



**Karolinska
Institutet**

Using Git/Github in Scientific Collaborations

Mikkel C. Vinding, PhD

Assistant Professor

NatMEG, CNS, Karolinska Insitutet

Email: mikkel.vinding@ki.se

@mc_vinding



NatMEG



The Swedish National Facility for Magnetoencephalography

Program

- **15:00-15:45 lecture**
- **15:45-16:00 Short Q&A**
- **16:00-17:00 Hands-on tutorial**

*Disclaimer: I have no affiliation with Git or GitHub, Inc.,
All materials are for non-profit educational purposes only. All materials in this repository are free and open for fair use under the CC license.*

Code example



```
$ git pull origin master
```

What is Git and Github?



- Tool for distributed version control
- Free and open source
- <https://git-scm.com/>



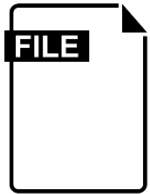
- Hosting service and interface for Git
 - Owned by Microsoft, but free*
 - Options for project management, collaboration, wikis, and more
 - <https://github.com/>
-

GitHub for scientific collaboration

1. Organize your data analysis scripts



My Project



preproc



stats



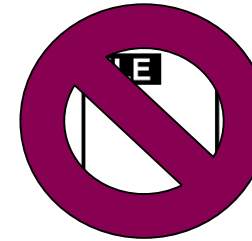
functions



more_func



follow-up



misc_test



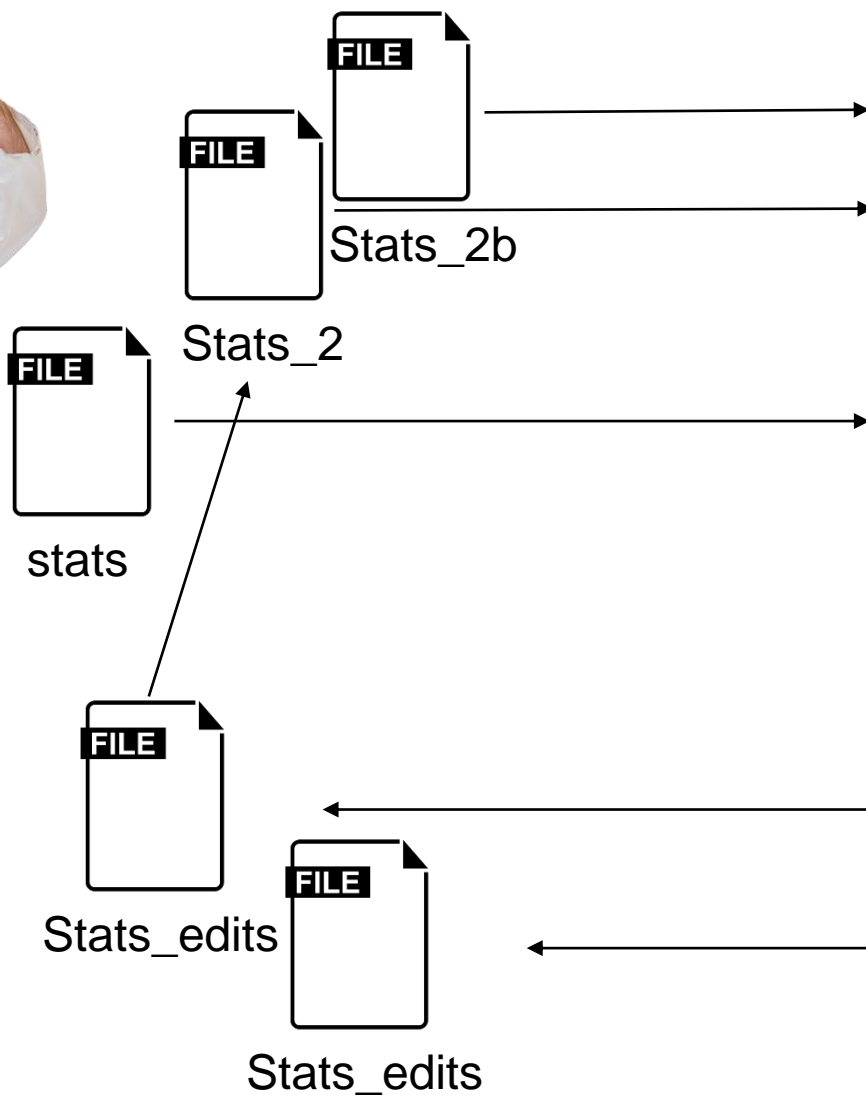
preproc2



functions_new

GitHub for scientific collaboration

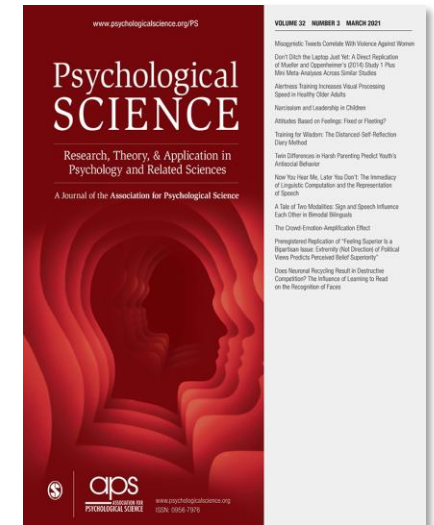
1. Organize your data analysis scripts
2. Easy to collaborate on tasks that requires programming (i.e. all neuroscientific data analysis)



Sorry, what
version?

GitHub for scientific collaboration

1. Organize your data analysis scripts
2. Easy to collaborate on tasks that requires programming (i.e. all neuroscientific data analysis)
3. Share finished analysis scripts



What to put on GitHub?

- Your scripts...
- Code in development
- Code to reproduce analysis of finished projects

Article | [Open Access](#) | Published: 22 February 2019

Attenuated beta rebound to proprioceptive afferent feedback in Parkinson's disease

Mikkel C. Vinding , Panagiota Tsitsi, Harri Piitulainen, Josefine Waldthaler, Veikko Jousmäki, Martin Ingvar, Per Svenningsson & Daniel Lundqvist

Scientific Reports **9**, Article number: 2604 (2019) | [Cite this article](#)

1340 Accesses | **3** Citations | **9** Altmetric | [Metrics](#)



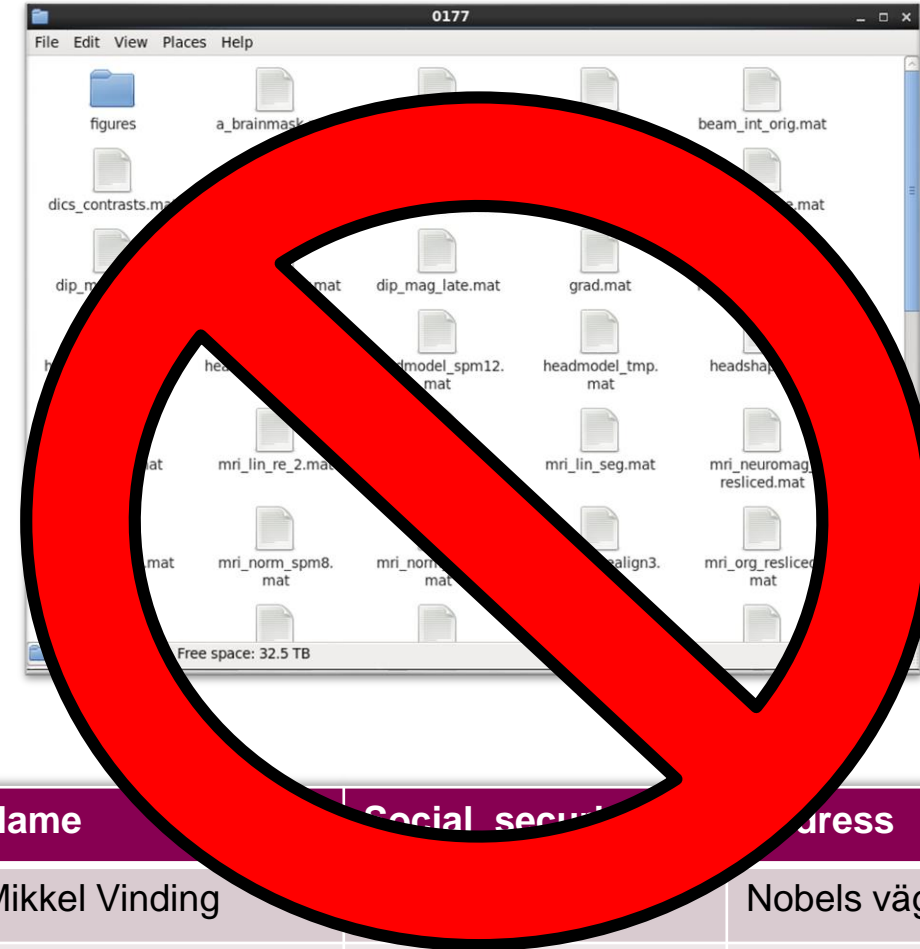
 [PD_proprioceptive_beta](#) 

Dataanalysis of beta-band induced responses measured by magnetoencephalography (MEG) during passive movements of the index finger in Parkinson's patients and healthy controls

 MATLAB

What NOT to put on GitHub

- Datafiles
 - Maximum file size is 100MB
 - Maximum repository size is 10GB
- Sensitive information



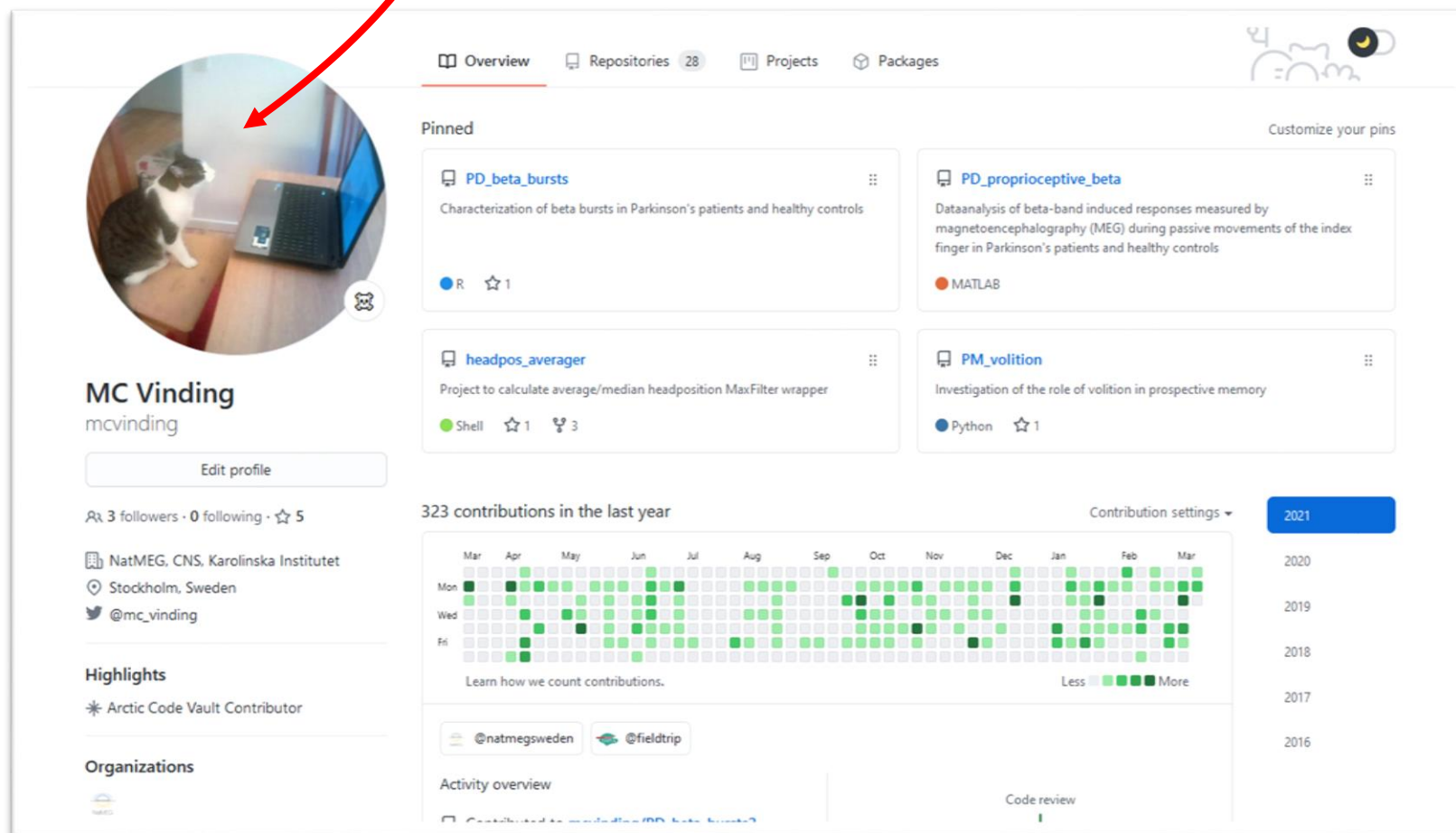
Subjid	Name	Social security number	Address
0001	Mikkel Vinding		Nobels väg 9
0002	...		

GitHub accounts

Personal account

- "your code"
 - Research projects
 - tools

Me

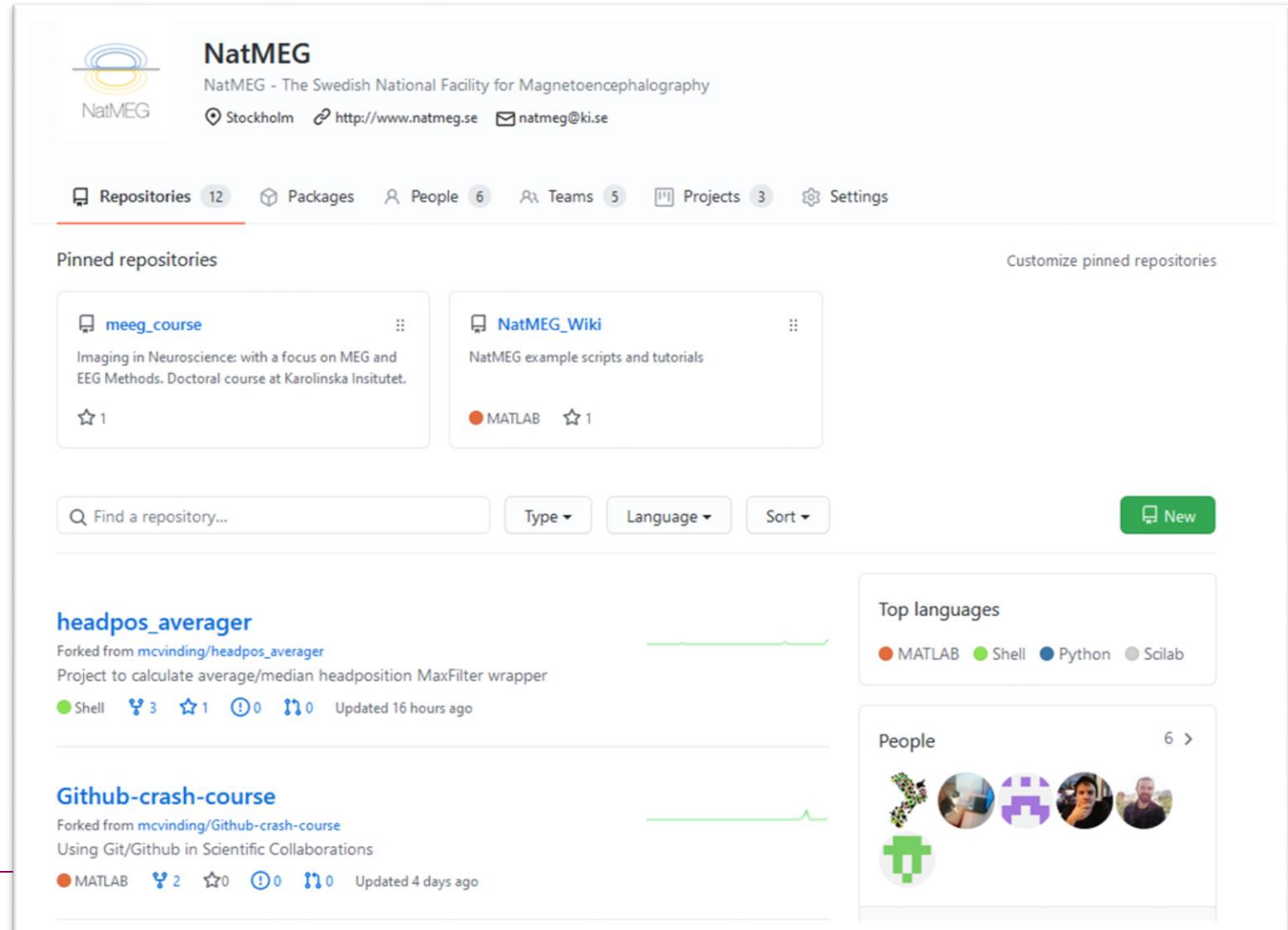


The screenshot shows the GitHub profile of MC Vinding (mcvinding). The profile picture is a circular image of a cat sitting at a desk with a laptop. A red arrow points from the word "Me" to this profile picture. The profile information includes 3 followers, 0 following, and 5 stars. The bio mentions "NatMEG, CNS, Karolinska Institutet" and "Stockholm, Sweden". The Twitter handle is @mc_vinding. The highlights section lists "Arctic Code Vault Contributor". The organizations section shows "NatMEG". The pinned repositories section displays four projects: "PD_beta_bursts" (R, 1 star), "PD_proprioceptive_beta" (MATLAB), "headpos_averager" (Shell, 1 star, 3 forks), and "PM_volition" (Python, 1 star). The contributions section shows a grid of 323 contributions in the last year, with a calendar view from March 2020 to March 2021. The activity overview section shows a list of recent contributions.

GitHub accounts

Organizations

- Research group or lab
 - Research projects
 - "Lab hacks"
 - Tools

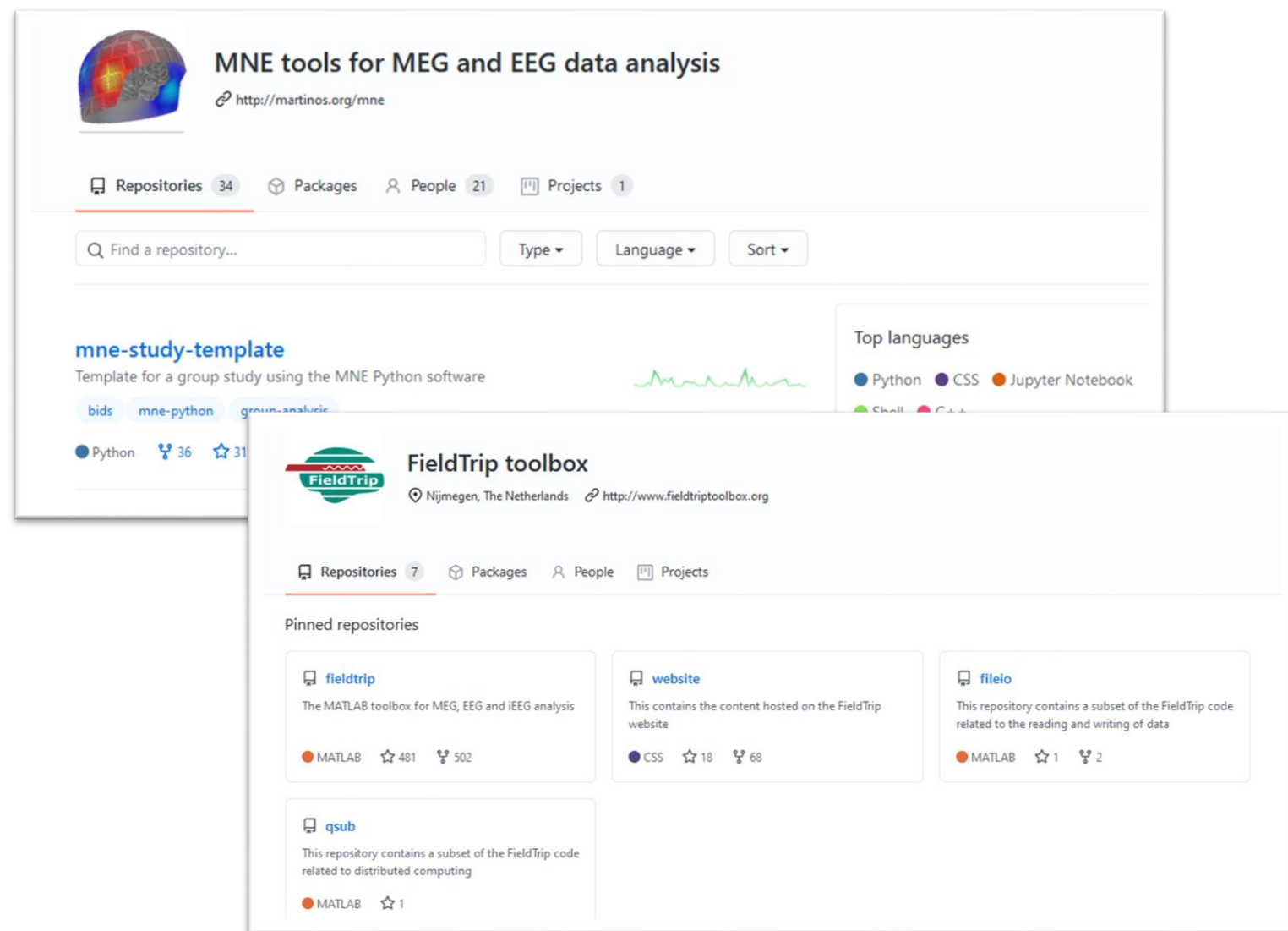


The screenshot displays the GitHub profile for the NatMEG organization. At the top, the organization's name "NatMEG" is shown alongside its logo, which consists of concentric blue and yellow circles. Below the name, a brief description reads "NatMEG - The Swedish National Facility for Magnetoencephalography". Contact information for Stockholm, the website <http://www.natmeg.se>, and the email natmeg@ki.se is provided. Navigation tabs for Repositories (12), Packages, People (6), Teams (5), Projects (3), and Settings are visible. The "Pinned repositories" section features two items: "meeg_course", described as an imaging neuroscience course, and "NatMEG_Wiki", containing example scripts and tutorials. Below this is a search bar and filters for repository type, language, and sort order. The main repository list shows "headpos_averager", a project for calculating headposition, and "Github-crash-course", a guide for scientific collaborations. On the right, a "Top languages" section lists MATLAB, Shell, Python, and Scilab, and a "People" section shows the profiles of the organization's members.

GitHub accounts

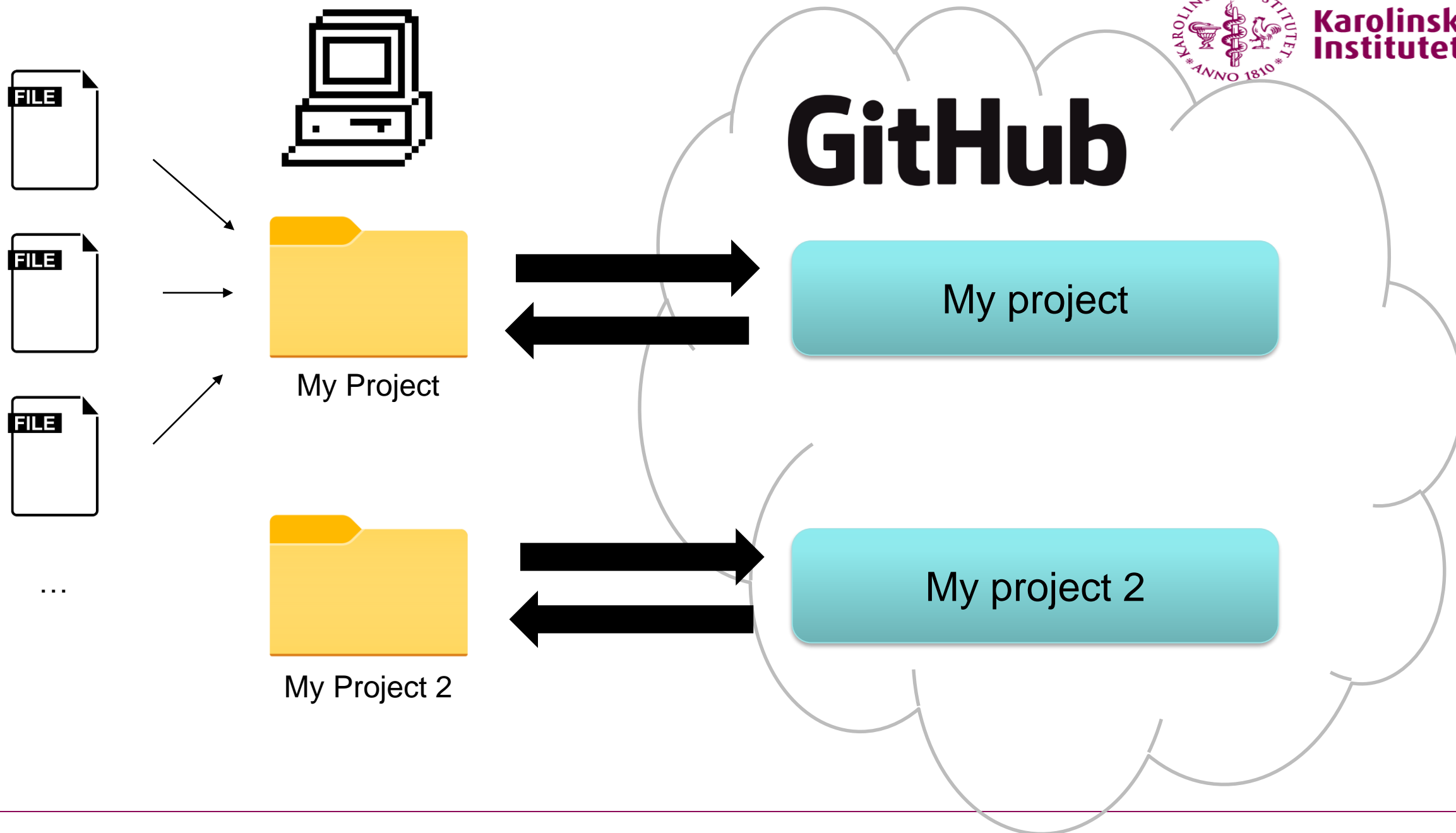
Organizations

- Research group or lab
 - Research projects
 - "Lab hacks"
 - Tools
- Toolboxes
 - Get code from developers
 - Contribute



How to use Git and GitHub for your project?

GIT WORKFLOW

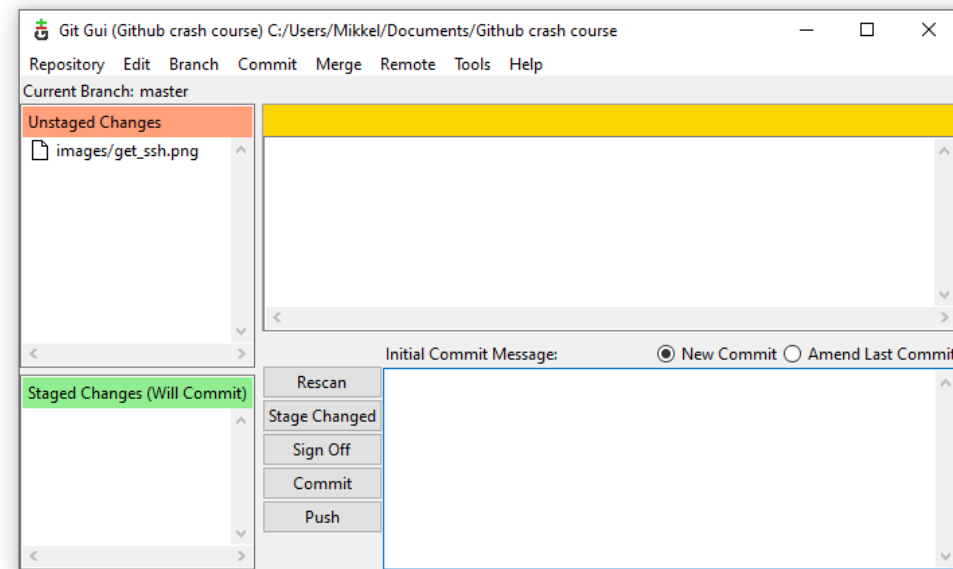


Using Git: the basics

```
$ git <command> <option(s)> <files> <...>
```


Do I have to use the terminal?

Yes*



*No: There is a GUI

Initiating a project

START A GIT PROJECT

Getting started...

Start a new project

Create a repository for your project.

Where you want to start!

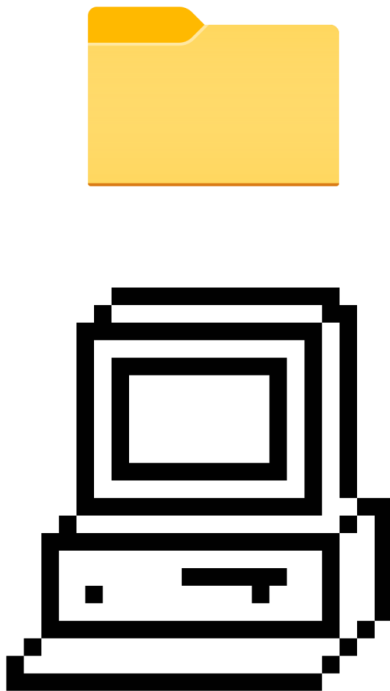
Join a project

E.g., a project where someone already have created a repository

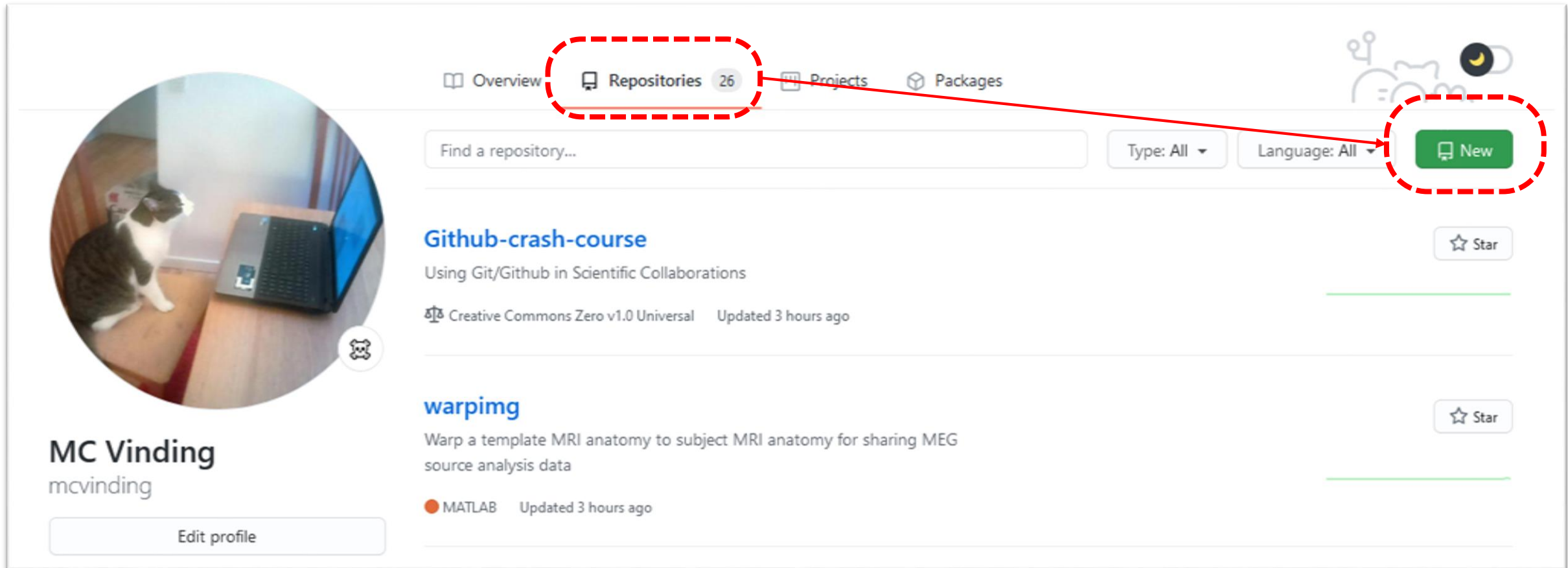
Clone a project

Get code from another project, or an analysis toolbox

Start a new project...



Start a new project...



Overview **Repositories 26** Projects Packages

Find a repository... Type: All Language: All **New**

Github-crash-course
Using Git/Github in Scientific Collaborations
Creative Commons Zero v1.0 Universal Updated 3 hours ago Star

warping
Warp a template MRI anatomy to subject MRI anatomy for sharing MEG source analysis data
MATLAB Updated 3 hours ago Star

MC Vinding
mcvinding
Edit profile


Create a new repository



Karolinska
Institutet

1. Name



Owner * Repository name *

 mcvinding /

Great repository names are short and memorable. Need inspiration? How about [glowing-octo-pancake?](#)

Description (optional)

2. Define visibility

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

3. Add files

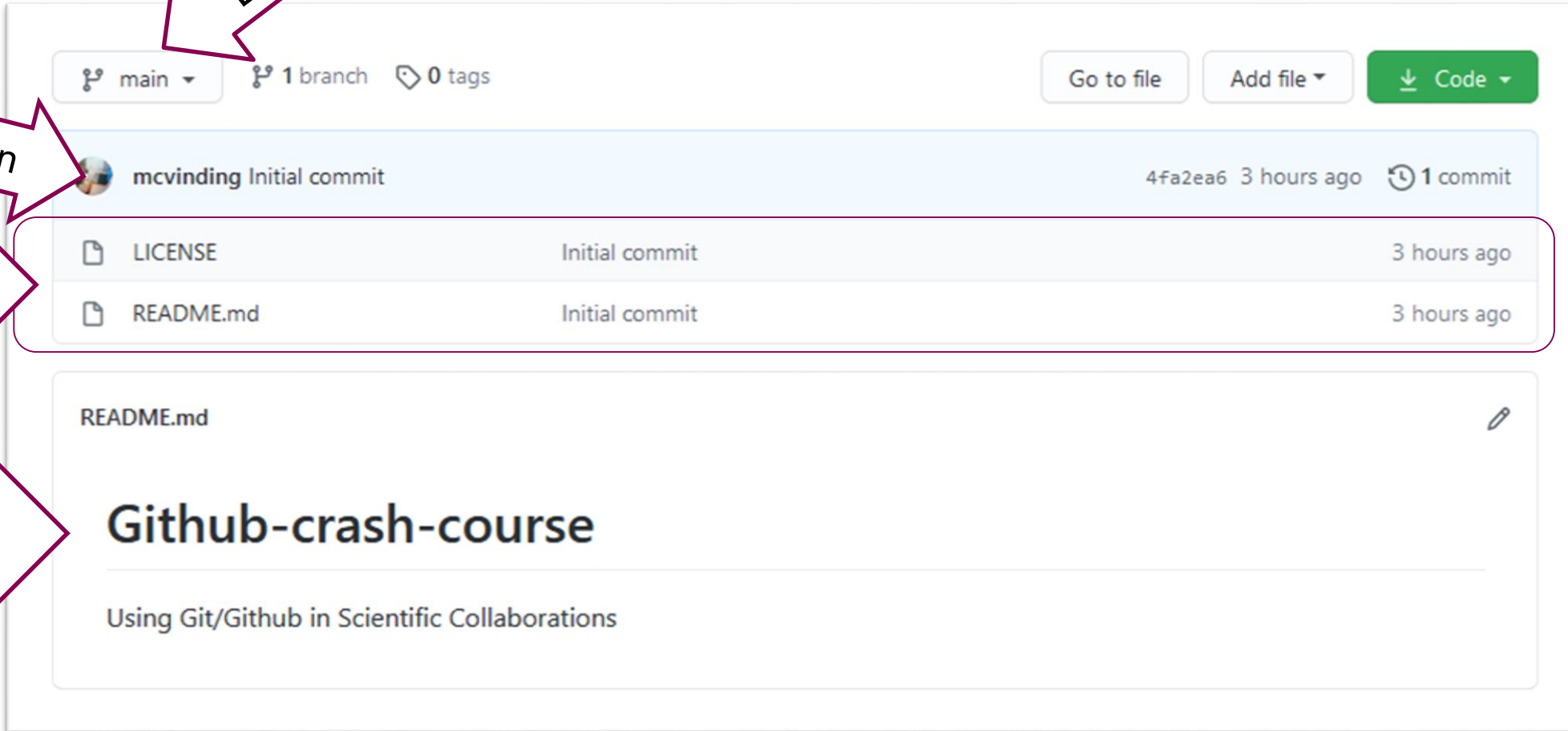
Initialize this repository with:

Skip this step if you're importing an existing repository.

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)
- ☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)
- ☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

4. Create

Create repository

A screenshot of a GitHub repository page for "Github-crash-course" by user "mcvinding". The page shows the "main" branch with 1 branch and 0 tags. The commit history shows an "Initial commit" by "mcvinding" 3 hours ago. The file list shows "LICENSE" and "README.md", both from the "Initial commit" 3 hours ago. The README content is titled "Github-crash-course" and describes "Using Git/Github in Scientific Collaborations". Annotations with arrows point to specific elements: "Branch" points to the "main" branch selector; "Current version" points to the commit information; "Your files" points to the file list; and "README.md" points to the README content.

Branch

main 1 branch 0 tags

Go to file Add file Code

mcvinding Initial commit 4fa2ea6 3 hours ago 1 commit

LICENSE	Initial commit	3 hours ago
README.md	Initial commit	3 hours ago

README.md

Github-crash-course

Using Git/Github in Scientific Collaborations

Current version

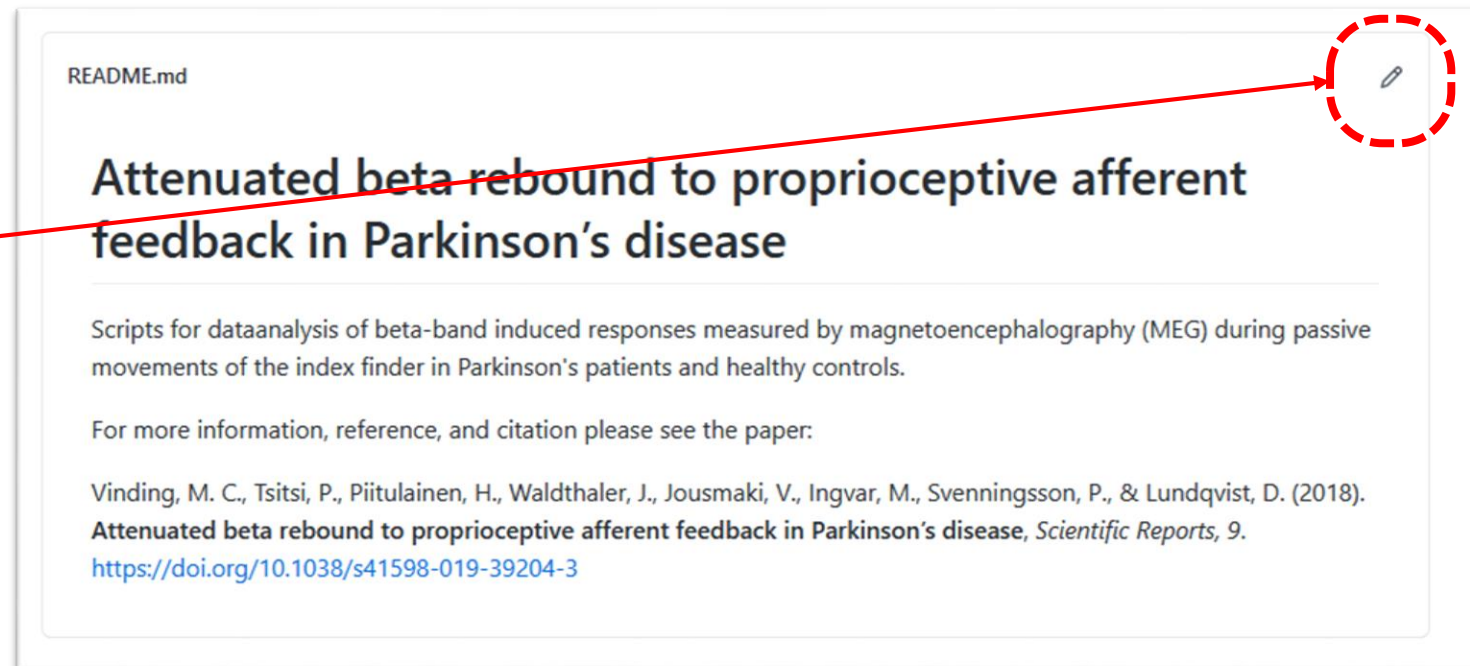
Your files

README.md

The README.md file

- Write *informative text* that helps people who find your repository
 - Collaborators...
 - Reviewers...
 - Public...
- Edit in text editor or in browser
- Markdown syntax

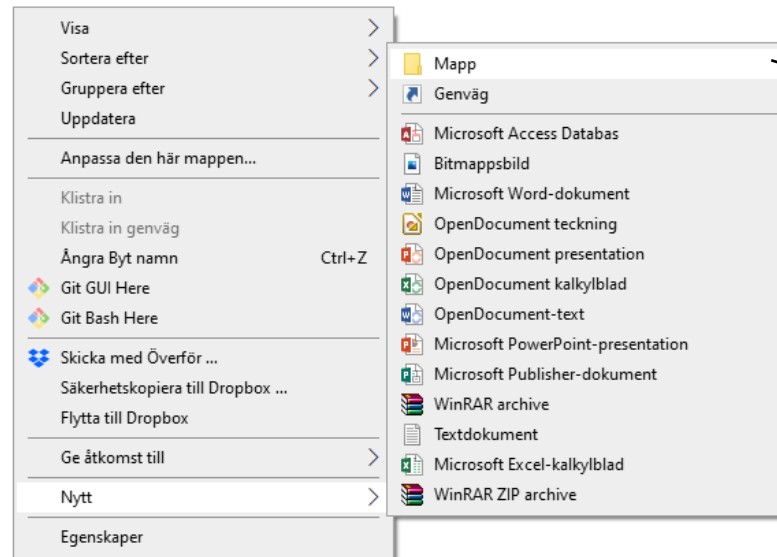
Nice Markdown cheatsheet:
<https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>



Start a new project...

1. Create the local folder at the desired location

```
$ mkdir my_project
```



My Project

Start a new project...

1. Create the local folder at the desired location
2. Go to folder

```
$ cd my_project
```



My Project

Start a new project...

1. Create the local folder at the desired location
2. Go to folder
3. **Initialize** folder

```
$ git init
```

```
Initialized empty Git repository in  
C:/Users/Mikkel/Documents/Github crash course/.git/
```



My Project

Start a new project...

1. Create the local folder at the desired location
2. Go to folder
3. **Initialize** folder

```
$ git init  
Initialized empty Git repository in  
C:/Users/Mikkel/Documents/Github crash course/.git/
```



My Project

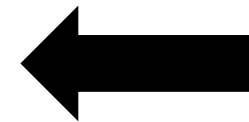
Start a new project...

1. Create the local folder at the desired location
2. Go to folder
3. **Initialize** folder
4. Set **remote**

GitHub



My Project

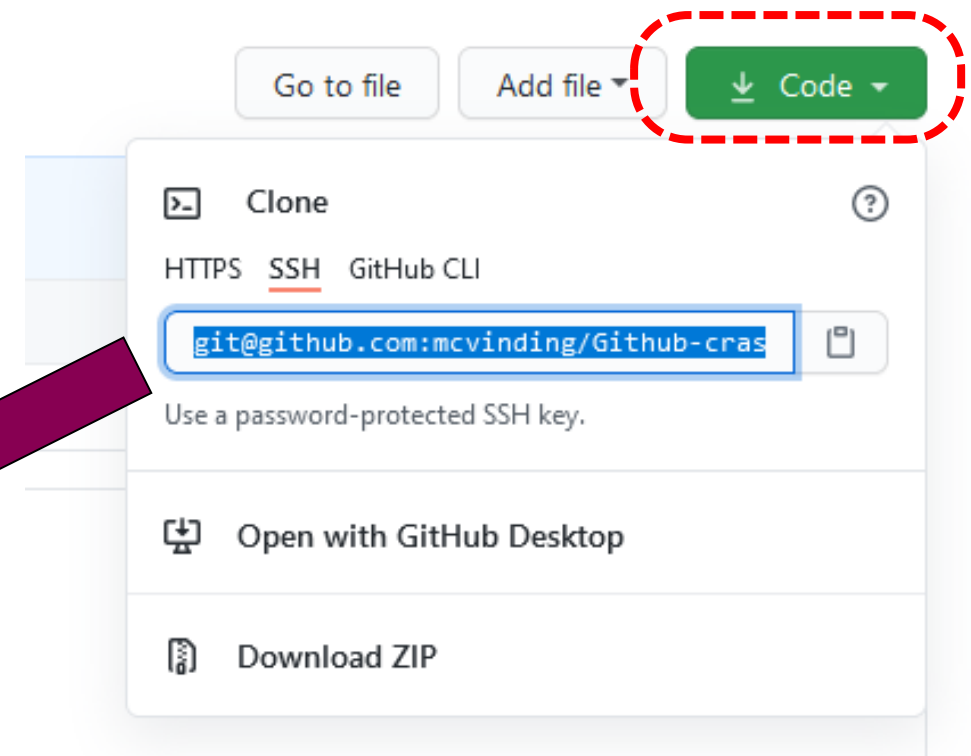


My Project

Start a new project...

1. Create the local folder at the desired location
2. Go to folder
3. **Initialize** folder
4. Set **remote**

```
$ git remote add origin <address>
```

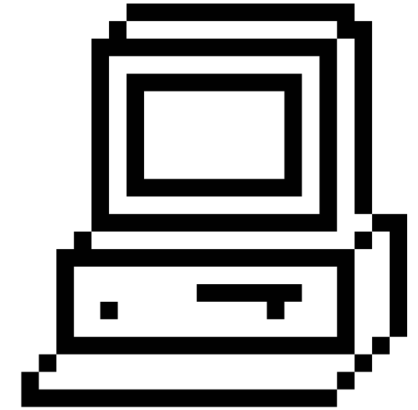


Start a new project...

1. Create the local folder at the desired location
2. Go to folder
3. **Initialize** folder
4. Set **remote**

```
$ git remote add origin <address>
```

GitHub



My Project
"origin"



My Project



See remote address

```
$ git remote -v  
origin  git@github.com:mcvinding/Github-crash-course.git (fetch)  
origin  git@github.com:mcvinding/Github-crash-course.git (push)
```


Start a new project...

1. Create the local folder at the desired location
2. Go to folder
3. **Initialize** folder
4. Set **remote**
5. **Pull** files from remote

```
$ git pull origin main
```

"Branch"

GitHub



My Project
"origin"



My Project

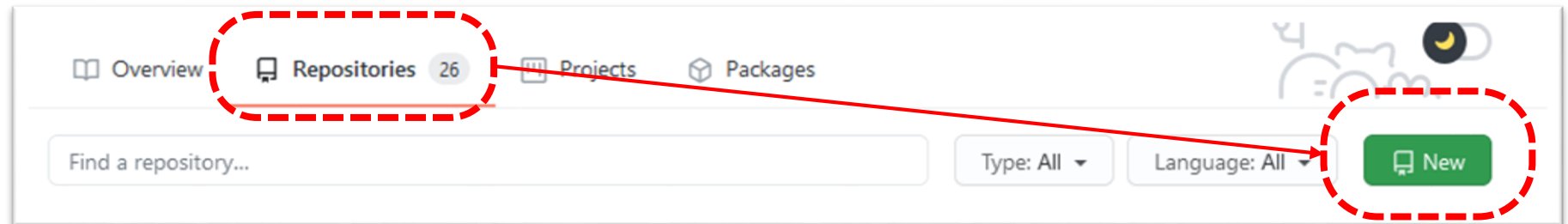


Get files

```
$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From github.com:mcvinding/Github-crash-course
* branch                main          -> FETCH_HEAD
```

Summary: initiating a Git project

GitHub



```
$ git init  
$ git remote add origin <address>  
$ git pull origin main
```

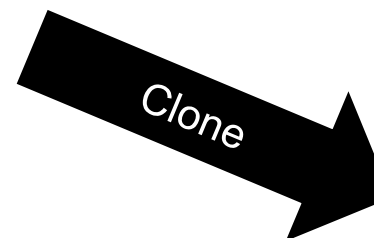
Git clone

```
$ git clone <address>
```

GitHub



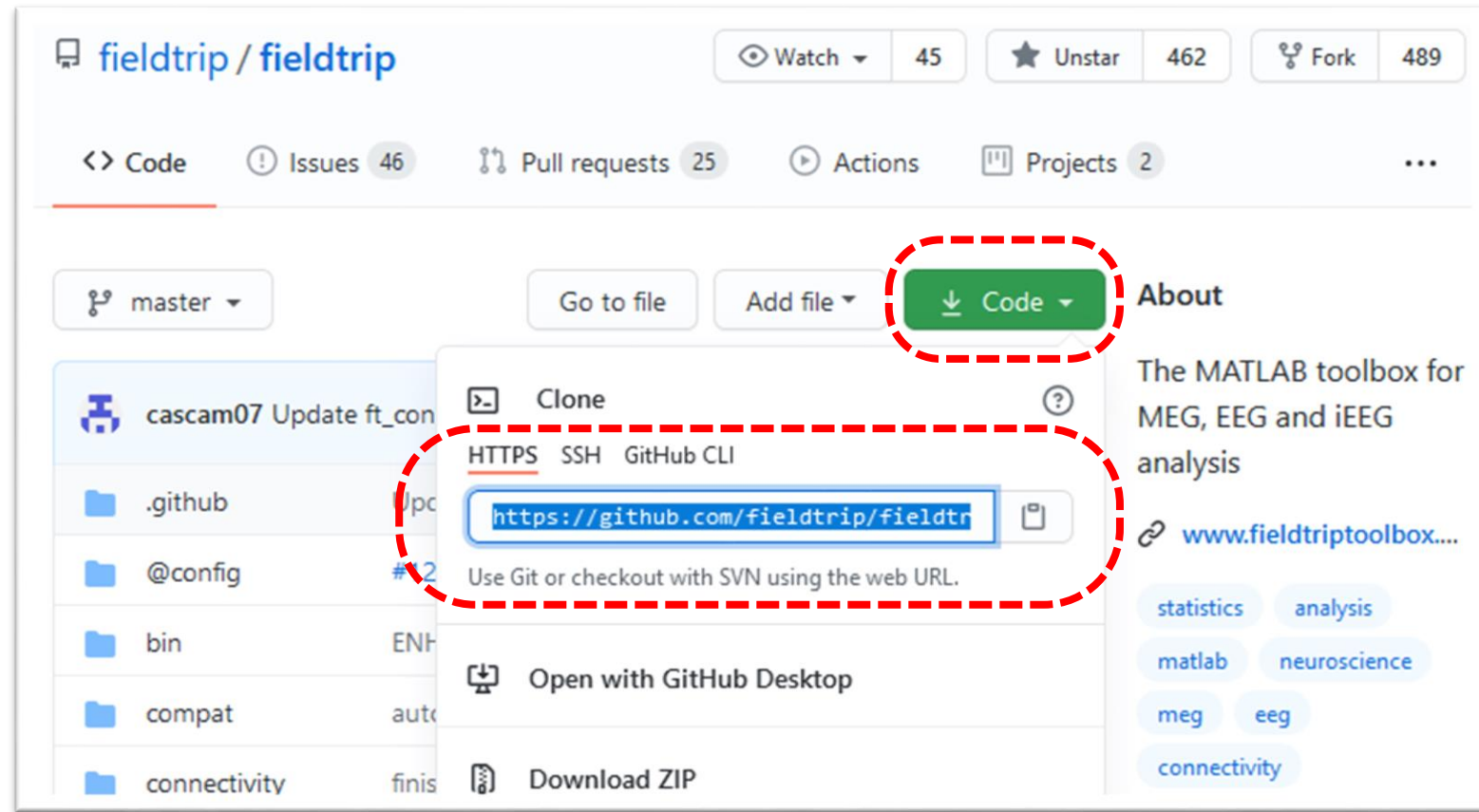
Any GitHub repo



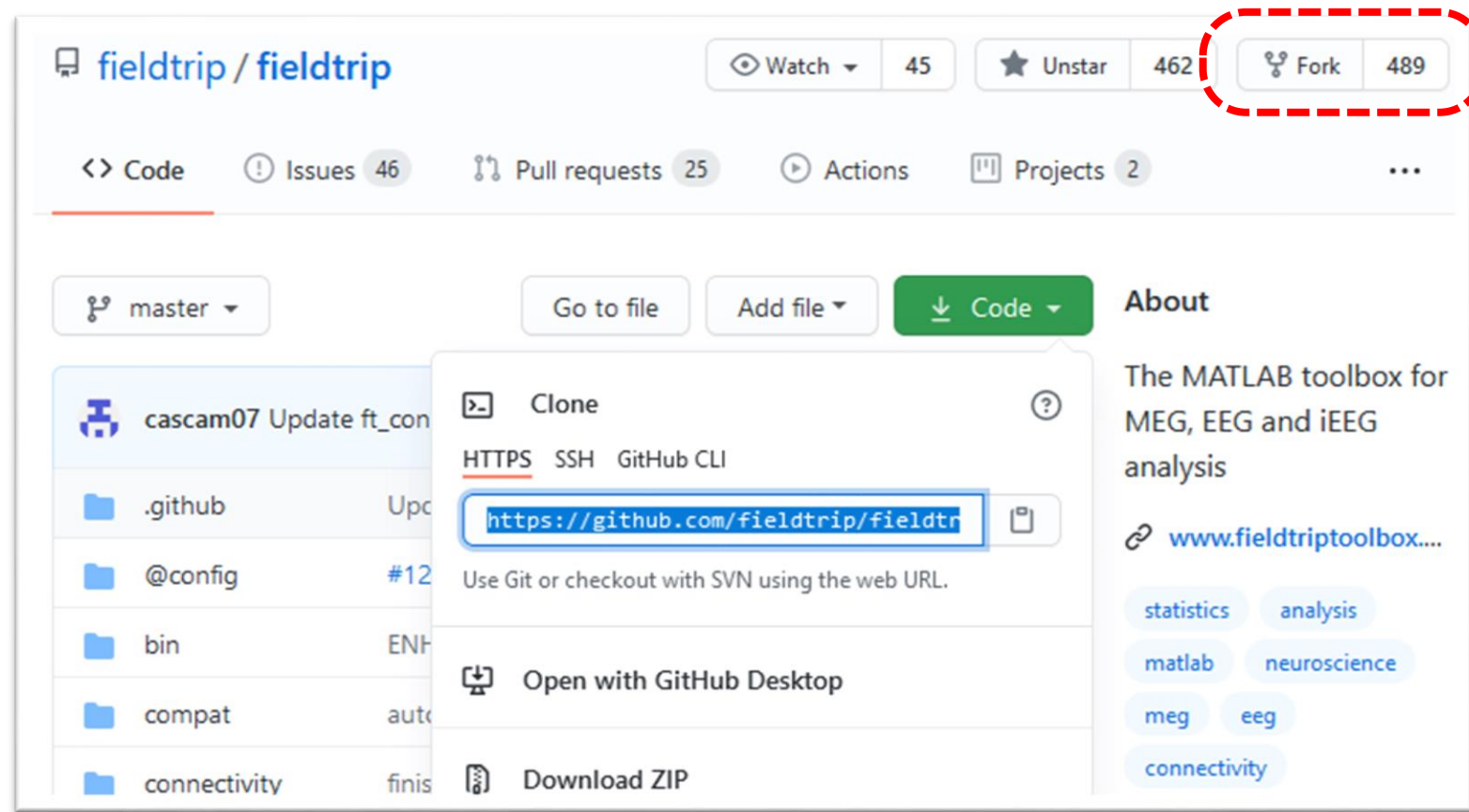
My local copy

Git clone

```
$ git clone <address>
```



Fork repository

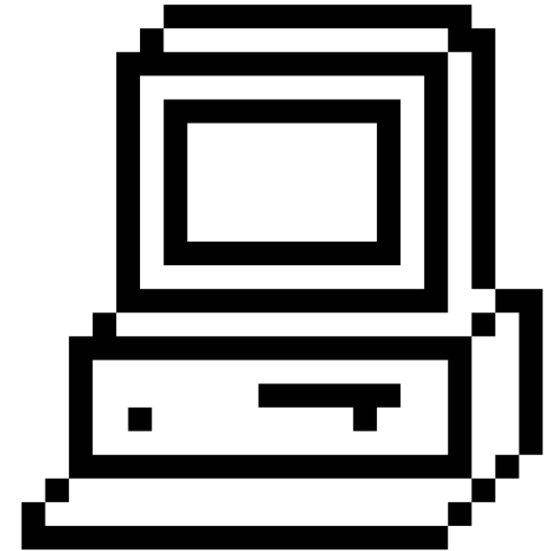
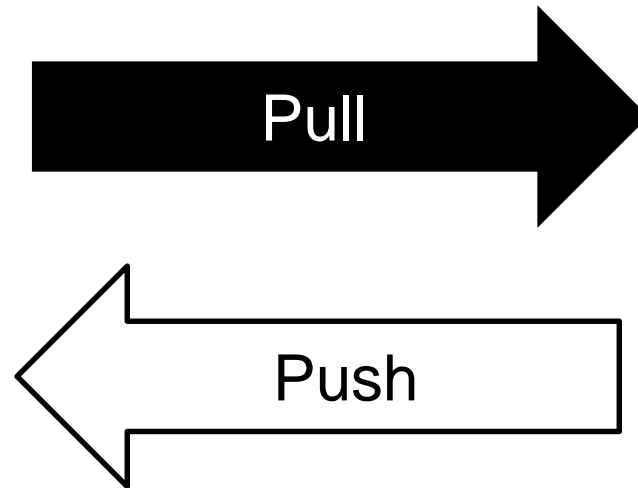


Pull, commit, push, and win at code management

WORKING WITH GIT REPOSITORIES

Git terminology

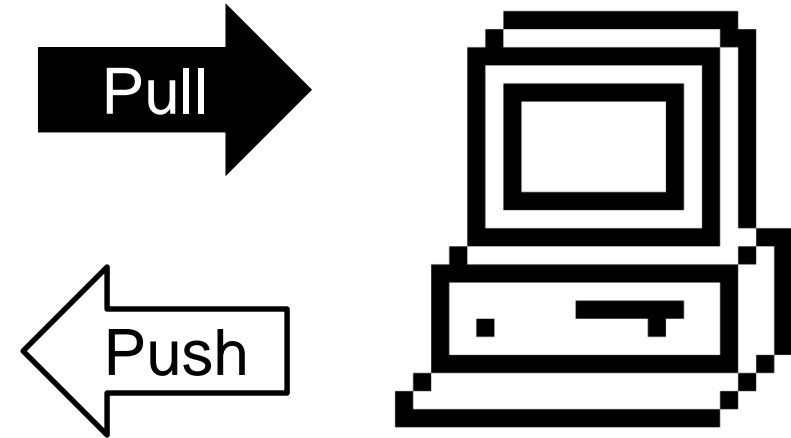
GitHub



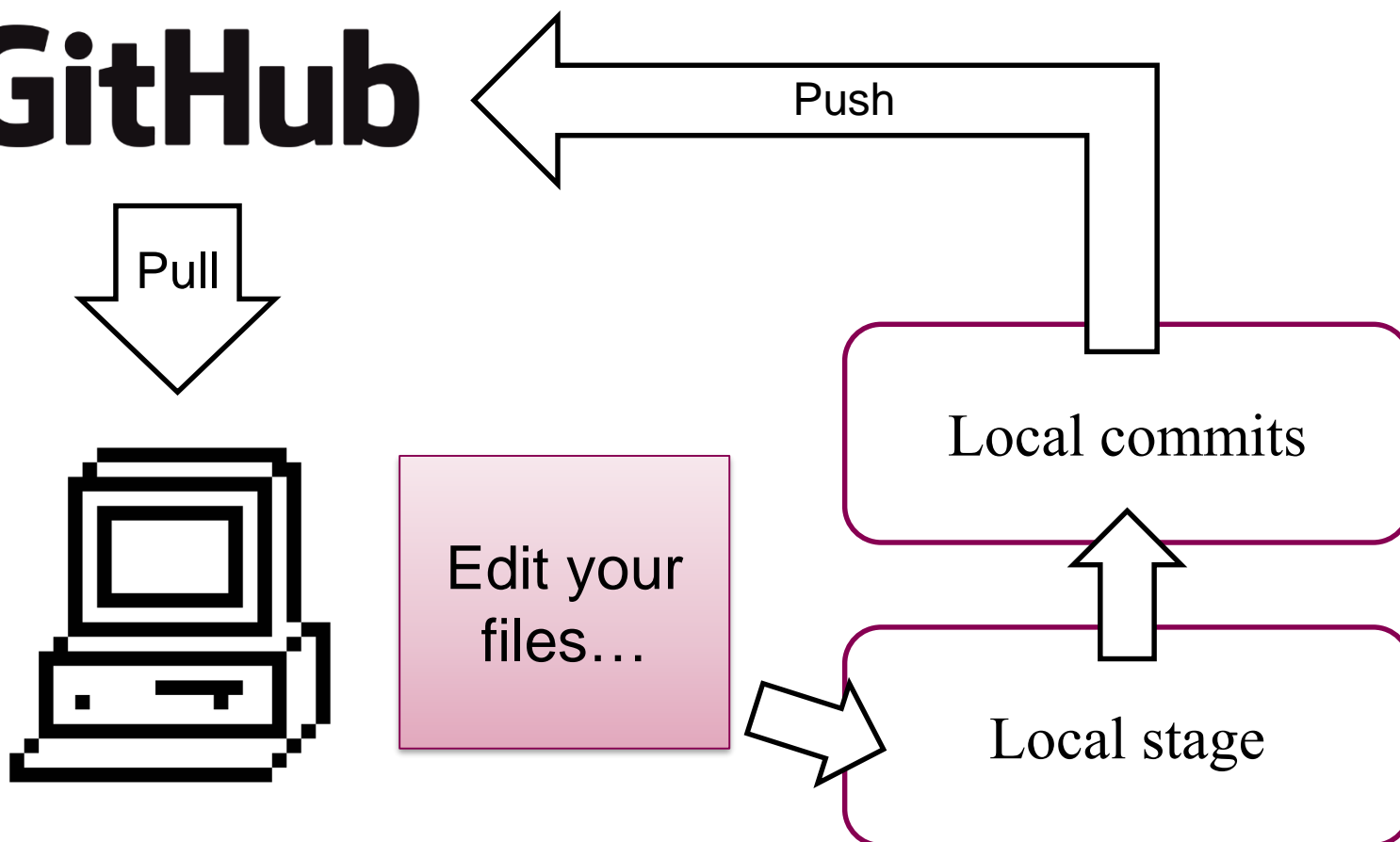
Git workflow at a glance

- **Pull** latest code
- Write your code...
- Stage edits
- Commit changes
- **Push**

GitHub



GitHub



```
$ git pull origin main
```

```
$ git add <options>
```

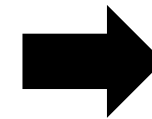
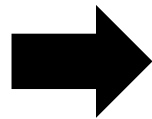
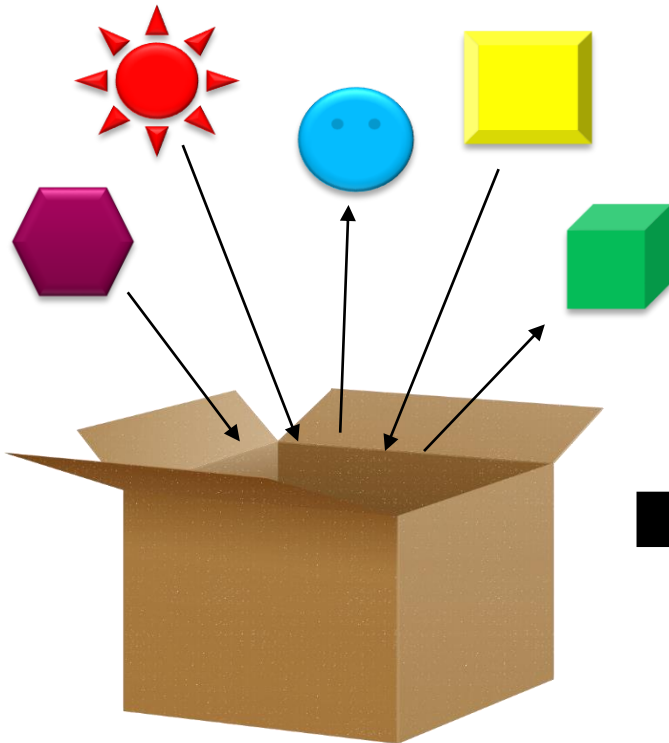
```
$ git commit
```

```
$ git push <remote> <branch>
```

Stage

Commit

Push



GitHub

Pull



Edit your
files...

Local stage

```
$ git add <filename1> <filename2> <...>
```



myFile1.m



myFile2.py



myFile3.txt

The most used Git command

```
$ git status
```

➡ `$ git status`
On branch main
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)

➡ `new file: new_file.txt`

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)

➡ `modified: another_new_file.txt`

➡ Untracked files:
(use "git add <file>..." to include in what will be committed)

➡ `images/`

Local stage



"Unstaged"

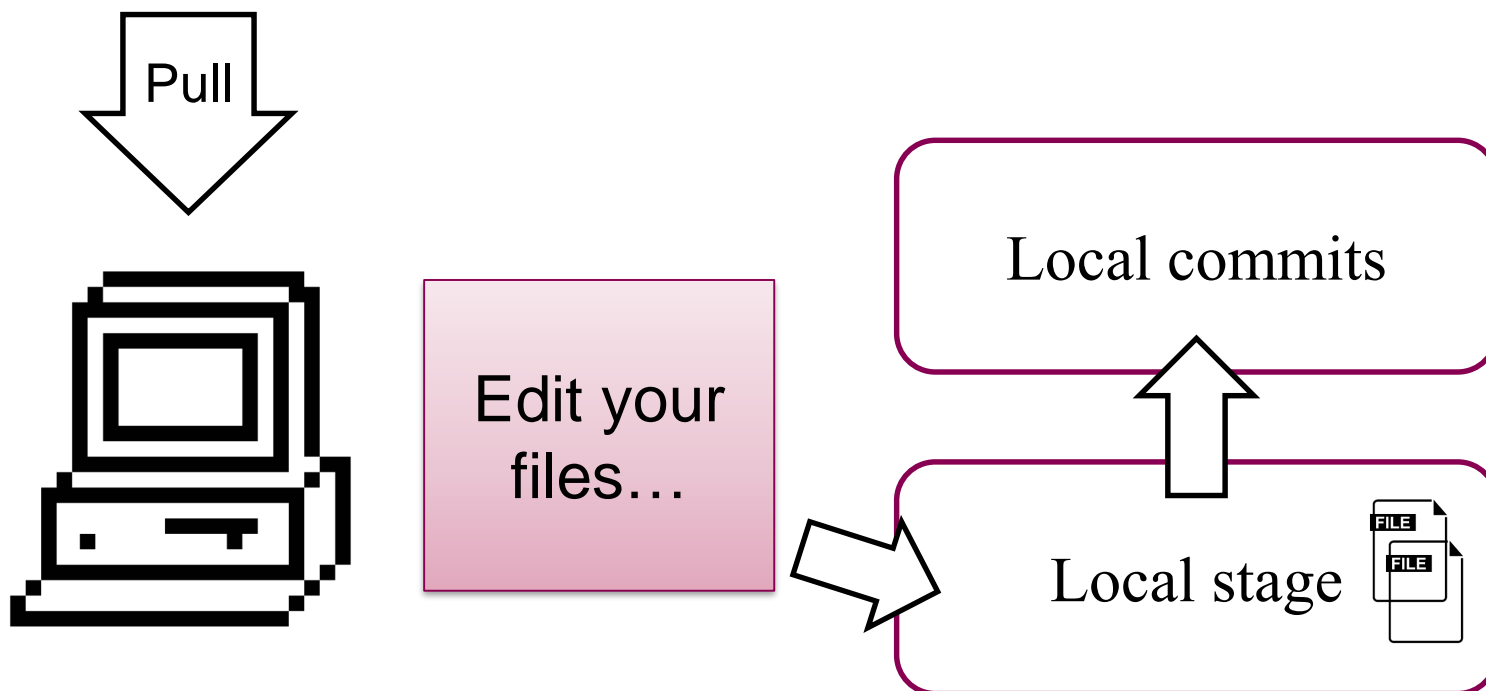
"Untracked"

Specific files

```
$ git add <filename1> <filename2> <...>  
$ git add -u
```

All tracked but unstaged files

GitHub

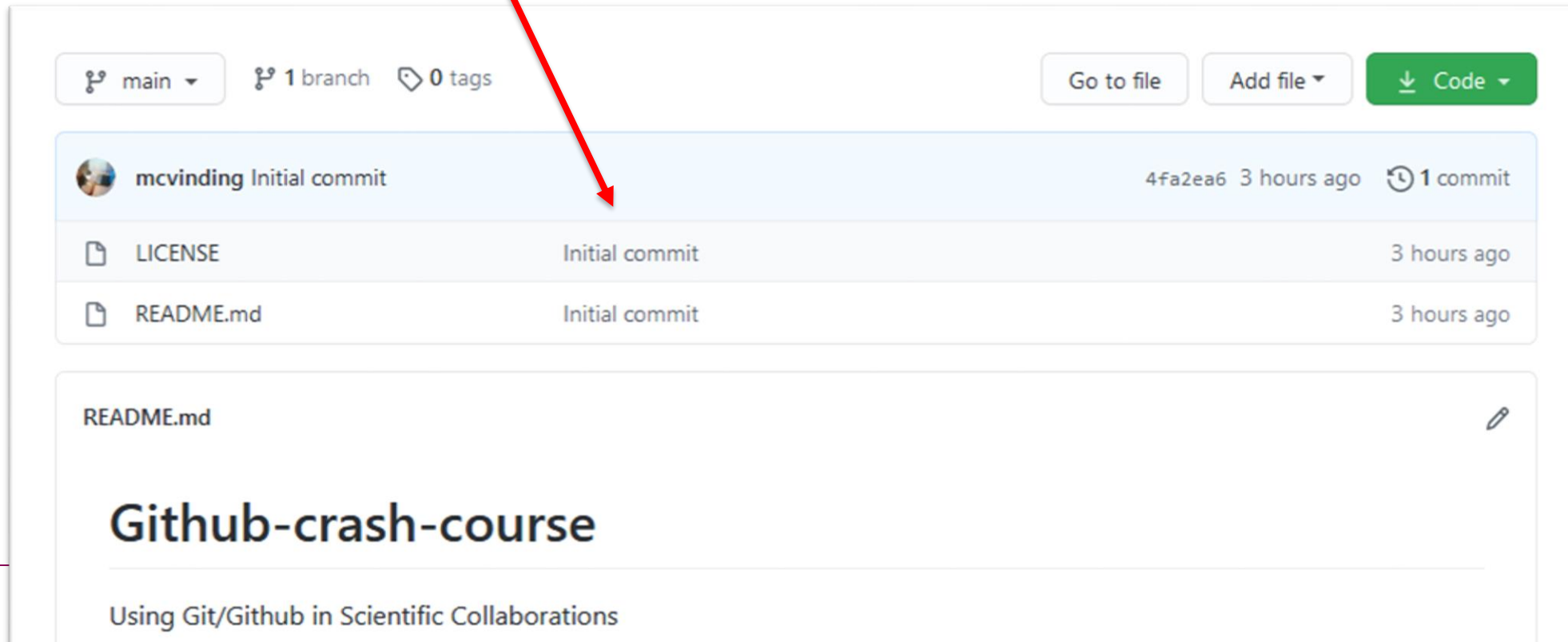


```
$ git commit -m "<...>"
```


Commit messages

```
$ git commit -m "<your message>"
```

Write a short informative
text about changes



The screenshot shows a GitHub repository page for 'Github-crash-course'. At the top, there are buttons for 'Go to file', 'Add file', and 'Code'. Below this, a commit by 'mcvinding' is shown with the message 'Initial commit'. The commit hash is '4fa2ea6' and it was made '3 hours ago'. Below the commit, a list of files is shown: 'LICENSE' and 'README.md', both with the message 'Initial commit' and '3 hours ago'. At the bottom, the 'README.md' content is visible, starting with 'Github-crash-course' and 'Using Git/Github in Scientific Collaborations'.

File	Commit Message	Time
LICENSE	Initial commit	3 hours ago
README.md	Initial commit	3 hours ago

Commit messages

```
$ git commit -m "<your message>"
```

Write a short informative
text about changes

Too little

"Some changes"

"Committed stuff"

OK

"changed filter settings"

"added files for plots"

"overhaul of pre-processing"

Too much

*"changed filter settings in line
42-46 and added a bunch
more options for processing
and new scripts for plots with
pretty colours that look like a
pretty flower"*

GitHub

Pull

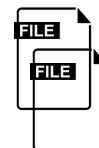


Edit your
files...

Push

Local commits

Local stage



```
$ git push <remote> <branch>
```

GitHub

Pull

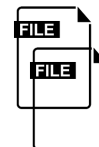


Edit your
files...

Push

Local commits

Local stage



```
$ git push origin main
```

Git workflow

- **Pull** latest code
- Write your code...
- Stage edits
- Commit changes
- **Push**

```
$ git pull origin main
```

```
$ git add <options>
```

```
$ git commit
```

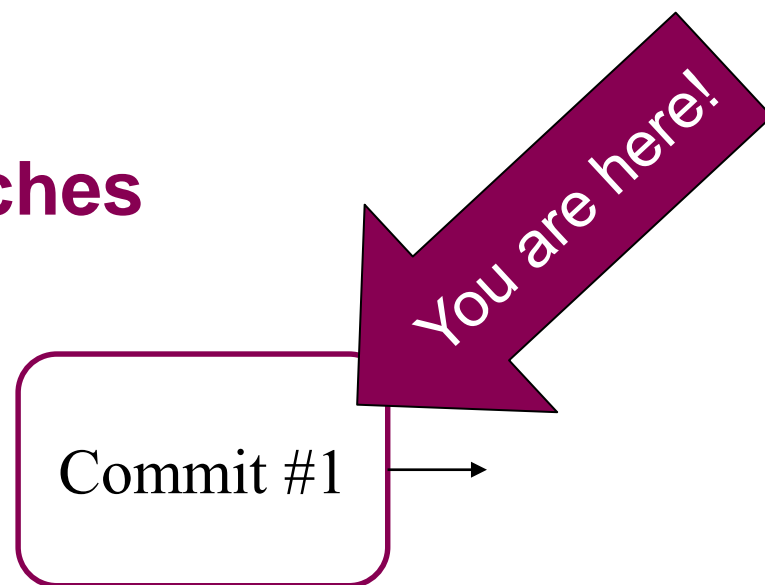
```
$ git push <remote> <branch>
```

The core of managing your code

GIT BRANCHES

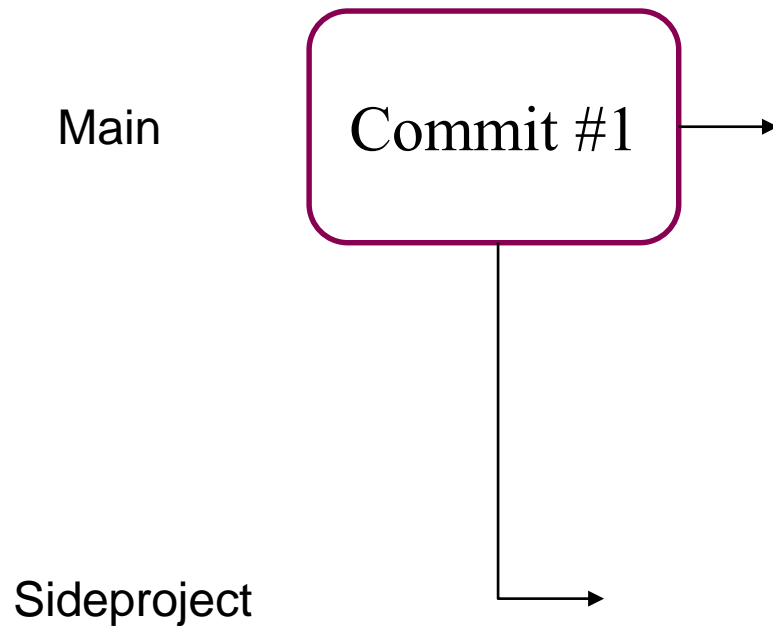
Branches

Main



```
$ git branch  
* Main
```

Branches

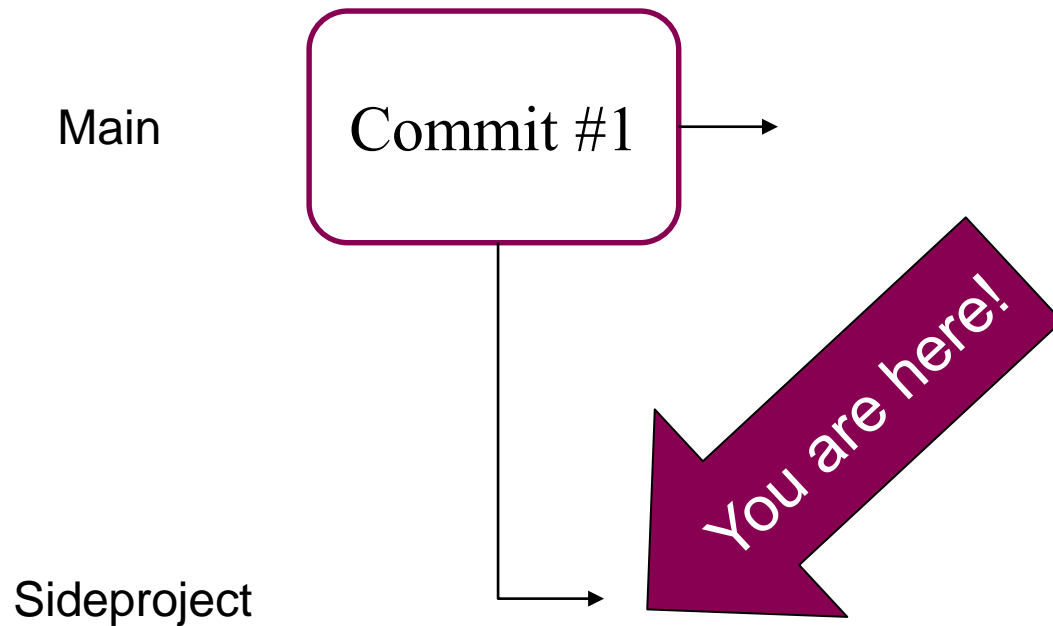


```
$ git branch  
* Main  
  
$ git checkout -b sideproject
```

Create new flag

Name of new branch

Branches

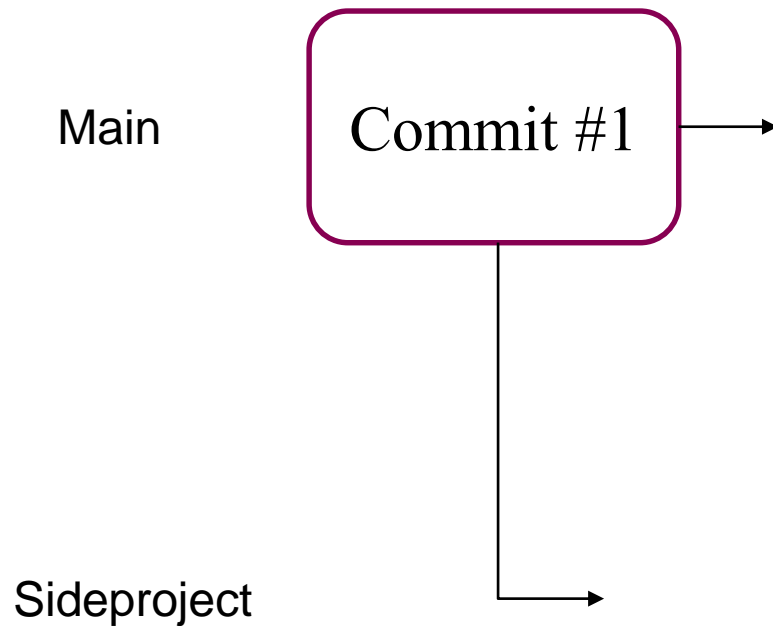


```
$ git branch
* Main

$ git checkout -b sideproject

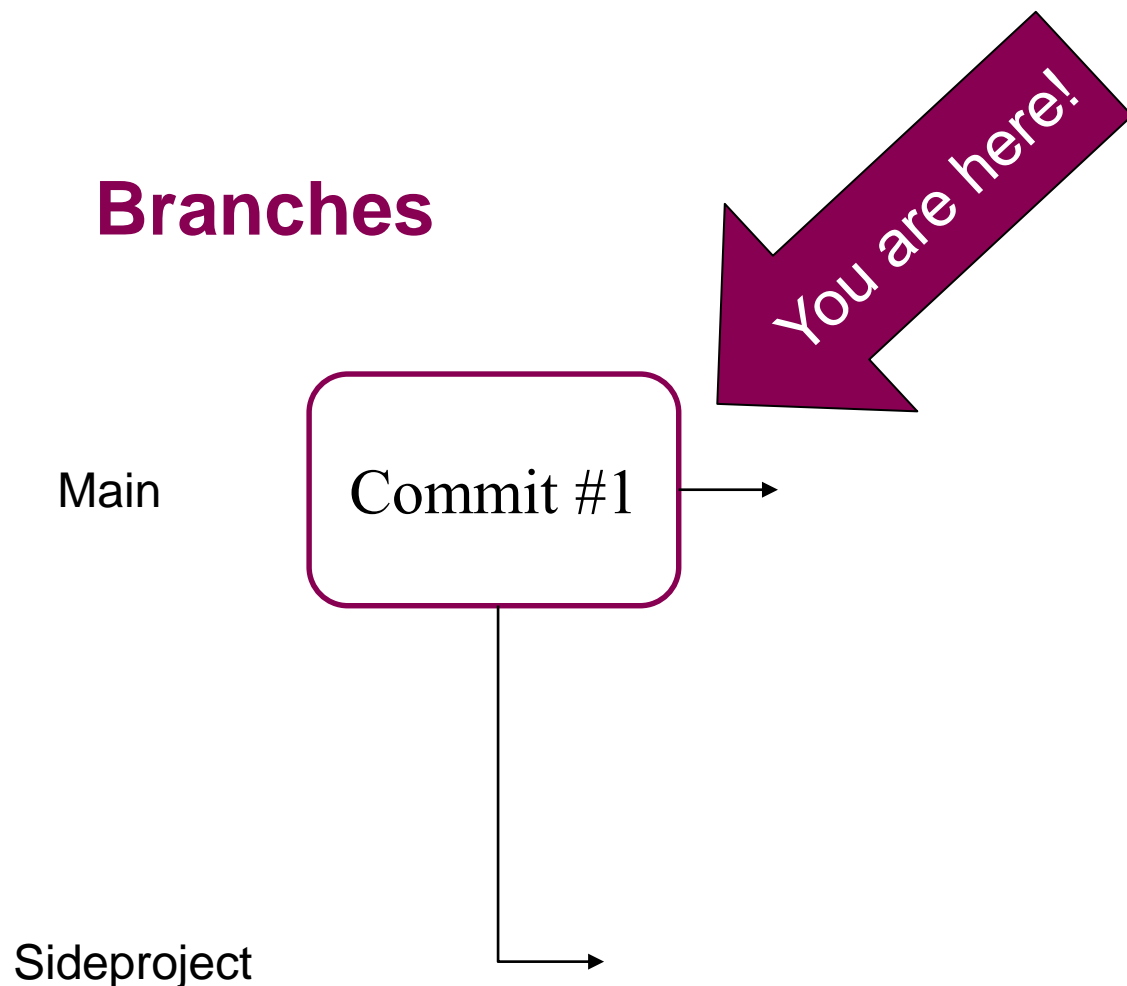
$ git branch
main
* sideproject
```

Branches



```
$ git commit -m "<...>"  
$ git push origin sideproject
```

Branches

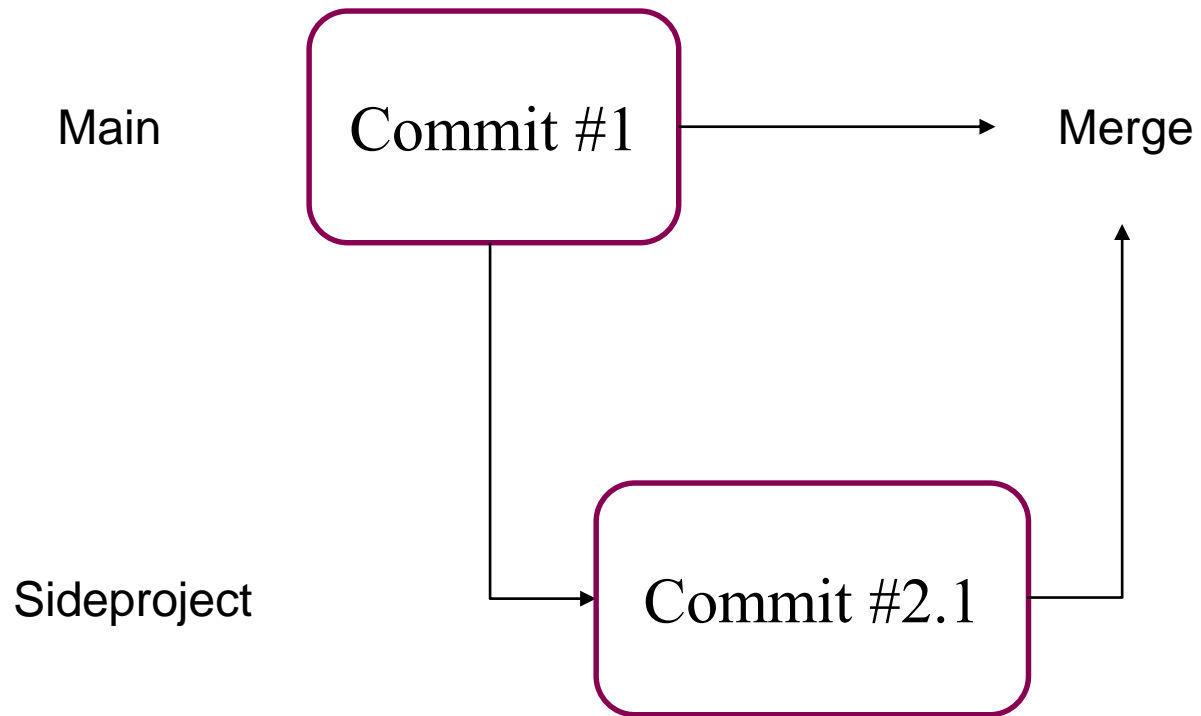


No "-b" flag



```
$ git checkout main  
$ git branch  
*  main  
   sideproject
```

Merge







```
$ git checkout main  
$ git merge sideproject
```


Alternative merge


```
$ git push origin sideproject
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'sideproject' on GitHub by visiting:
remote:   https://github.com/mcvinding/Github-crash-
course/pull/new/sideproject
remote:
To github.com:mcvinding/Github-crash-course.git
 * [new branch]      sideproject -> sideproject
```

Alternative merge

 sideproject had recent pushes less than a minute ago [Compare & pull request](#)

 main  3 branches  0 tags [Go to file](#) [Add file](#) [Code](#)

 mcvinding added new_file 9fb4a7c 33 minutes ago ⌚ 2 commits

 LICENSE Initial commit 6 hours ago

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



base: main ▾



compare: sideproject ▾

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



two new files

Write

Preview

H

B

I

≡

<>



≡


≡



Leave a comment

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

Create pull request ▾

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

More on Git commits

main

1 branch

0 tags

mcvinding Initial commit

LICENSE

Initial commit

README.md

Initial commit

README.md

Github-crash-course

Using Git/Github in Scientific Collaborations

added new_file

main

mcvinding committed 28 seconds ago

Showing 2 changed files with 3 additions and 0 deletions.

0

another_new_file.txt

Empty file.

3

new_file.txt

@@ -0,0 +1,3 @@

1 + Nothing to see here.

2 +

3 + This is just a normal text file.

When to use branches

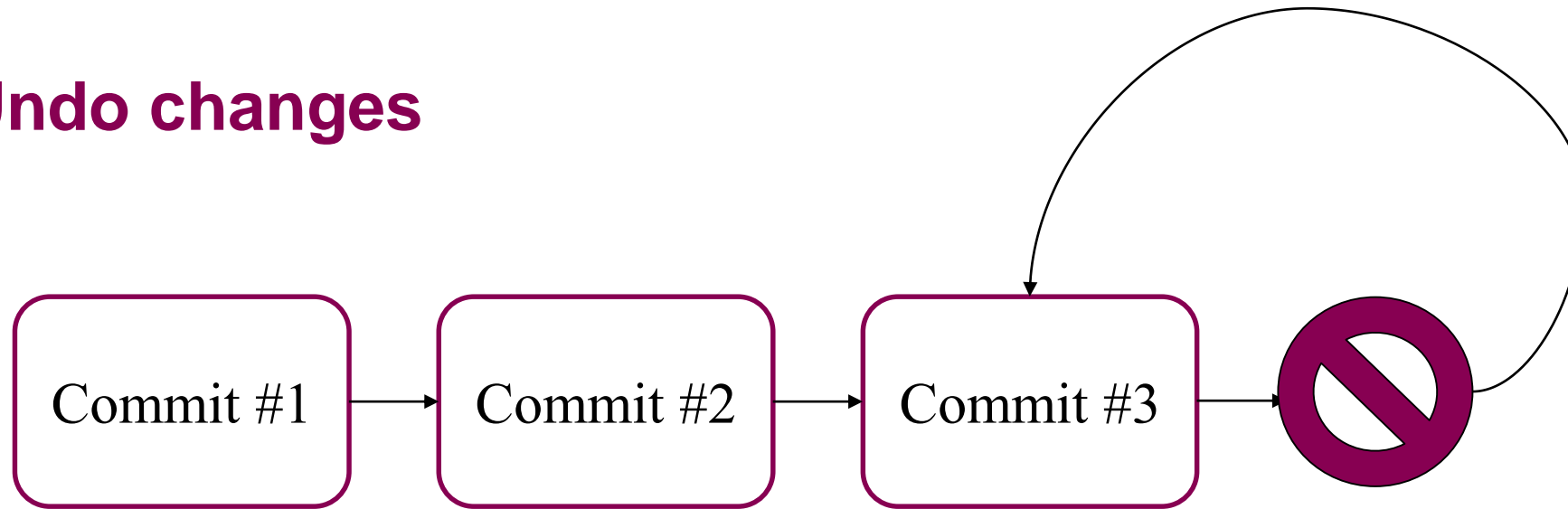
- When implementing entirely new features / analysis parts in existing code
- Testing out others code (and don't want to risk ruining it)

When not to use branches

- Small changes
 - To scrap current line (stay on main)
-

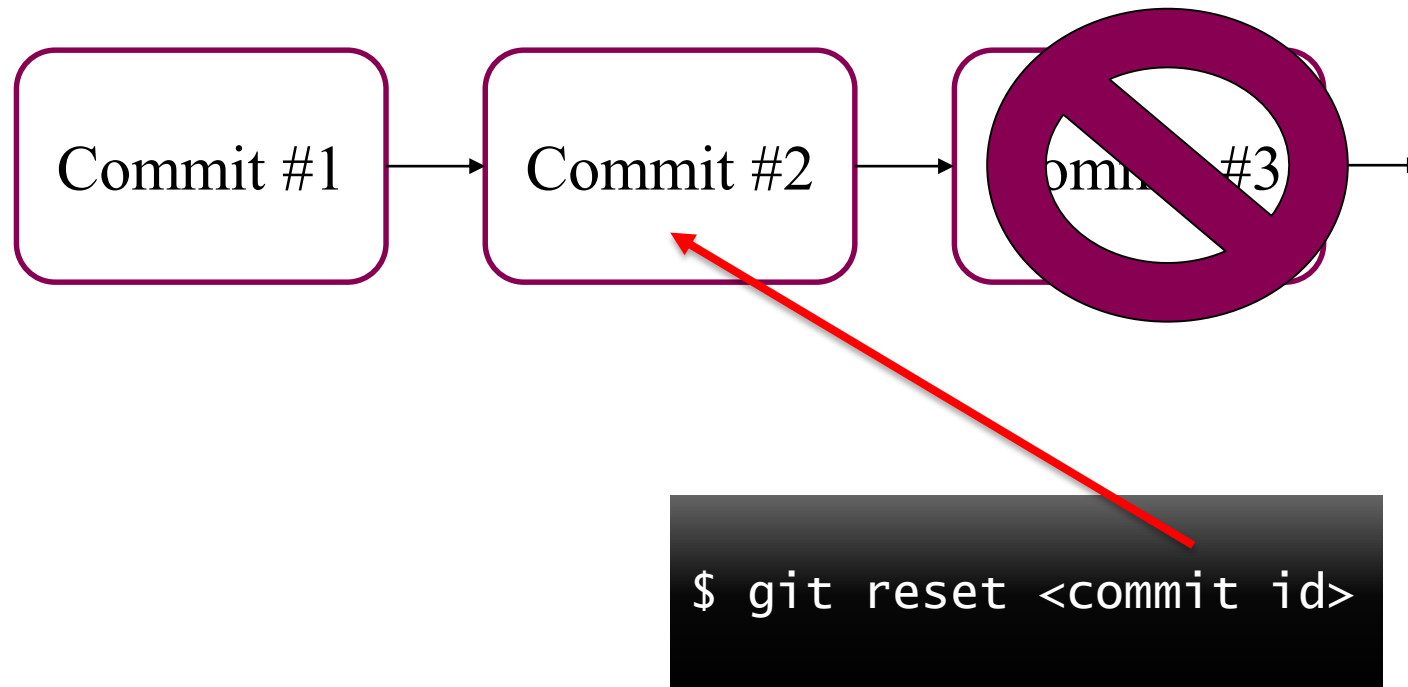


Undo changes



```
$ git reset --hard HEAD
```

Undo changes



Undo changes



```
$ git reset 4fa2ea669e898f2ba0549f50eada713ccd65cf89
```

```
$ git log
commit 9fb4a7cbe74a87abccb5fd3fb1d04e529f3417bb (HEAD ->
main, origin/main)
Author: mcvinding <mikke1.vinding@gmail.com>
Date:   Mon Jan 20 16:33:56 2021 +0100

    added new_file

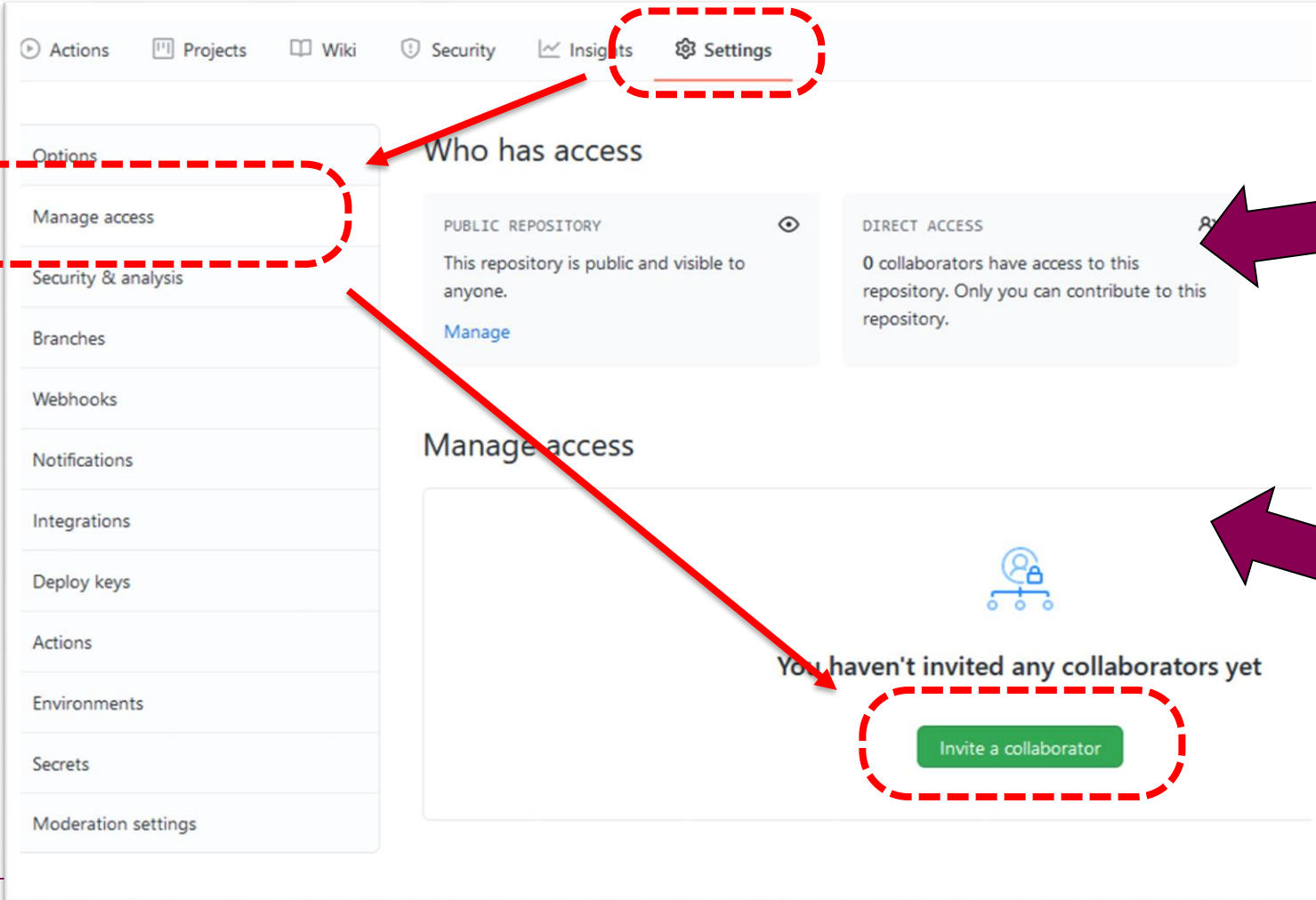
commit 4fa2ea669e898f2ba0549f50eada713ccd65cf89
(origin/master, main)
Author: MC Vinding <mikke1.vinding@gmail.com>
Date:   Mon Jan 20 16:26:53 2021 +0100

    initial commit
```

The more, the merrier: using GitHub with your lab mates

GITHUB COLLABORATIONS

Adding collaborators



The screenshot shows the GitHub repository settings page. The 'Settings' tab is selected in the top navigation bar. On the left sidebar, the 'Manage access' option is highlighted. The main content area is divided into two sections: 'Who has access' and 'Manage access'. The 'Who has access' section shows 'PUBLIC REPOSITORY' and 'DIRECT ACCESS' with '0 collaborators'. The 'Manage access' section shows a message 'You haven't invited any collaborators yet' and a green 'Invite a collaborator' button. Red dashed boxes highlight the 'Settings' tab, the 'Manage access' option, and the 'Invite a collaborator' button. Red arrows point from the 'Manage access' option to the 'Who has access' section and from the 'Manage access' section to the 'Invite a collaborator' button. A purple arrow points from the 'DIRECT ACCESS' section to the right, and another purple arrow points from the 'Manage access' section to the bottom right.

Actions Projects Wiki Security Insights **Settings**

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Environments

Secrets

Moderation settings

Who has access

PUBLIC REPOSITORY

This repository is public and visible to anyone.

[Manage](#)

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

[Invite a collaborator](#)

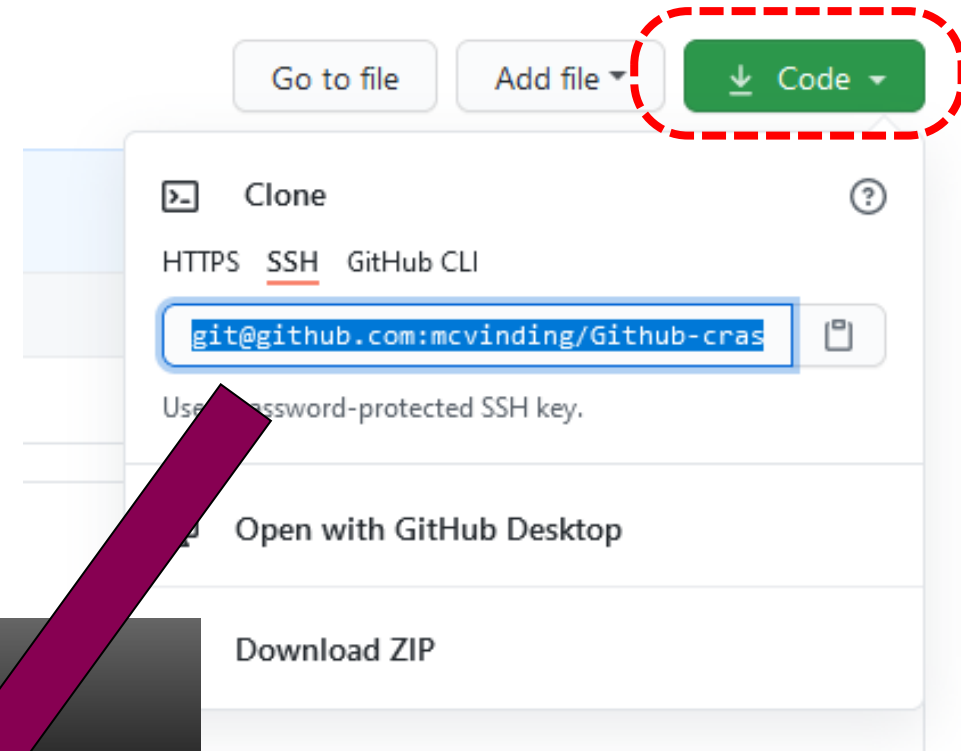


Setting up collaboration

- Initialize folder on your computer
- Go to your collaborators project
- Add remote



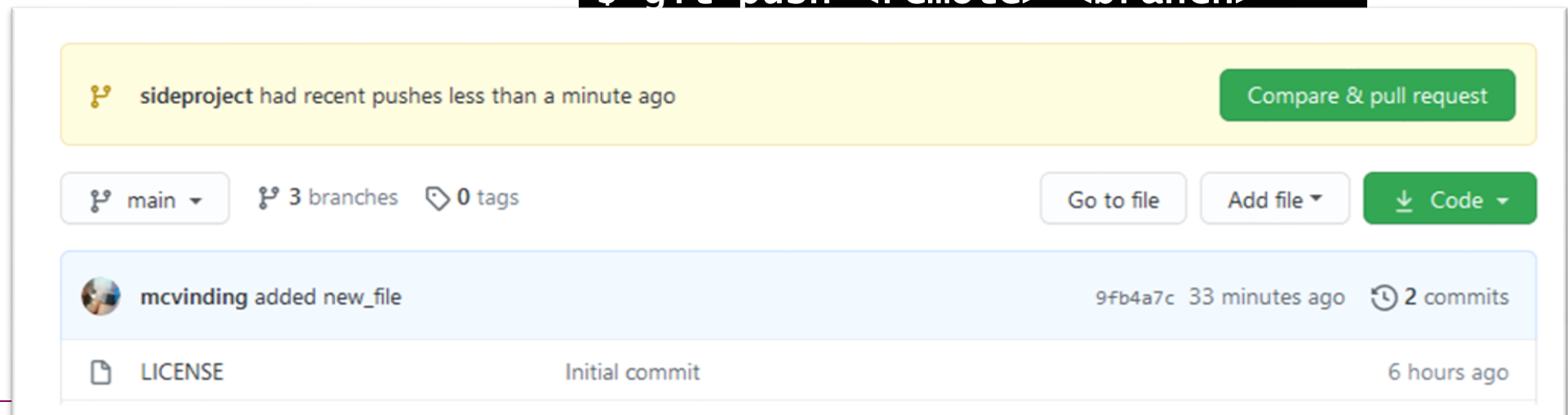
```
$ git init  
$ git remote add upstream <address>  
$ git pull origin main
```



Collaborating (same as before)

- **Pull** latest code
- Write your code...
- Stage edits
- Commit changes
- **Push**

```
$ git pull origin main  
  
$ git add <options>  
  
$ git commit  
  
$ git push <remote> <branch>
```



Summary

- Use Git/GitHub to manage your analysis scripts (not data).
- Basic usage (same if you are working on your own or in collaboration)

```
$ git remote add <remote> <branch>  
  
$ git pull origin main  
  
$ git add <options>  
  
$ git commit  
  
$ git push <remote> <branch>
```