/speeddating_copy.py
  test1.ipynb
  test2.ipynb
  this_is_a_jupyter_notebook.ipynb

nothing added to commit but untracked files present (use "git add" to track)
(base) → da-getting-started git:(new_branch) × █
```

Before adding and committing your changes, you might want to use *git pull* to get the latest version of the repo.

4.1. b) Add the files you have changed and want to push

In terminal type:

git add <file name>

This adds the file that you want to push to the remote repository to the 'staging area'.

NOT REALLY RECOMMENDED: With *git add .* OR *git add --all* you can add all files that are new or modified to the staging area. Just make sure that you really want to add them all (or check if your gitignore file is up to date) to avoid pushing the wrong file(s) or sensitive data to GitHub!

Use *git status* again to see what changed.

```
→ da-getting-started git:(new_branch) × git add test1.ipynb
→ da-getting-started git:(new_branch) ×
```

```
[(base) → da-getting-started git:(new_branch) × git status
On branch new_branch
Your branch is up to date with 'origin/new_branch'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   test1.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    pairings/speed_dating.py
    pairings/speeddating_copy.py
    test2.ipynb
    this_is_a_jupyter_notebook.ipynb

(base) → da-getting-started git:(new_branch) ×
```

4.1. c) Commit the changes

In terminal type:

git commit -m 'your message'

Commit the staged file(s) and add a message to tell others (or your future self) what you did. A commit is like a checkpoint. Use commits to track your changes.

Again, use *git status* to see what changed.

If you try to commit without a message, you will end up in a vim editor. Type *i* in order to be able to insert a message. After you typed a message, press “esc” on your keyboard, then type *:wq* and press “Enter” to save your commit message. More on Vim in this video: [Vim in 100 Seconds](#)

```
(base) → da-getting-started git:(new_branch) × git commit -m 'test message'
[new_branch d9a9c34] test message
1 file changed, 30 insertions(+)
create mode 100644 test1.ipynb
(base) → da-getting-started git:(new_branch) ×
```

```
((base) → da-getting-started git:(new_branch) × git status
On branch new_branch
Your branch is ahead of 'origin/new_branch' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    pairings/speed_dating.py
    pairings/speeddating_copy.py
    test2.ipynb
    this_is_a_jupyter_notebook.ipynb

nothing added to commit but untracked files present (use "git add" to track)
(base) → da-getting-started git:(new_branch) ×
```

4.1. d) Push to remote repository

In terminal type:

git push

This will push your added and committed files to GitHub.

If you pushed your changes in main, you're done.

If you used a branch other than main, then you have to go to GitHub to open a pull request (see next slides).

If you created your branch locally, then you also need to push the branch:

git push -u origin <branch name>

or

git push --set-upstream origin <branch name>

```
((base) → da-getting-started git:(new_branch) ✖ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 565 bytes | 565.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:konni-b/da-getting-started.git
   e8d2e6b..d9a9c34  new_branch -> new_branch
((base) → da-getting-started git:(new_branch) ✖
```

4.2 Open pull request and merge changes (if necessary)

When you collaborate with other people in a project, you most likely worked in your own branch.

If you have changes that should be merged into main, you need to open a pull request, so your colleagues can compare the changes to the current version of the file(s).

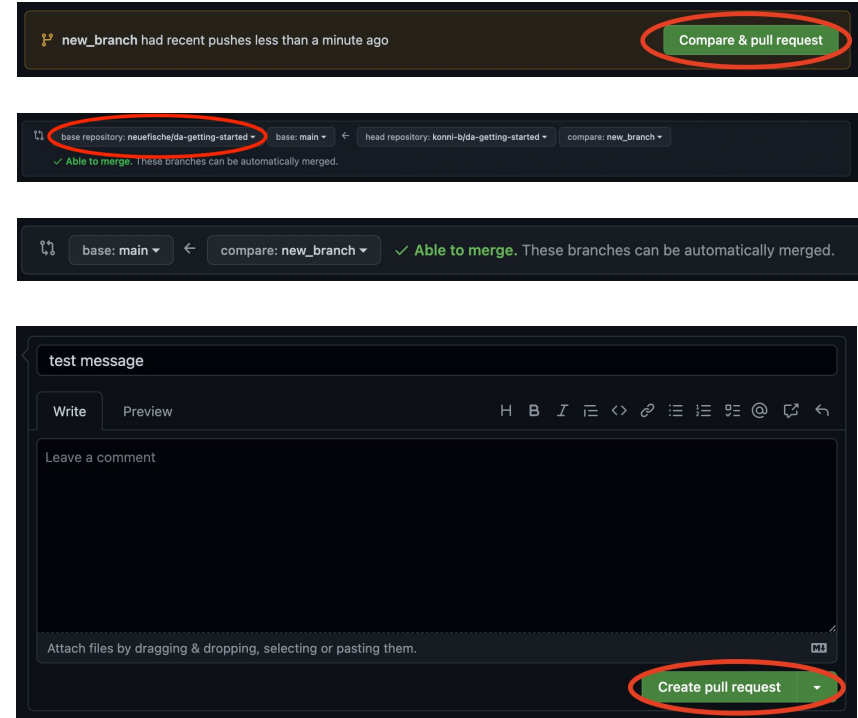
Your reviewers can approve, comment or request changes.

Once everyone agrees with the changes, you can merge them into main and you're done.

4.2. a) When you push in a branch other than main, you need to open a pull request

- After you push the changes, go to GitHub to the remote repository and open the pull request: **Compare & pull request.**
- If you see another base repository than the one you actually want to push the changes to, than change (in this case the neufische repository marked in red) to the one you really want to open the pull request in!
- Then you can create the pull request: **Create pull request.**

The pull request is creating a comparison between main and the new_branch to enable the review.

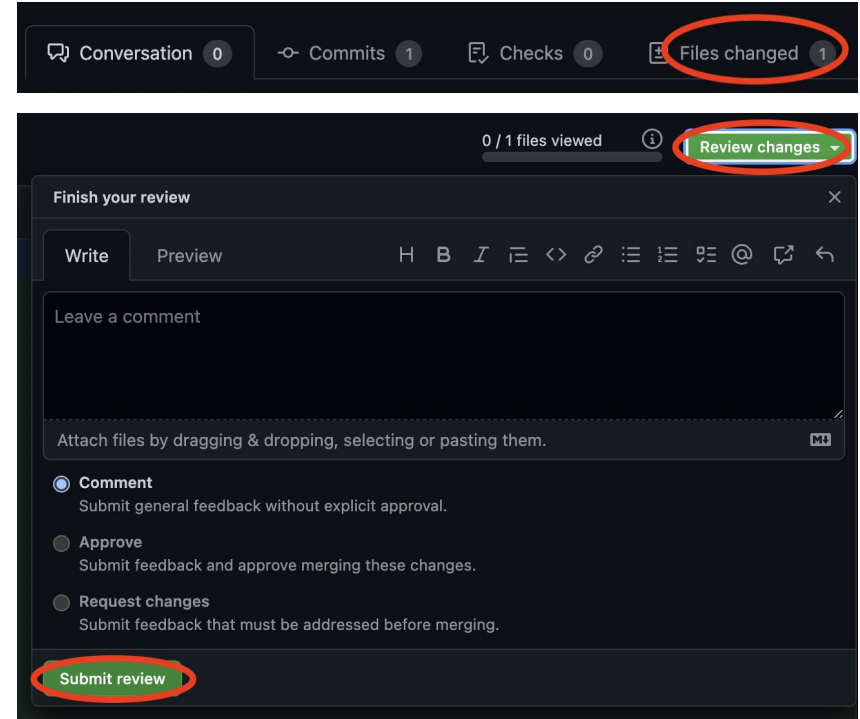


4.2. b) Now your pull request gets reviewed

- Under 'Files changed' you can review the changes that were made to the file.
- Under 'Review changes' you can either:
 - Comment
 - Approve
 - Request changes
- If the reviewers are okay with the changes then they have to select 'Approve' and press the 'Submit review' - button
- You can find more information on pull requests and how to handle requested changes [here](#).

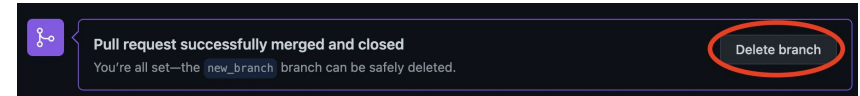
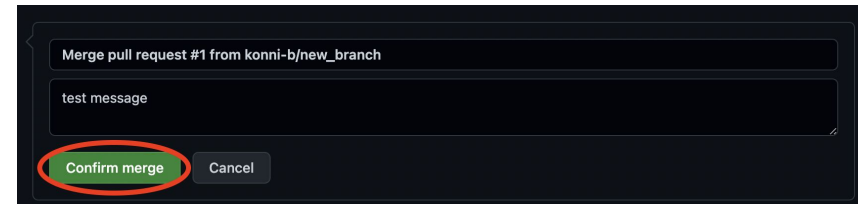
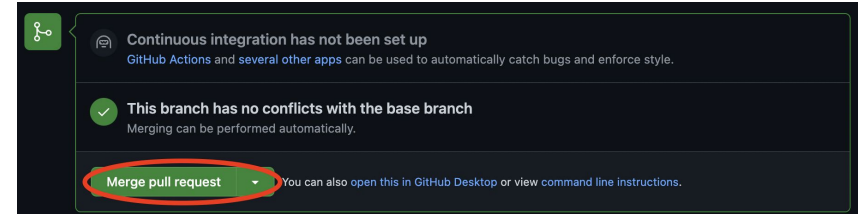
Tip:

- If you have more than one comment, instead of making multiple single comments to the changed file, **start a review** as shown in this [video](#).



4.2. c) Merge the pull request after approval

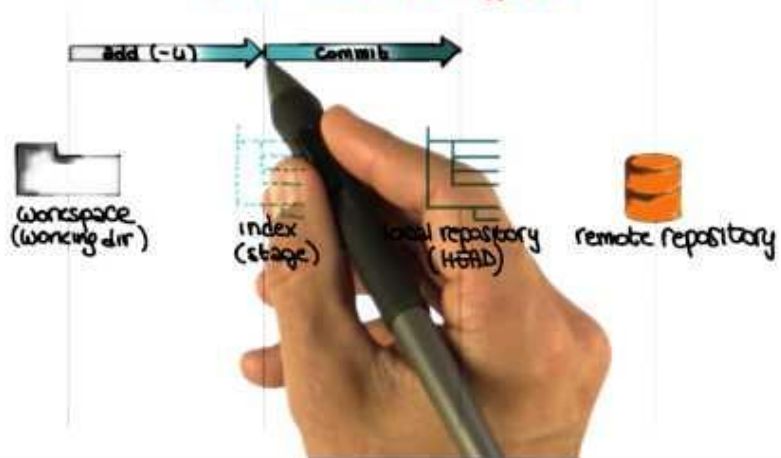
- When your pull request was approved, you can merge it (and confirm the merge).
- Then you can delete your branch.
- Go back to your terminal and type *git switch main* to return to the main branch.
- To get the newest version/ all changes of a remote repository, that you cloned locally, use *git pull* to get the changes.



```
(base) → da-getting-started git:(new_branch) × git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
(base) → da-getting-started git:(main) ×
```

More Resources:

GIT WORKFLOW



100 seconds of

git

STARTED!



Even more resources:

If you want to know how to initialize a local repository and the workflow to get it on GitHub, follow along this video:

