Using Git/Github in Scientific Collaborations

Mikkel C. Vinding, PhD
Assistant Professor
NatMEG, CNS, Karolinska Insititutet
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.

```
$ git pull origin master
```

Code example

What is Git and Github?

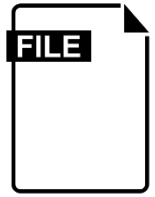
- Tool for distributed version control
- Free and open source
- https://git-scm.com/

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

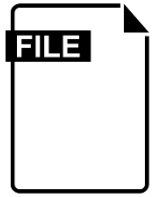
GitHub for scientific collaboration

1. Organize your data analysis scripts

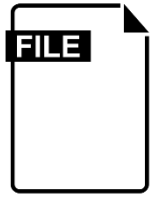
My Project



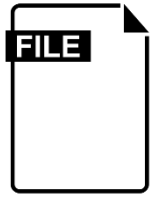
preproc



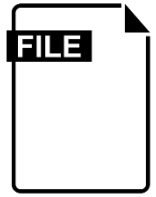
stats



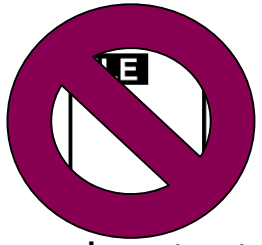
functions



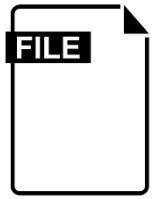
more_func



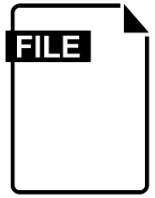
follow-up



misc_test



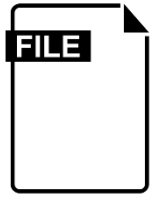
preproc2



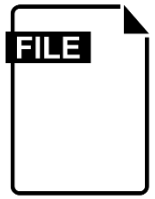
functions_new

GitHub for scientific collaboration

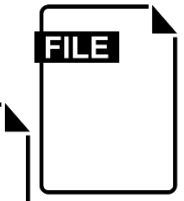
1. Organize your data analysis scripts
2. Easy to collaborate on tasks that requires programming (i.e. all neuroscientific data analysis)



stats

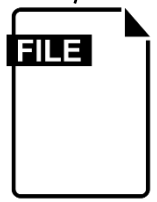


Stats_2

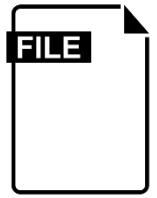


Stats_2b

Sorry, what version?



Stats_edits



Stats_edits

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

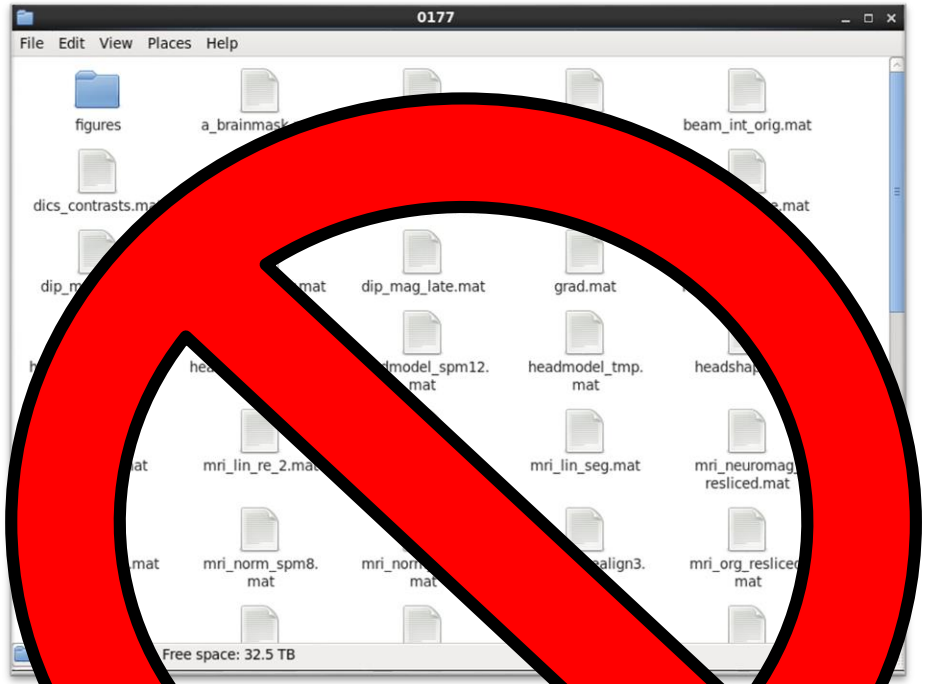
PD_proprioceptive_beta

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

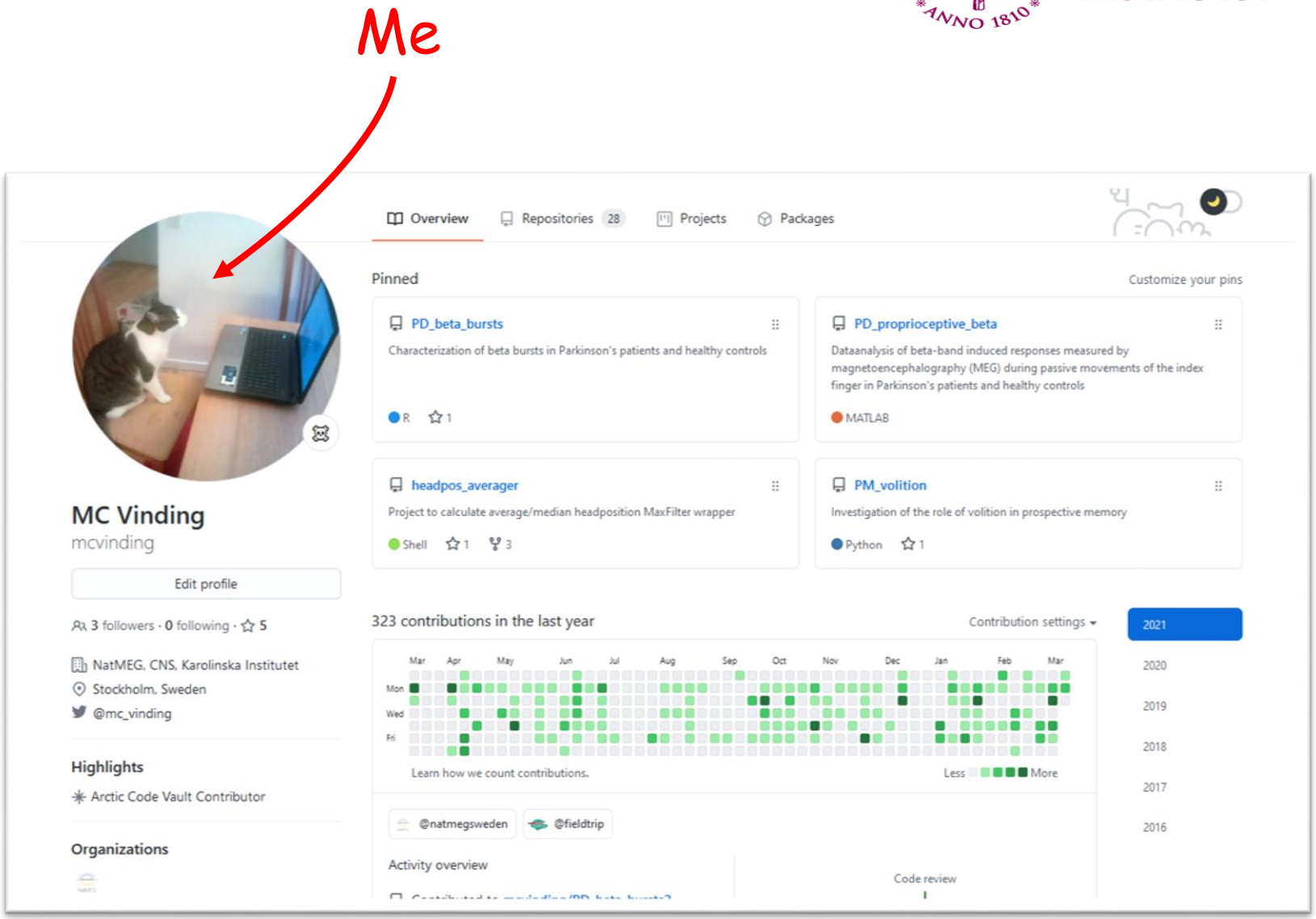


Subjid	Name	Social_secu	ress
0001	Mikkel Vinding		Nobels väg 9
0002	…		

GitHub accounts

Personal account

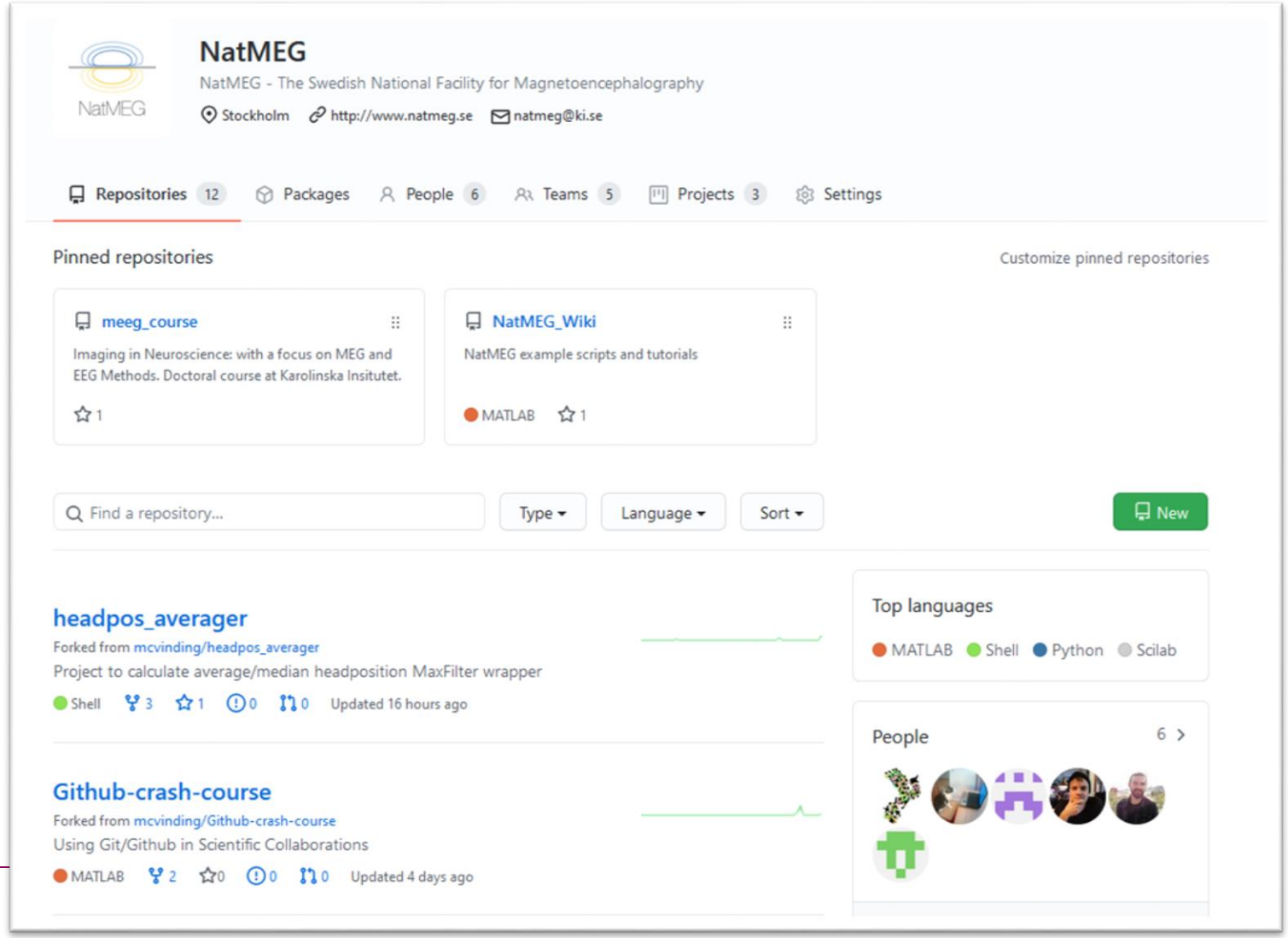
- ”your code”
  - → Research projects
  - → tools



GitHub accounts

Organizations

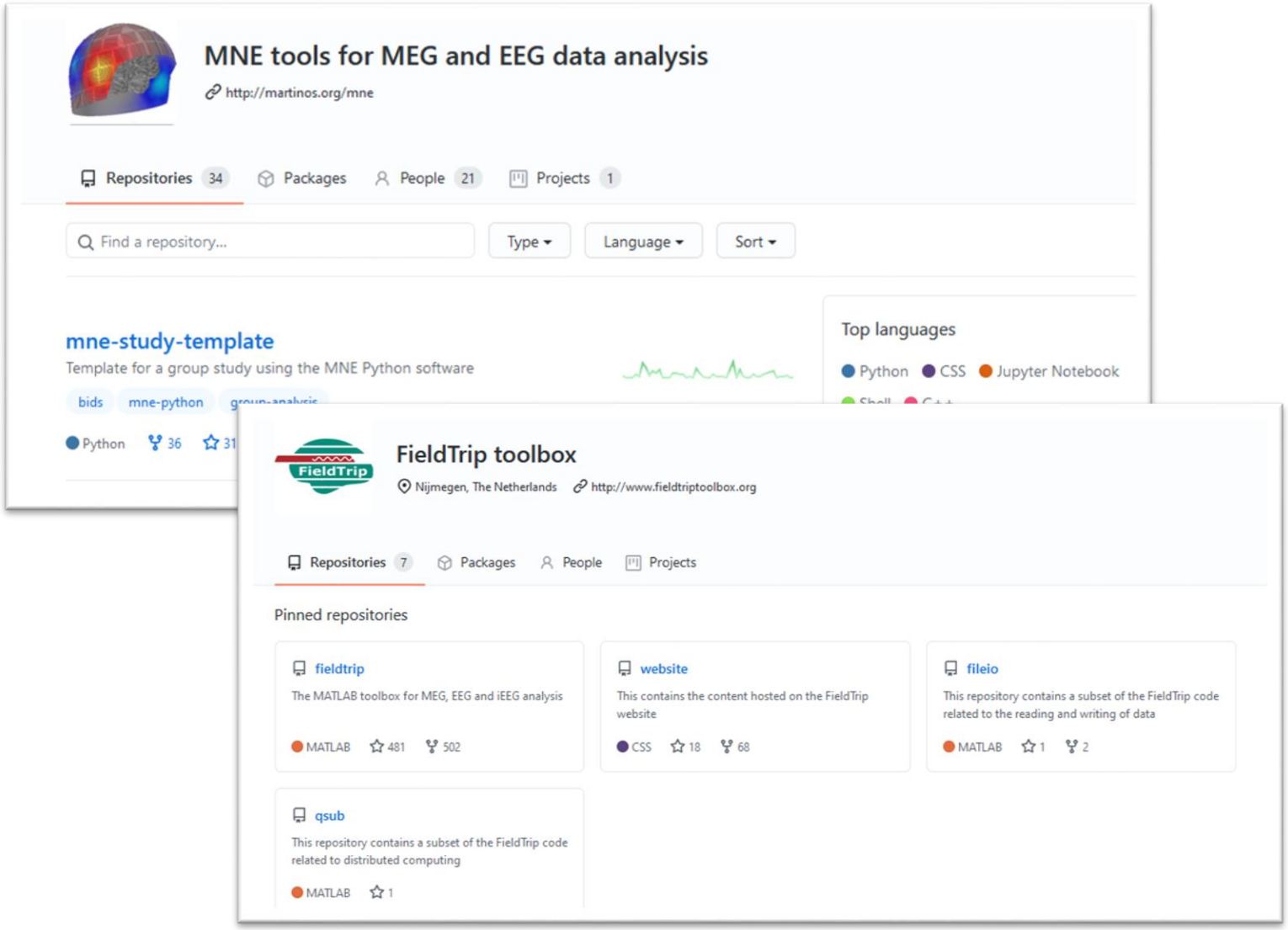
- Research group or lab
  - → Research projects
  - → ”Lab hacks”
  - → Tools



GitHub accounts

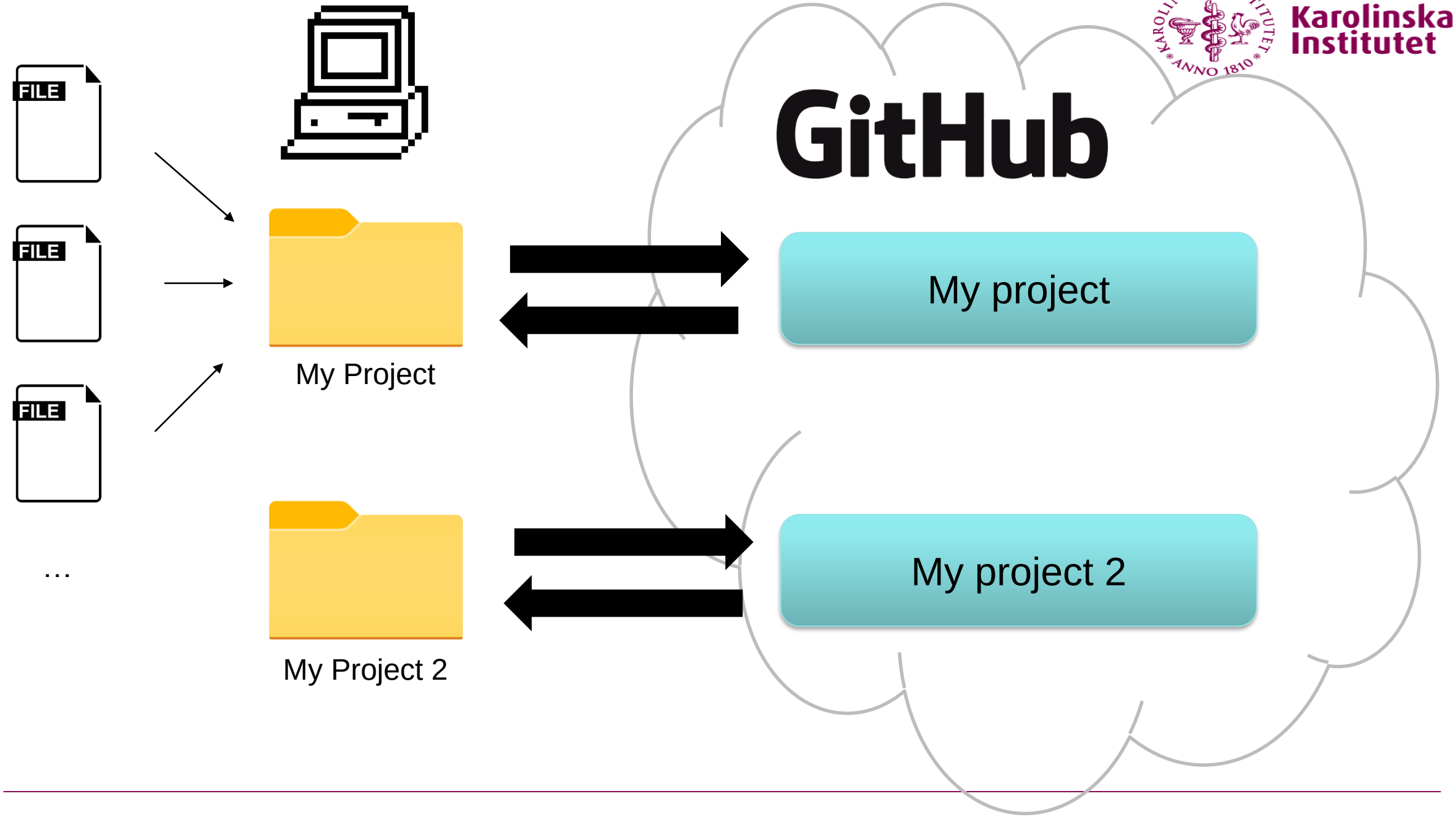
Organizations

- 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

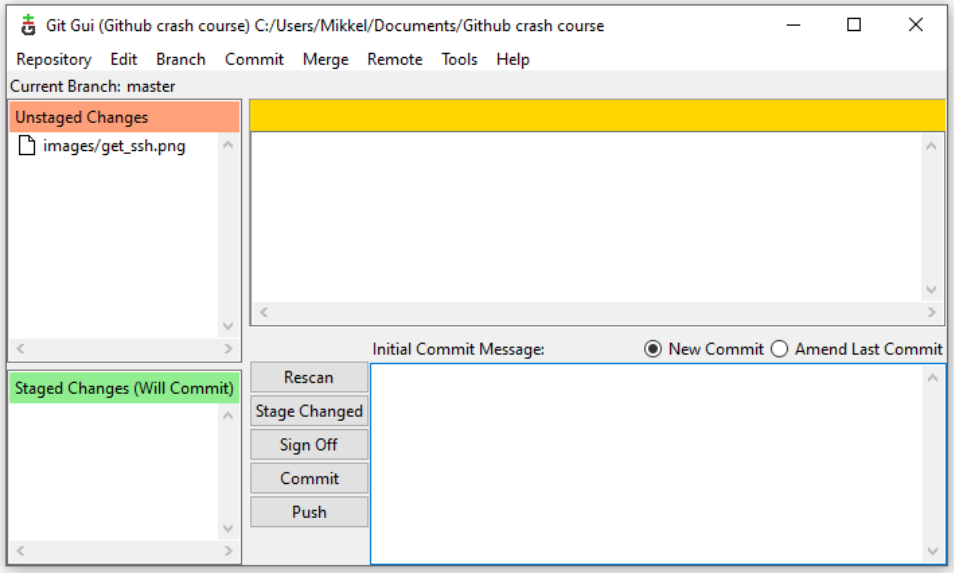

FILE
FILE
FILE
…
My Project
My Project 2
GitHub
My project
My project 2

Using Git: the basics

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

Do I have to use the terminal?

Yes*



*No: There is a GUI

Karolinska Institutet

Initiating a project

START A GIT PROJECT

Getting started…

Start a new project

Create a repository for your project.

Where you want to start!

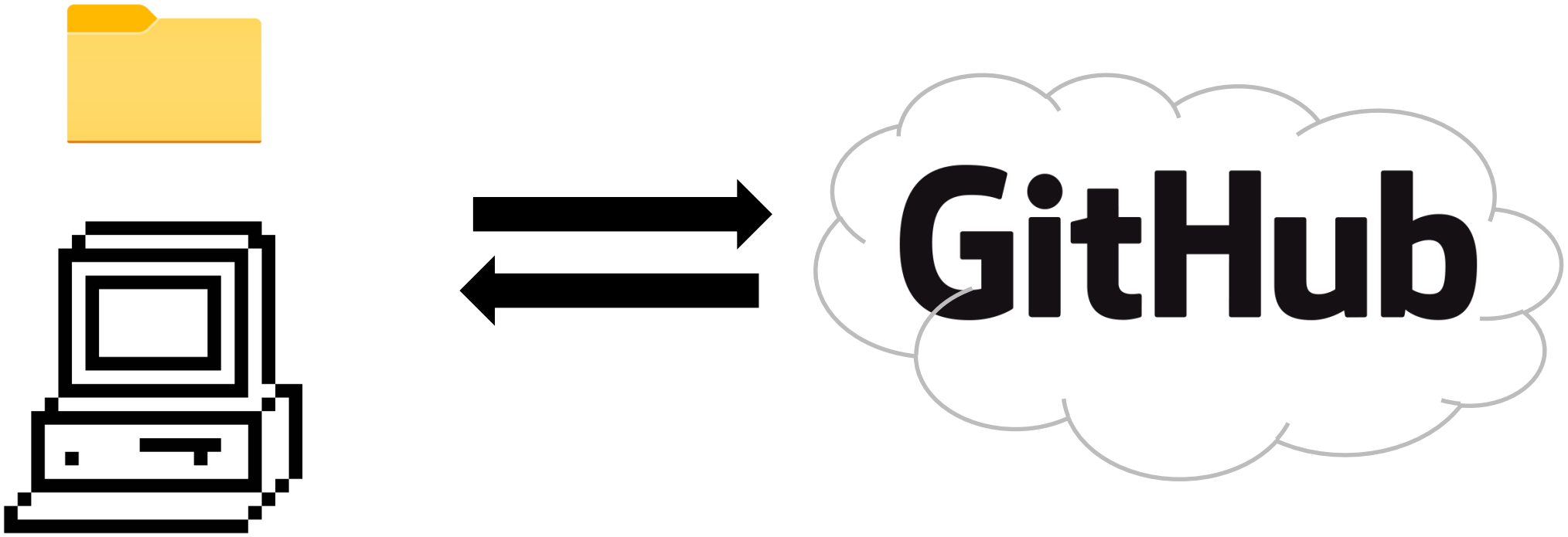
Join a project

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

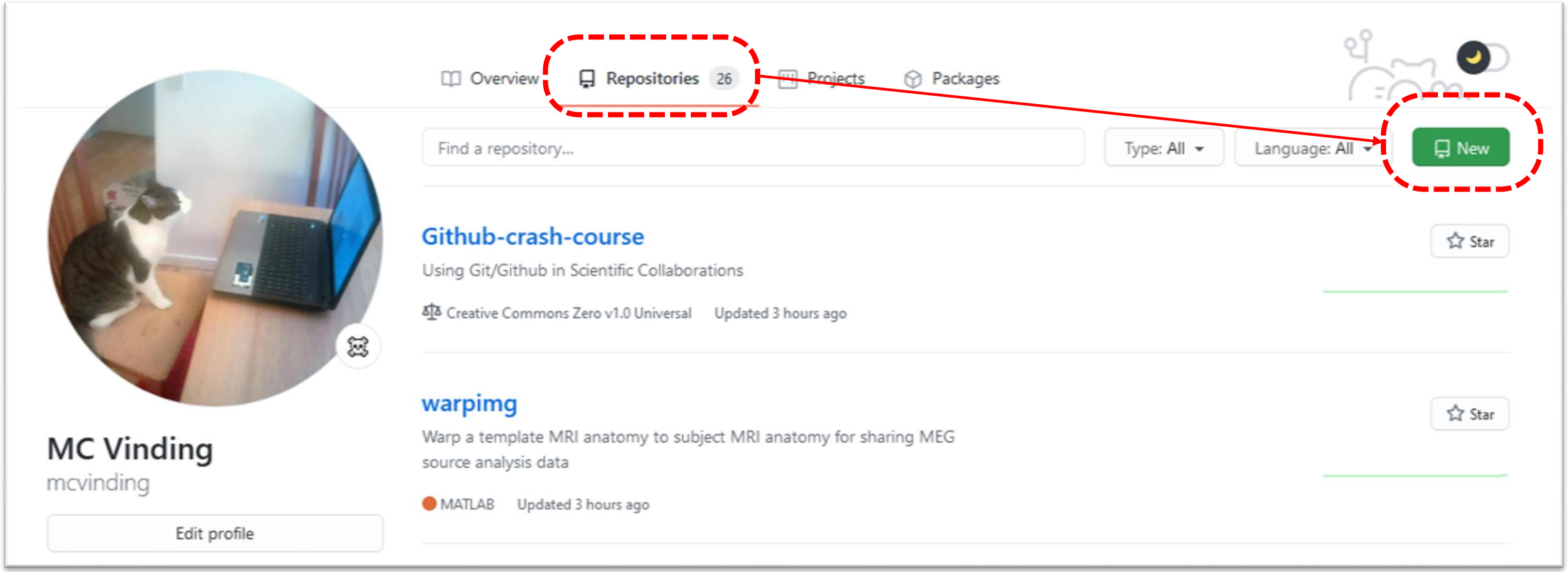
Clone a project

Get code from another project, or an analysis toolbox

Start a new project…



Start a new project…



1. Name

2. Define visibility

3. Add files

4. Create

Create a new repository

Owner * Repository name *

mcvinding /

Great repository names are short and memorable. Need inspiration? How about glowing-octo-pancake?

Description (optional)

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.

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

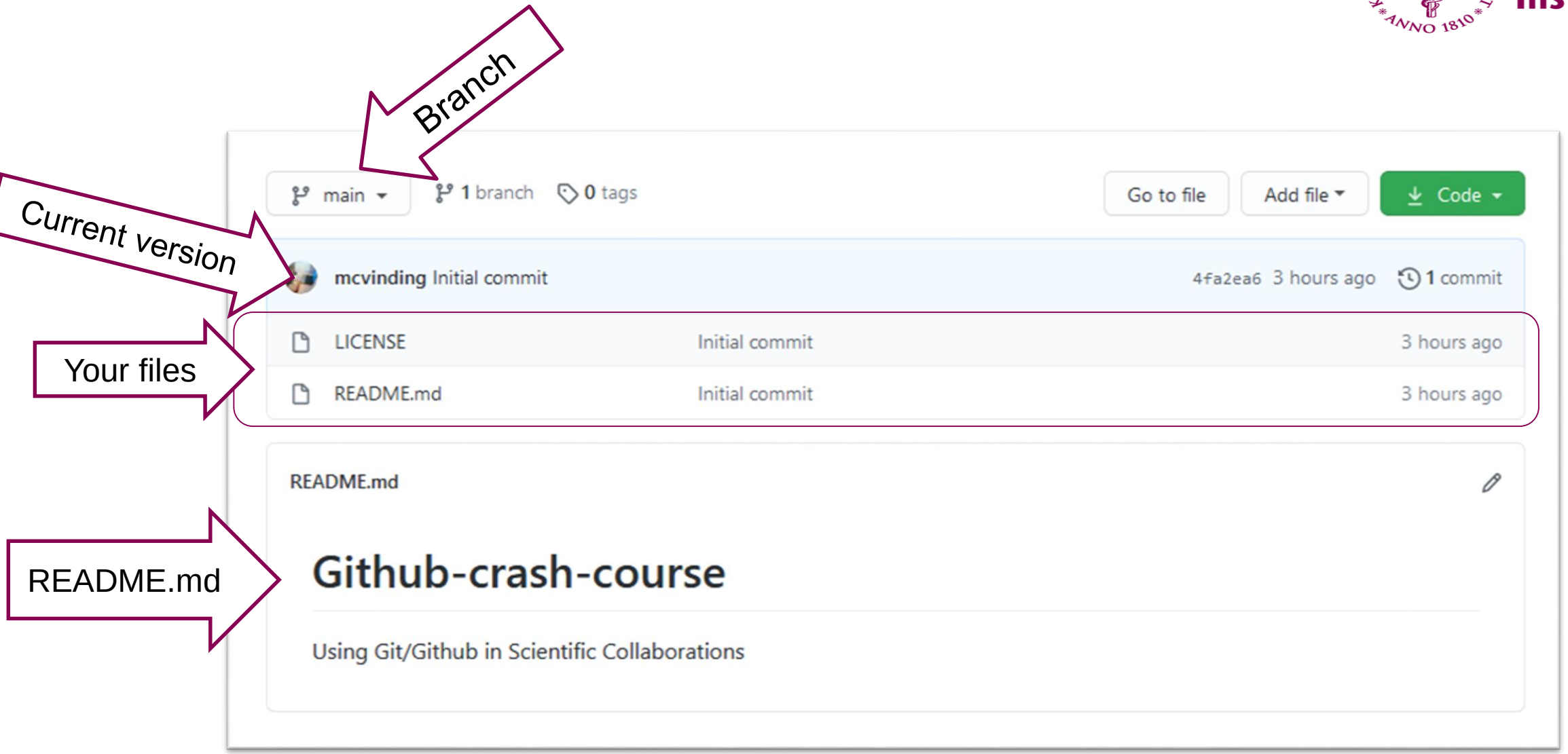
Add a README file
This is where you can write a long description for your project. Learn more.

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.

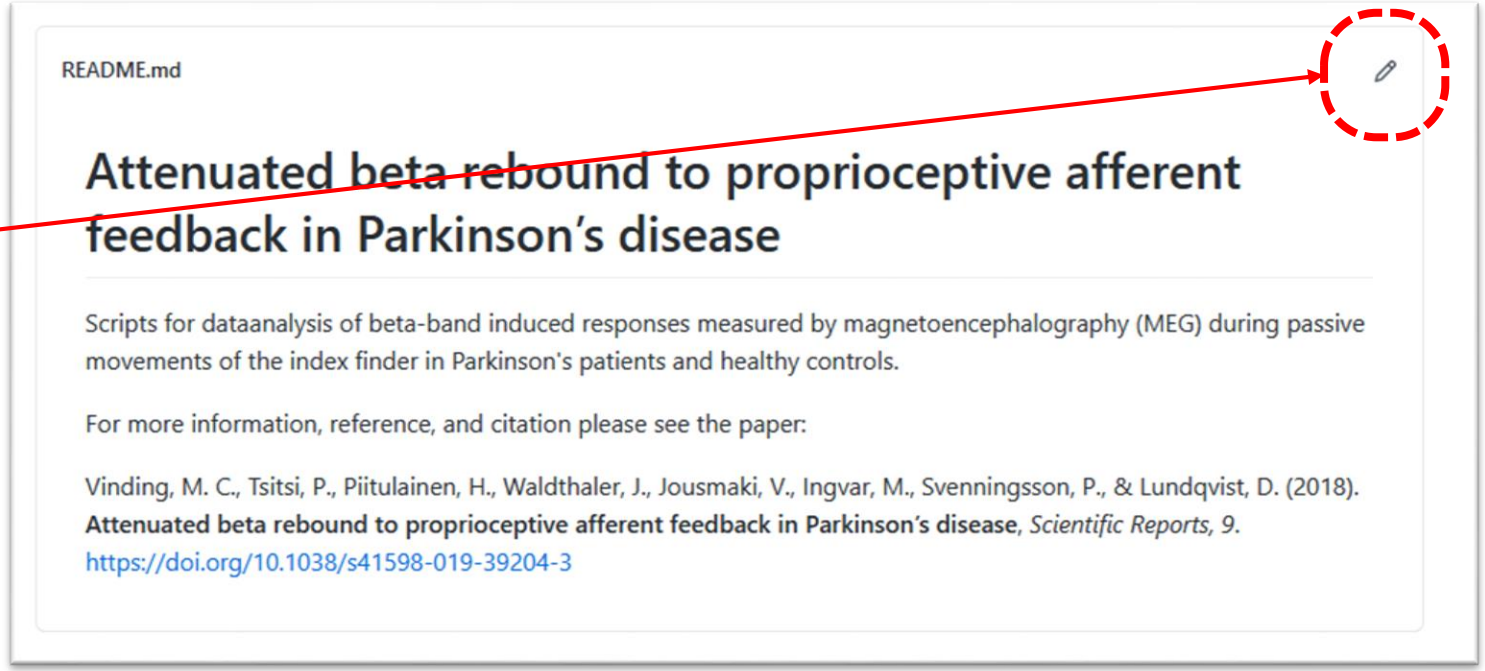
Create repository

Karolinska Institutet



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

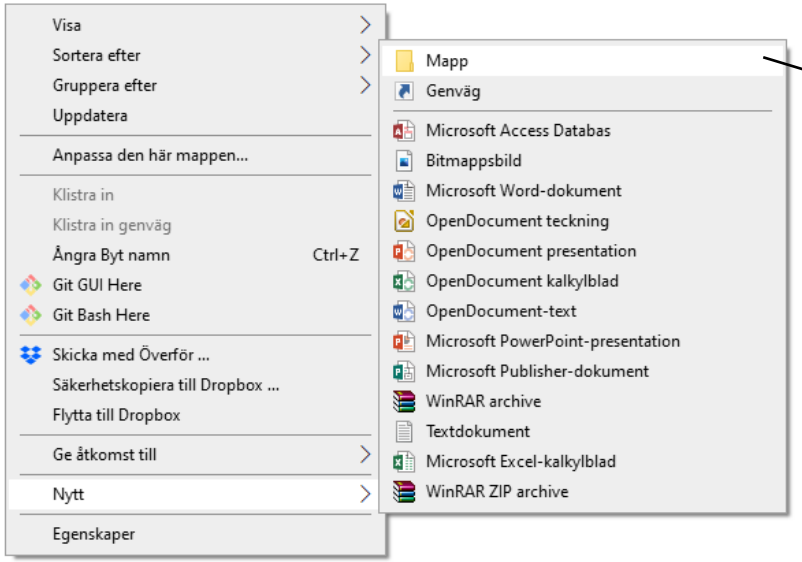


Nice Markdown cheatsheet:
https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

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…

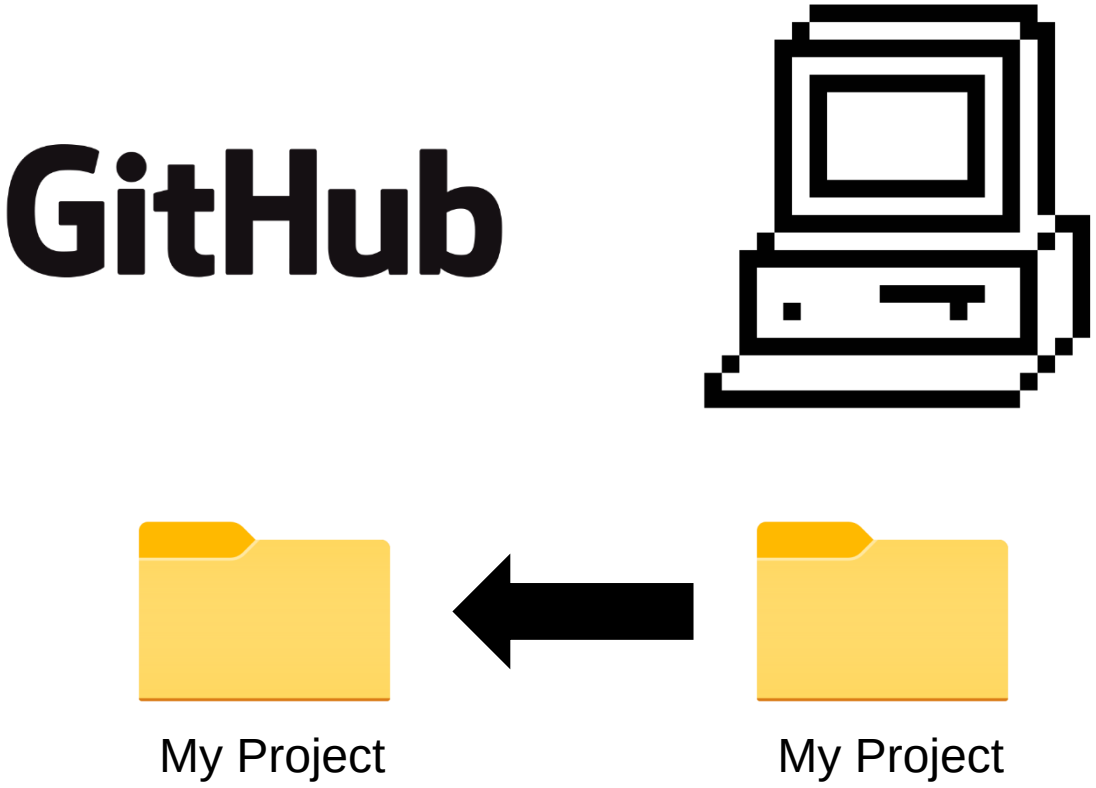
1. Create the local folder at the desired location
2. Go to folder
3. **Initialize** folder

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

My Project

Start a new project…

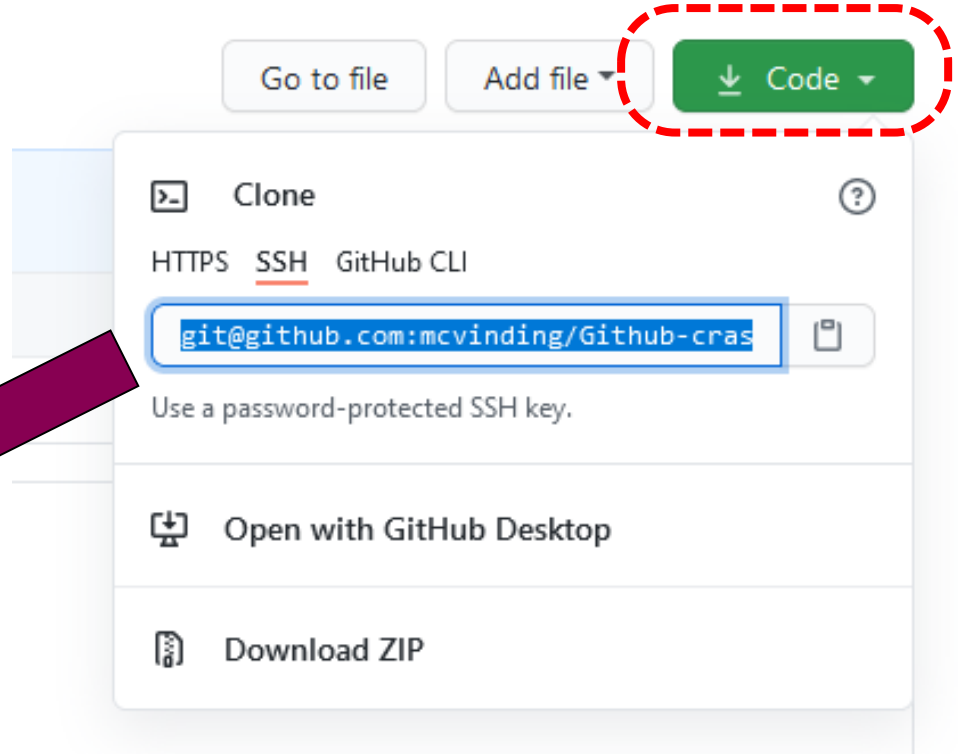
1. Create the local folder at the desired location
2. Go to folder
3. **Initialize** folder
4. Set **remote**



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>
```

My Project
“origin”

My Project

See remote address

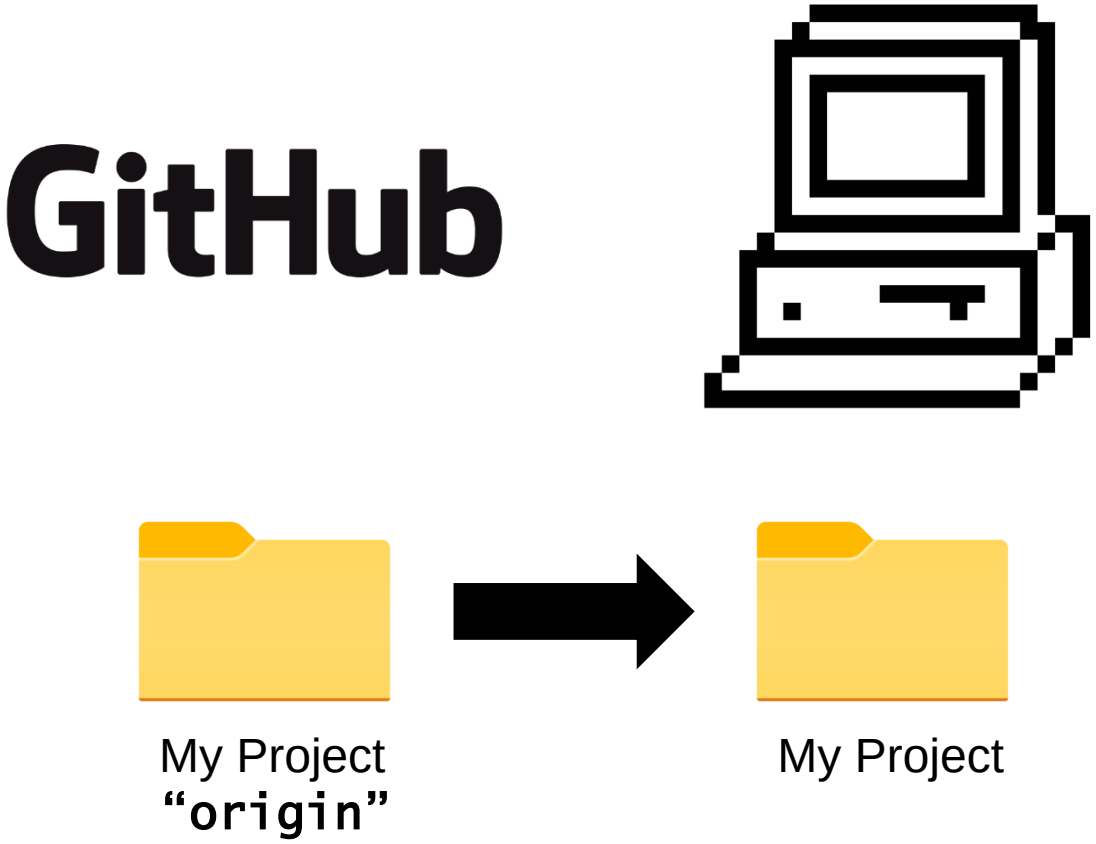
```
$ git remote -v
origin  git@github.com:mcvinding/Github-crash-course.git (fetch)
origin  git@github.com:mcvinding/Github-crash-course.git (push)
```

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"

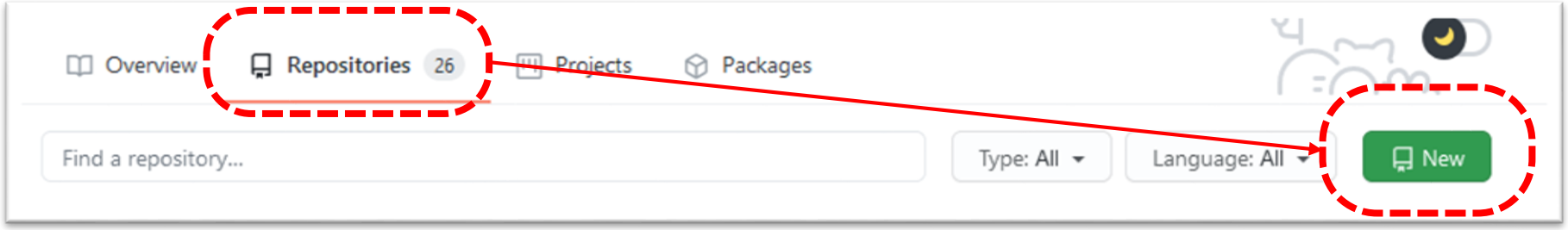


Get files

```
$ 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

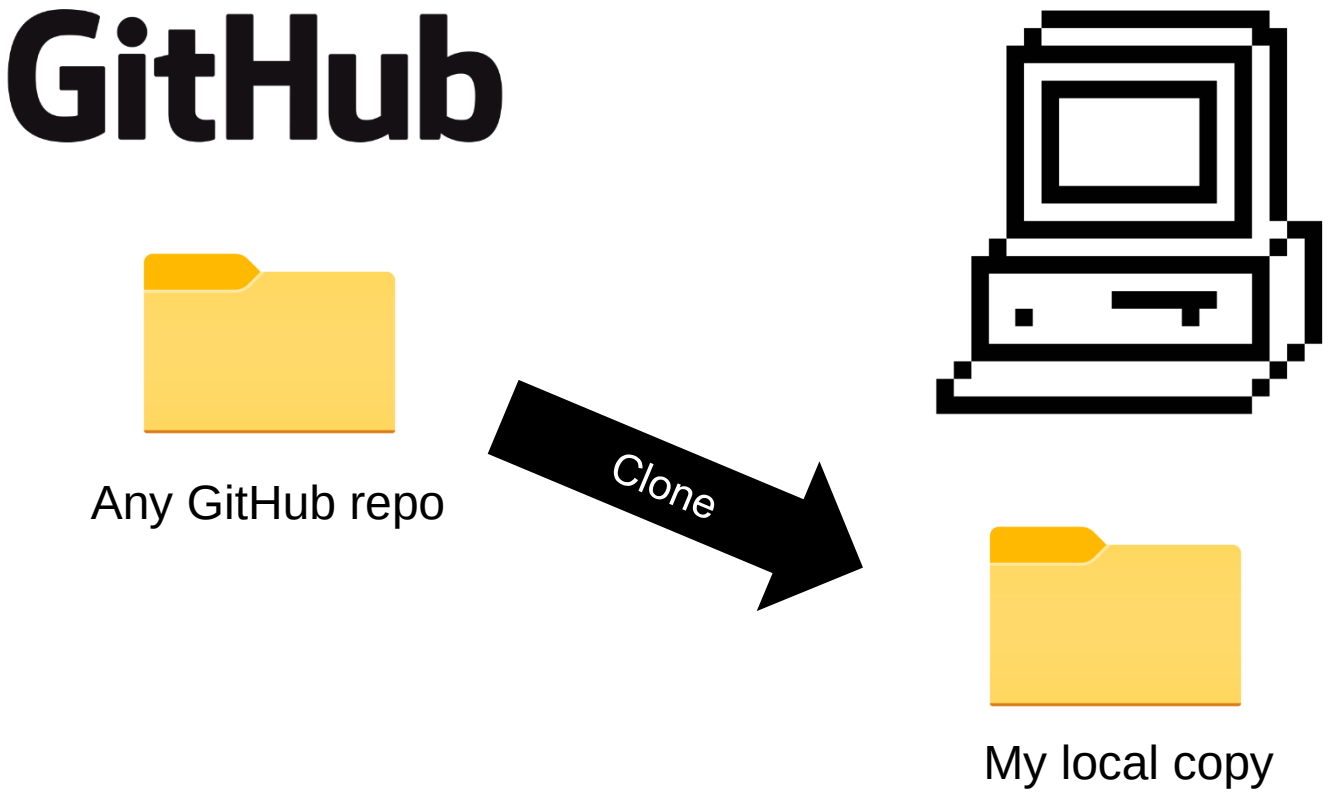
GitHub



```
$ git init

$ git remote add origin <address>

$ git pull origin main
```

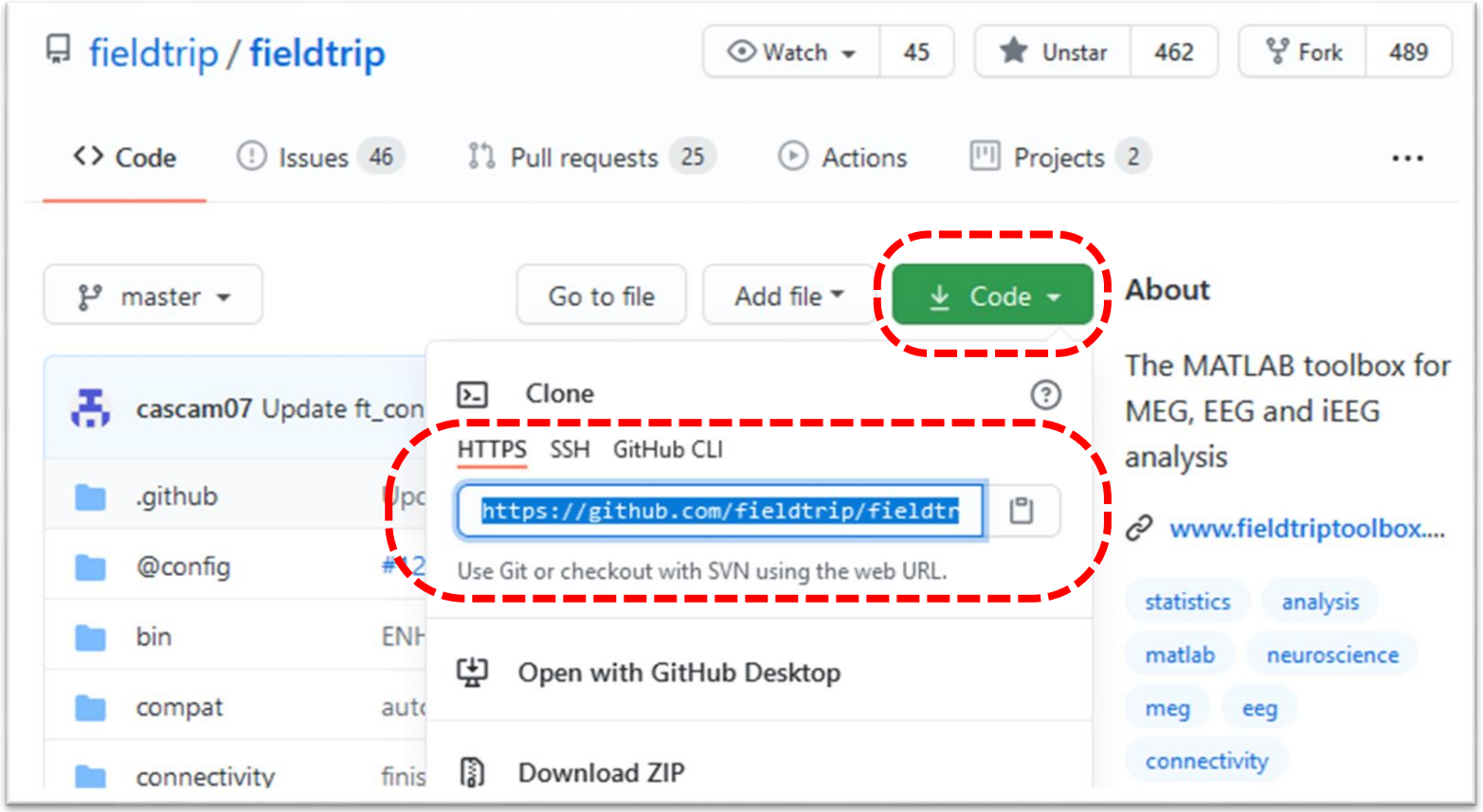
Git clone

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

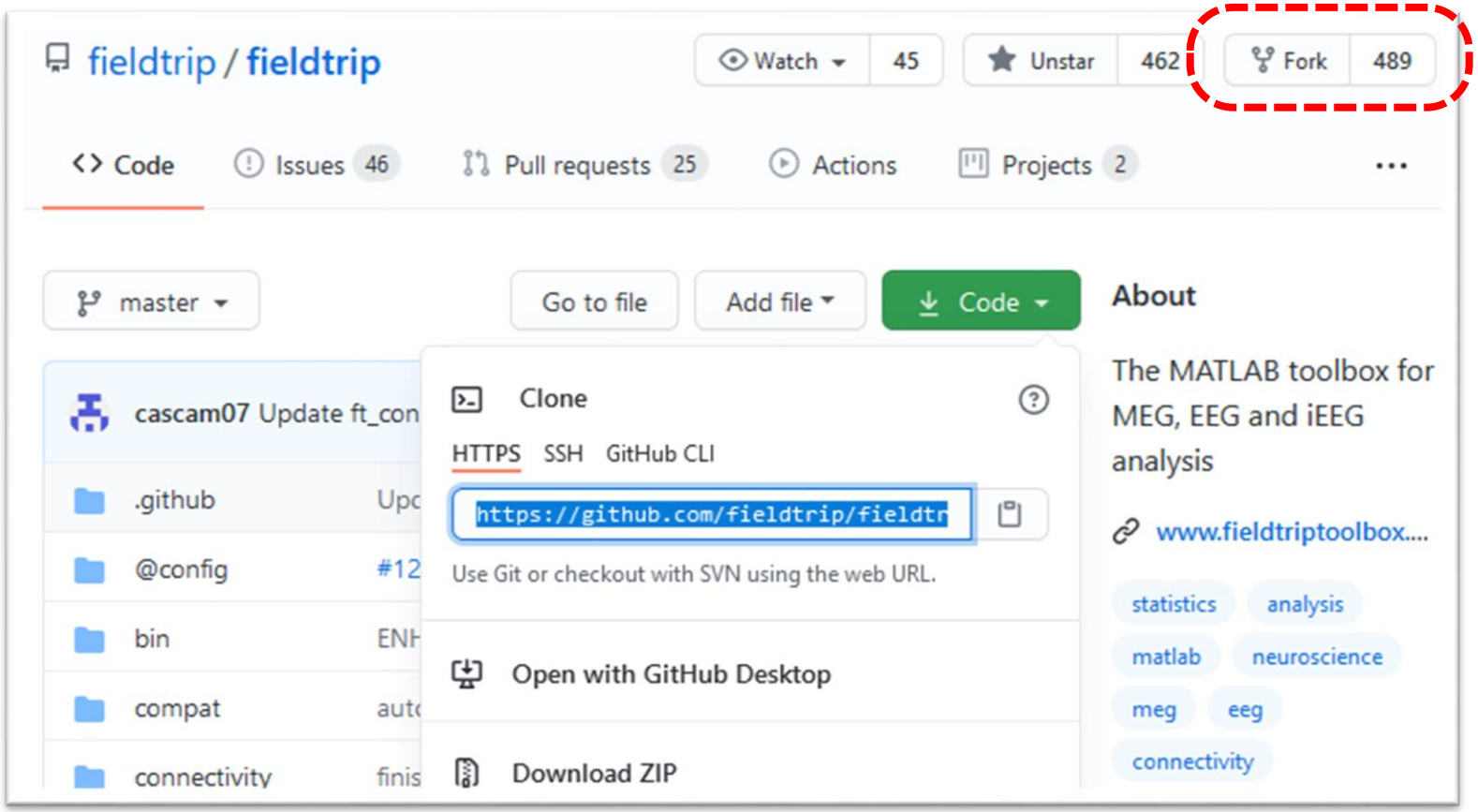


Git clone

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



Fork repository

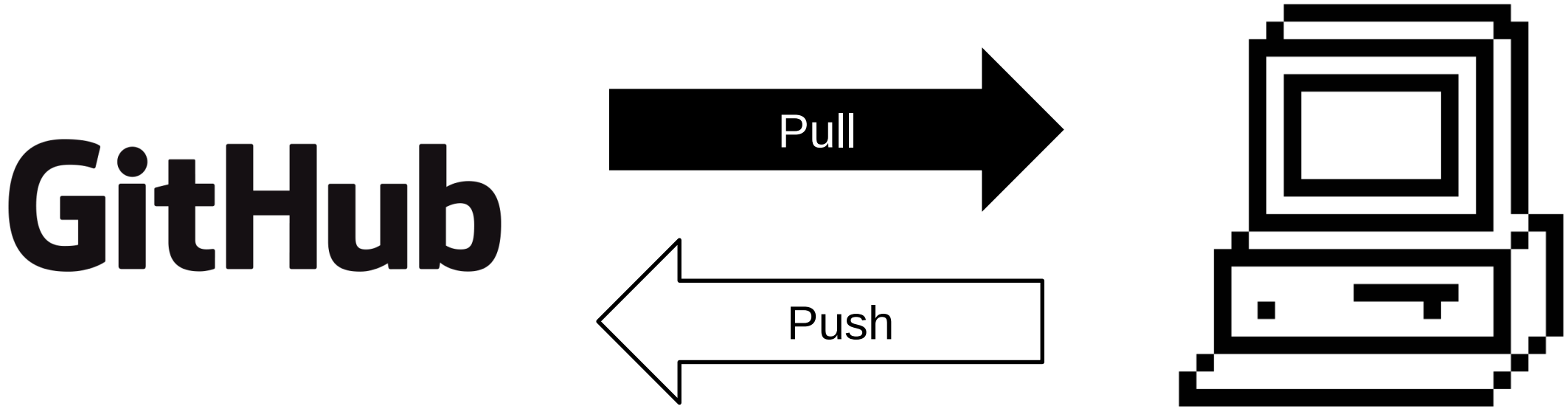


Karolinska Institutet

Pull, commit, push, and win at code management

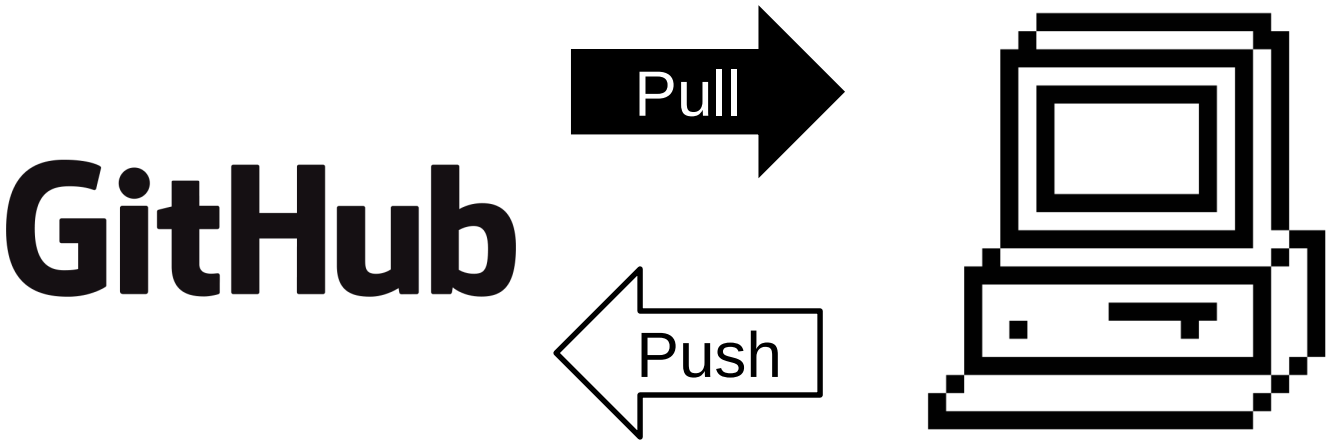
WORKING WITH GIT REPOSITORIES

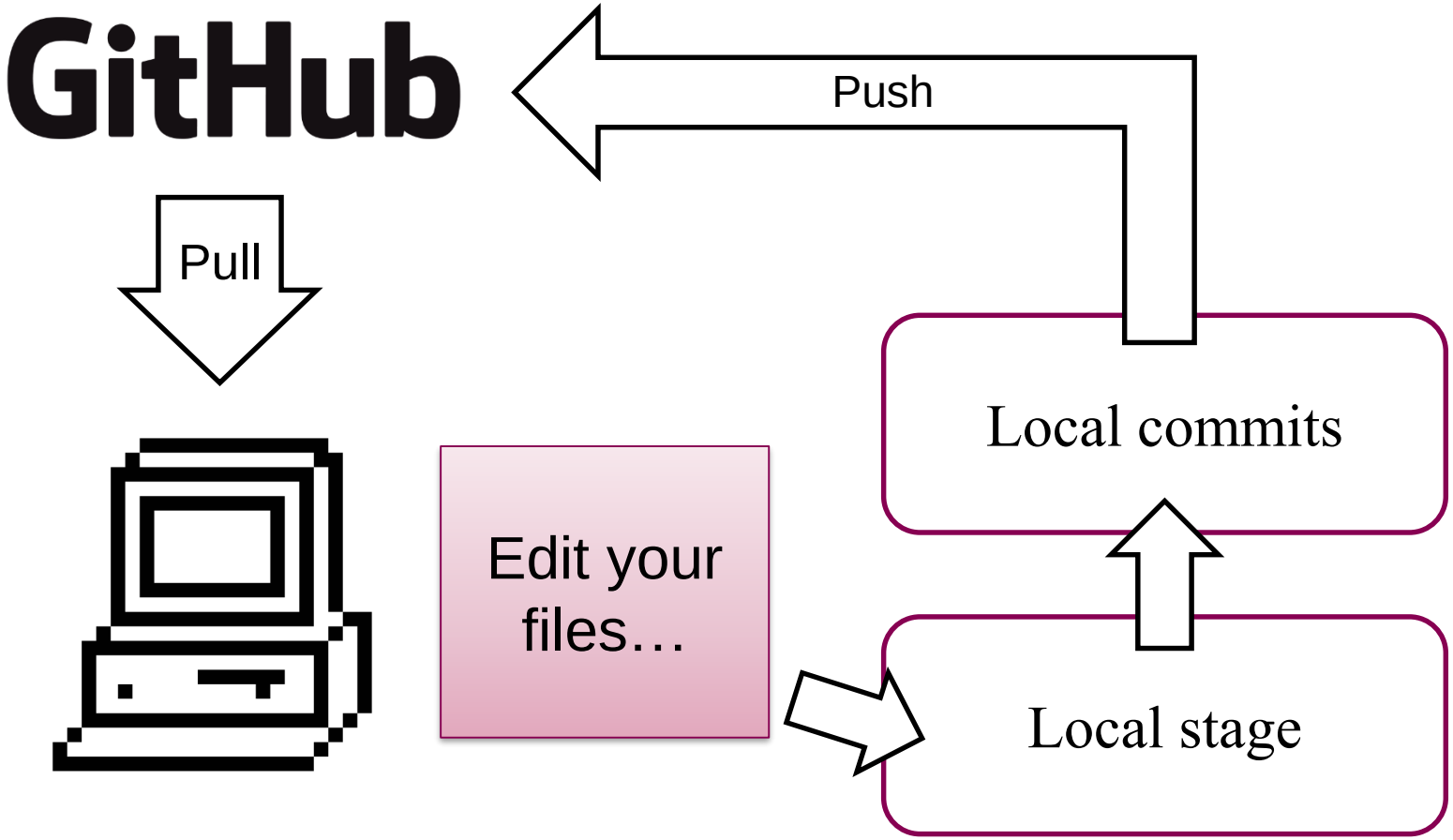
Git terminology



Git workflow at a glance

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

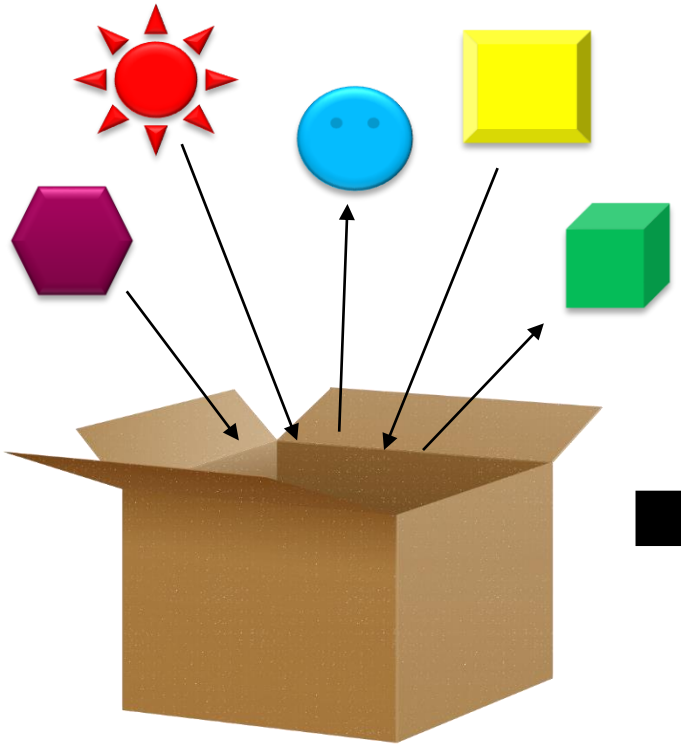





```
$ git pull origin main

$ git add <options>

$ git commit

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

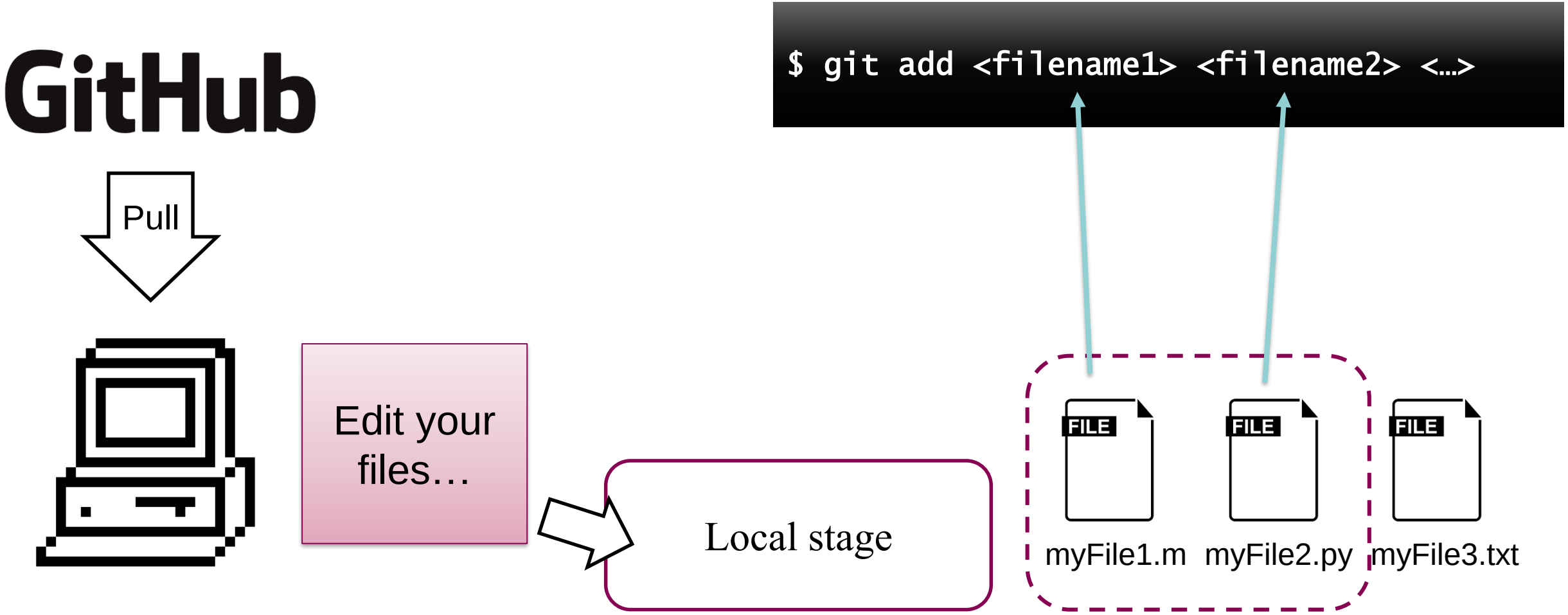
Karolinska Institutet

Stage



Commit

Push

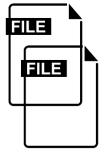

GitHub
Pull
Edit your files…
Local stage
$ git add <filename1> <filename2> <…>
FILE
FILE
FILE
myFile1.m
myFile2.py
myFile3.txt

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



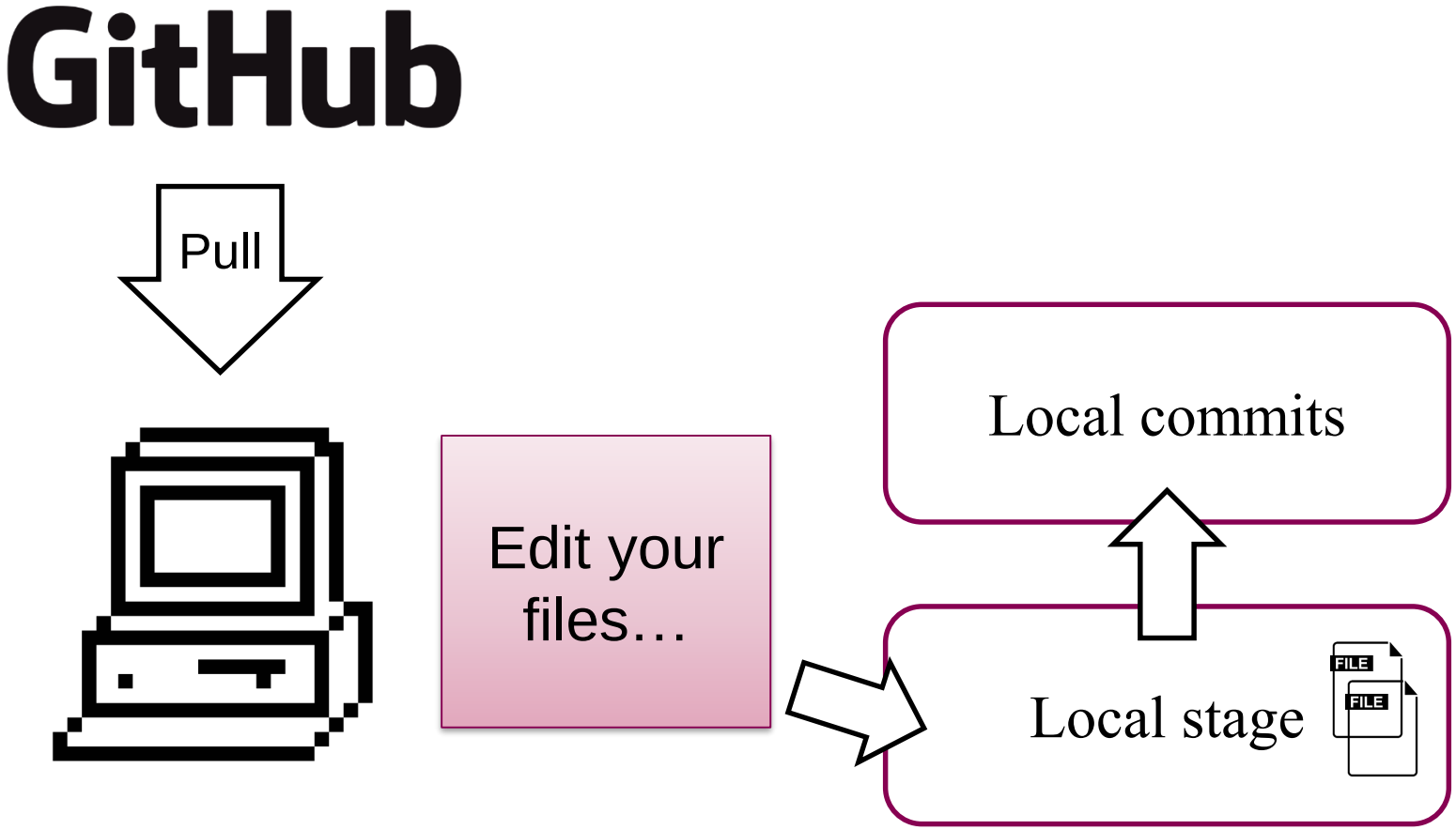
”Unstaged”

”Untracked”

Specific files

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

$ git add -u
```

All tracked but unstaged files

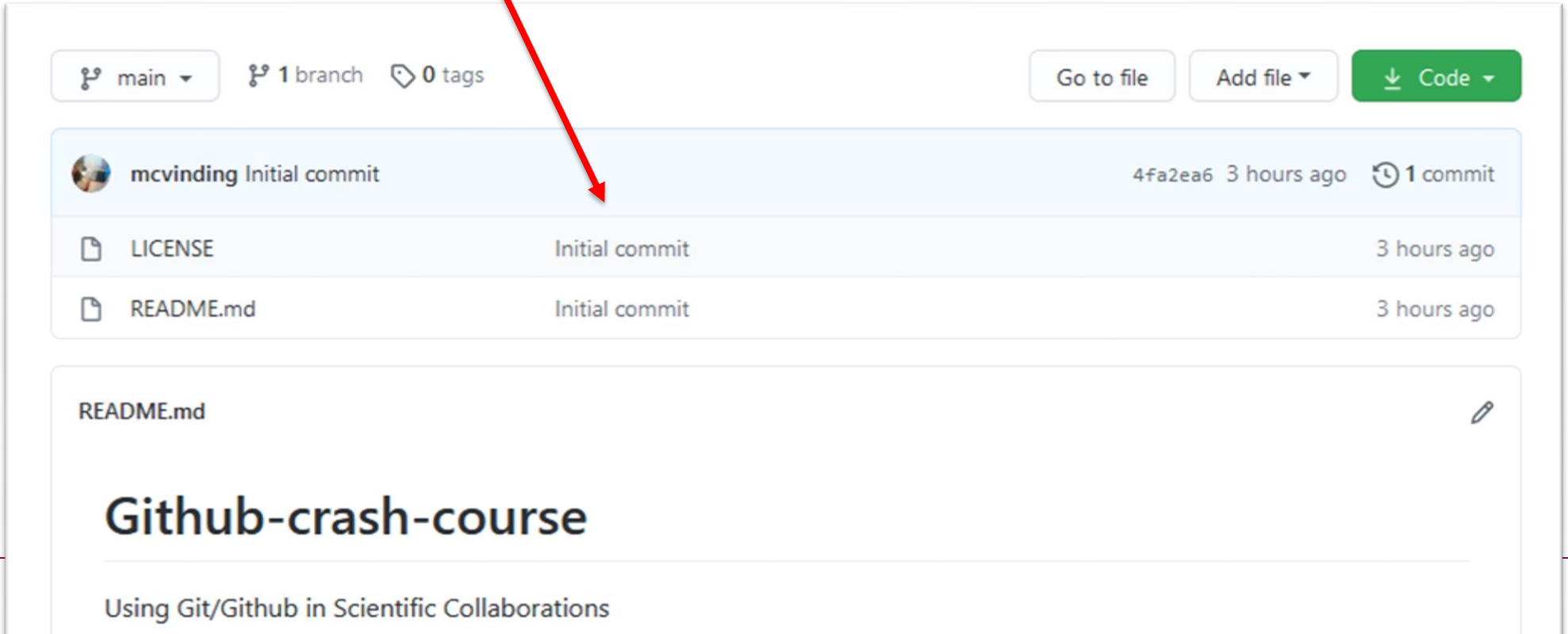

GitHub
Pull
Edit your files…
Local commits
Local stage
FILE
FILE

```
$ git commit -m ”<…>”
```

Commit messages

```
$ git commit –m ”<your message>”
```

Write a short informative text about changes



Commit messages

```
$ git commit –m ”<your message>”
```

Write a short informative text about changes

Too little

”Some changes”

”Committed stuff”

OK

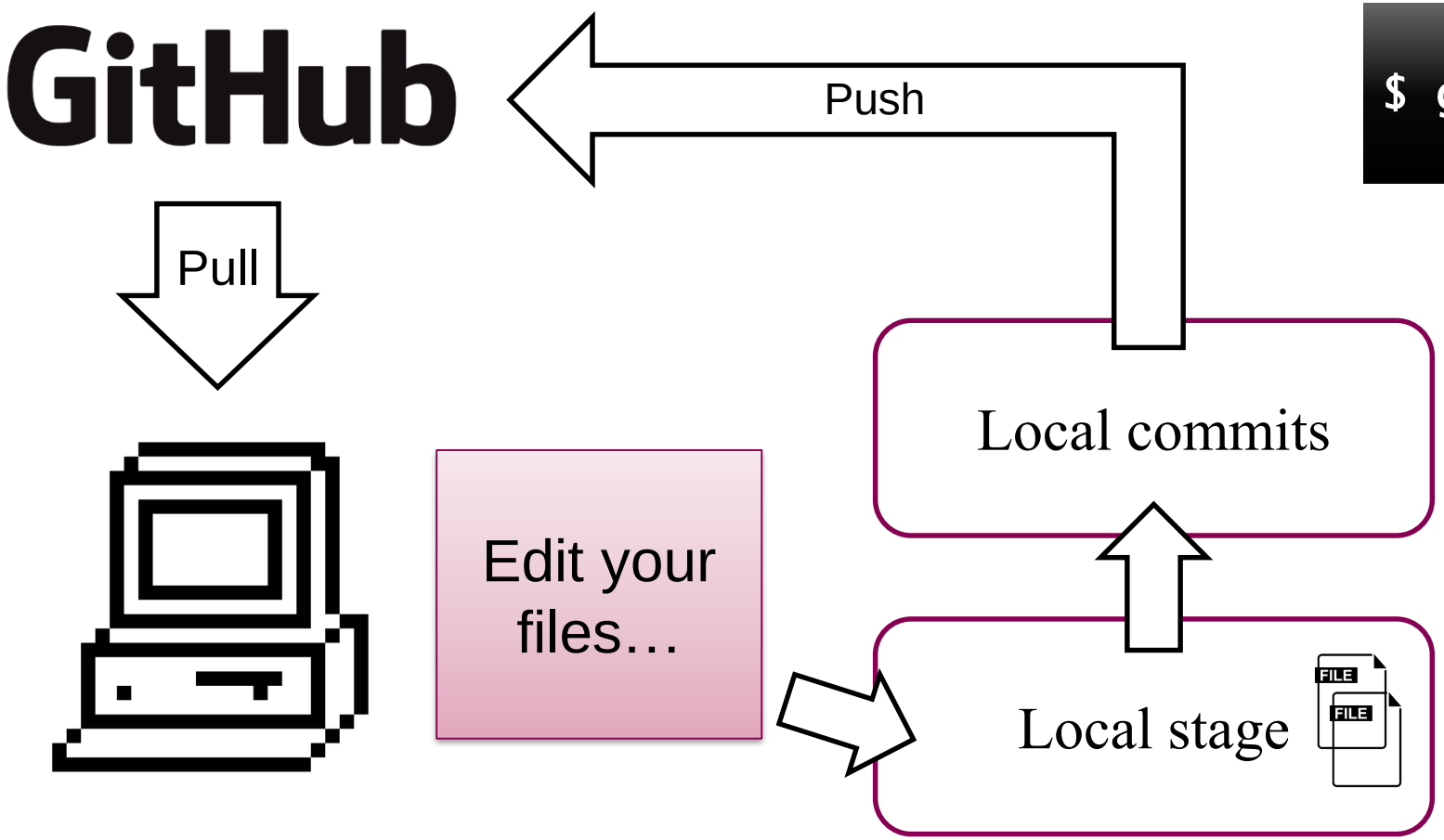
”changed filter settings”

”added files for plots”

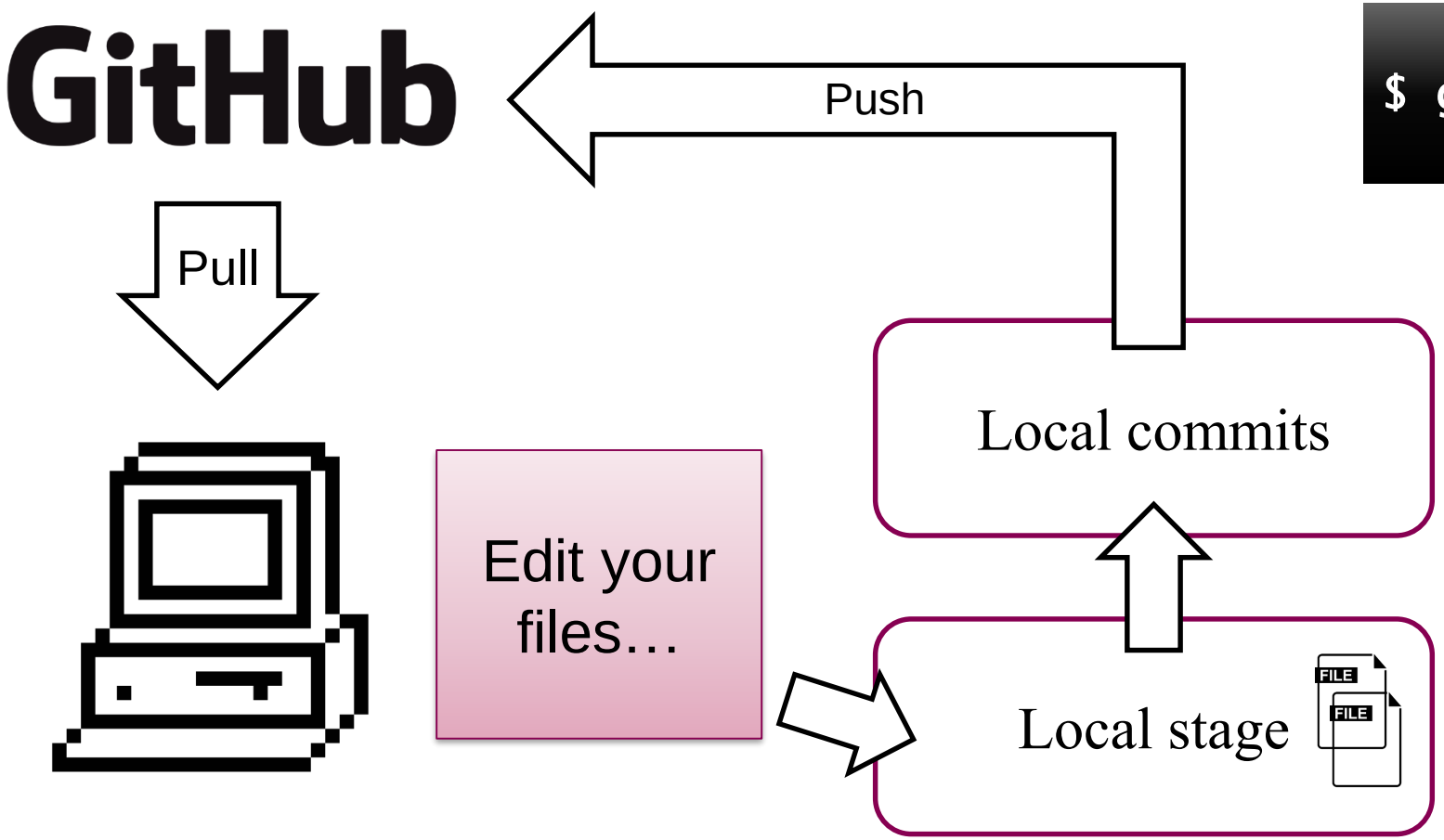
”overhaul of pre-processing

Too much

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




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




```
$ git push origin main
```

Git workflow

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

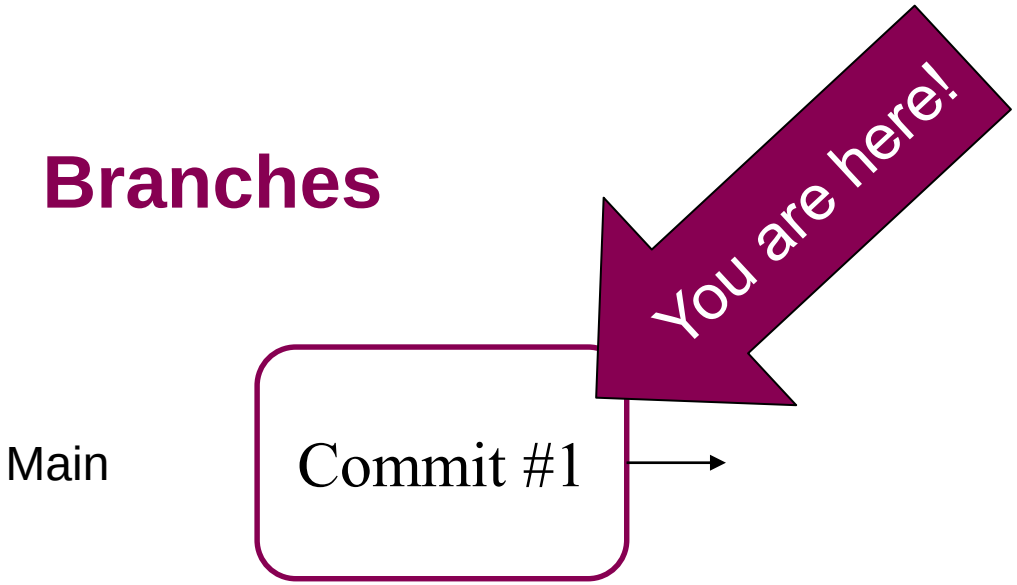
```
$ git pull origin main

$ git add <options>

$ git commit

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

Karolinska Institutet

The core of managing your code

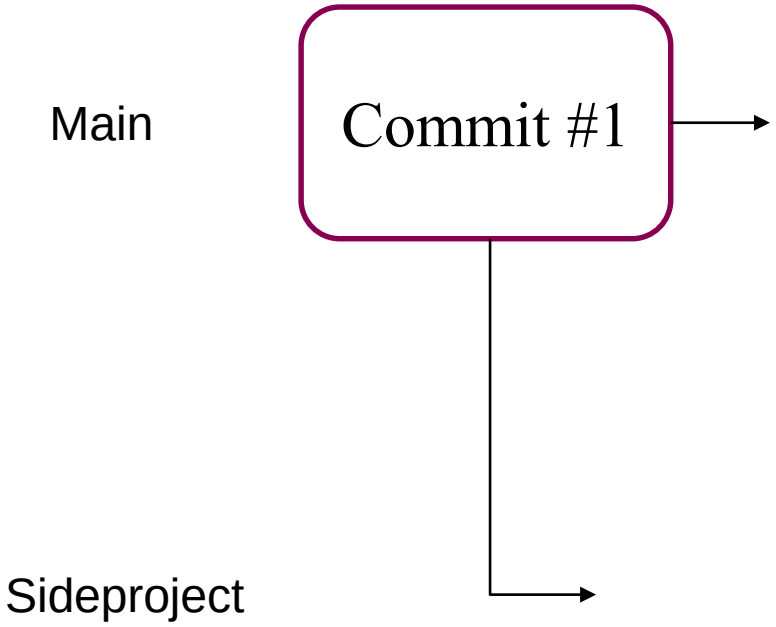
GIT BRANCHES

Branches




```
$ git branch
*  Main
```

Branches

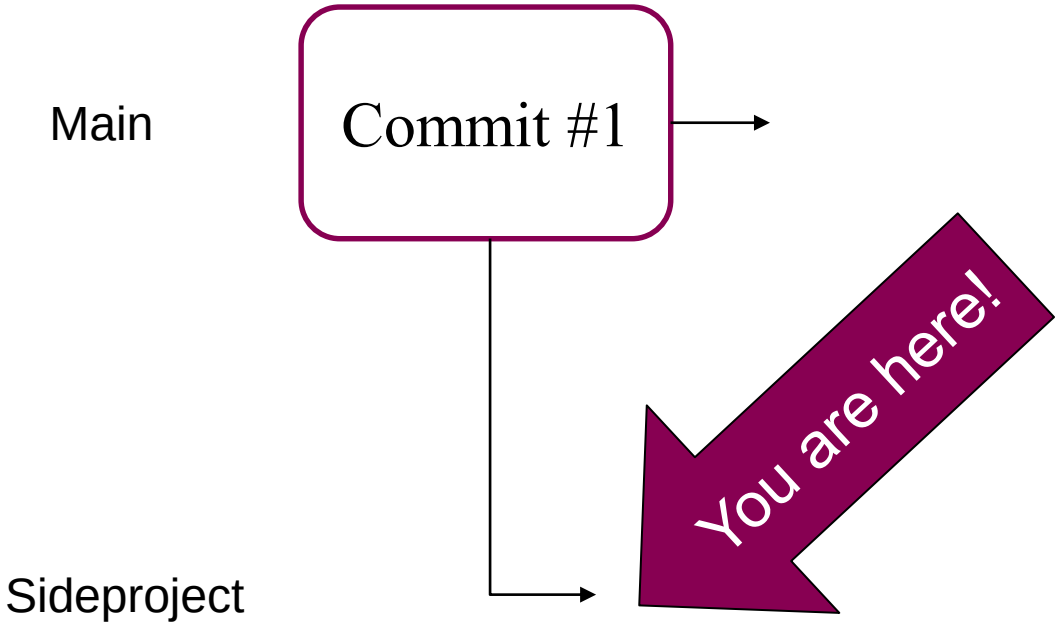



```
$ git branch
*  Main

$ git checkout -b sideproject
```

Create new flag

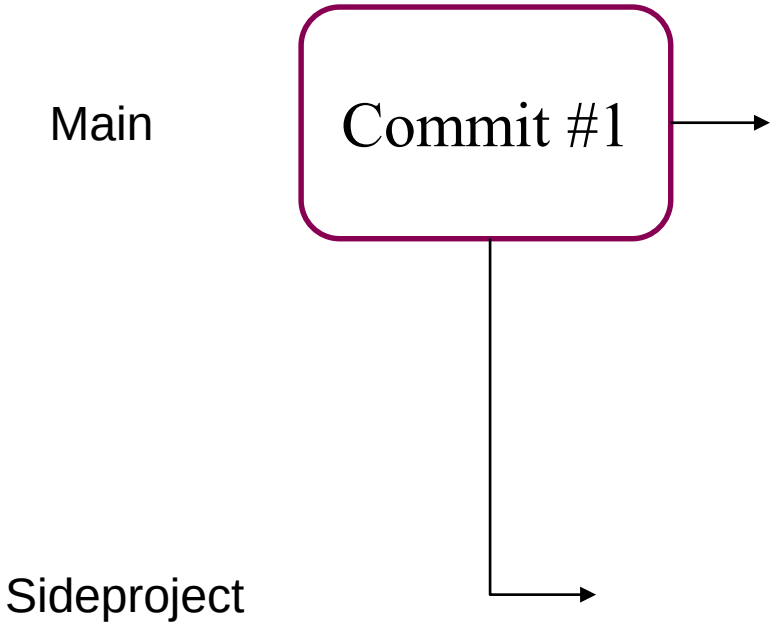
Name of new branch

Branches



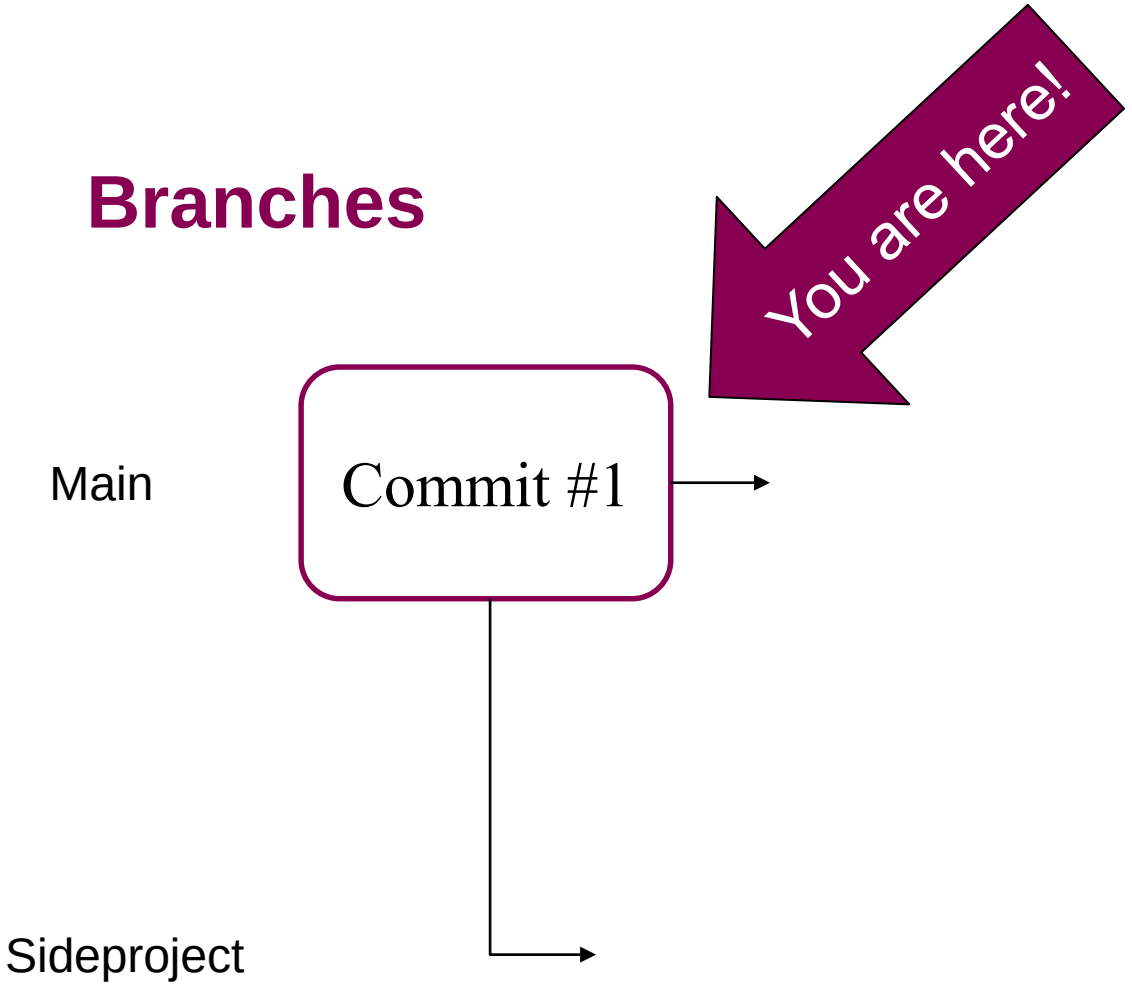

```
$ git branch
*  Main

$ git checkout -b sideproject

$ git branch
  main
* sideproject
```

Branches




```
$ git commit -m ”<…>”

$ git push origin sideproject
```

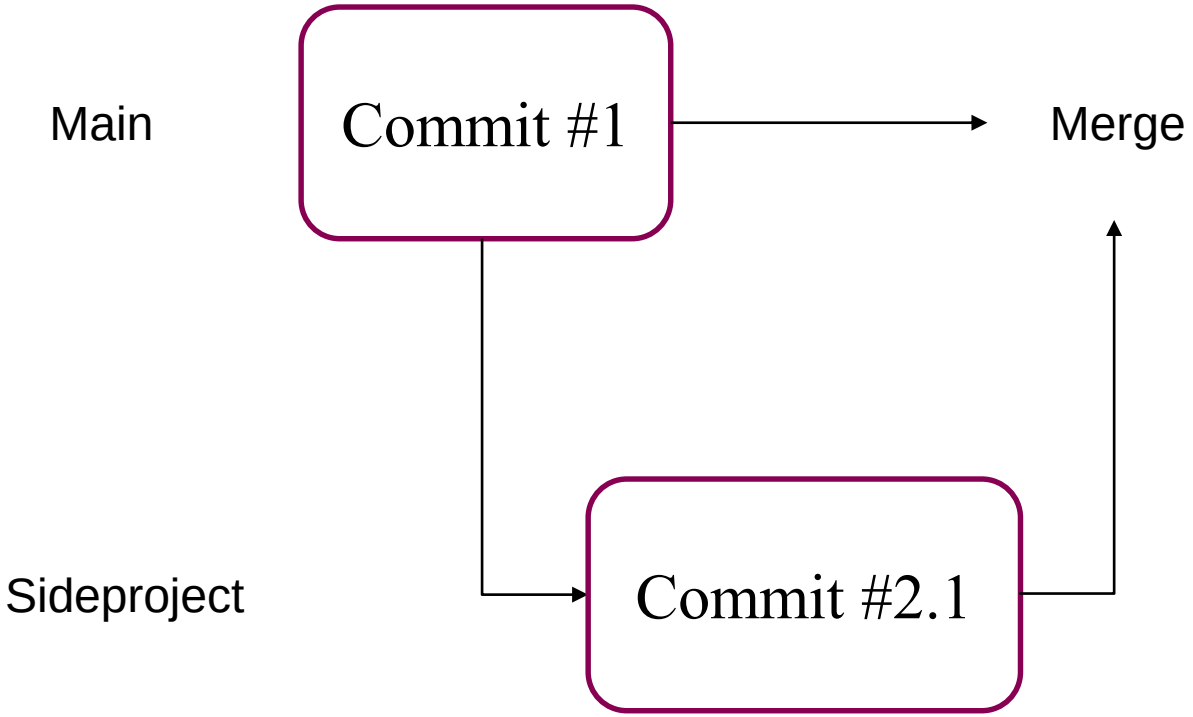
Branches



No ”-b” flag

```
$ git checkout main

$ git branch
*  main
   sideproject
```

Merge

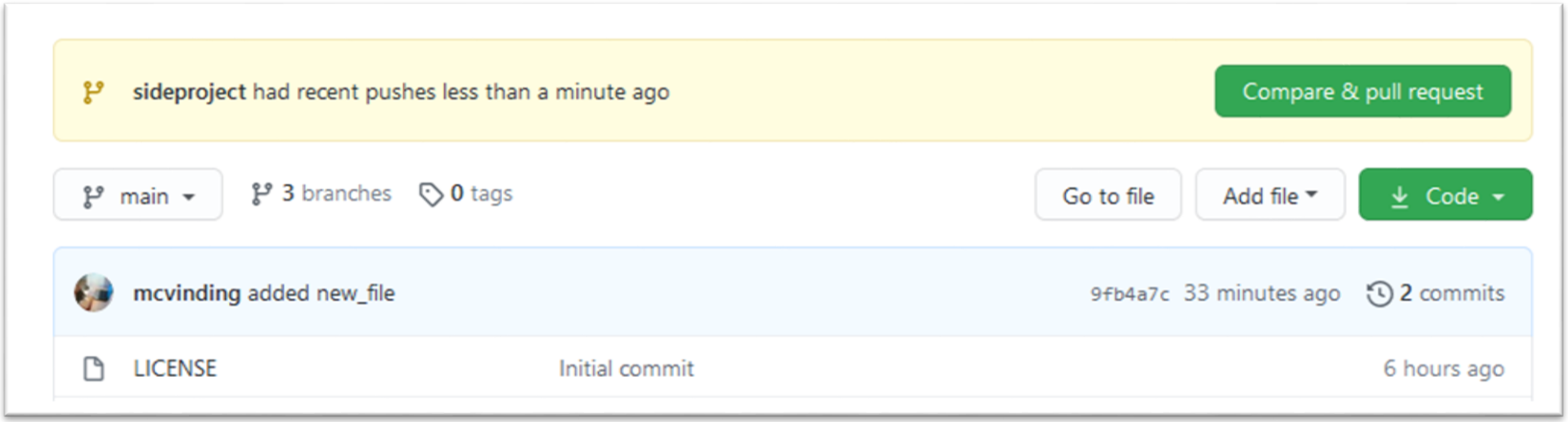



```
$ git checkout main

$ 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



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

Leave a comment

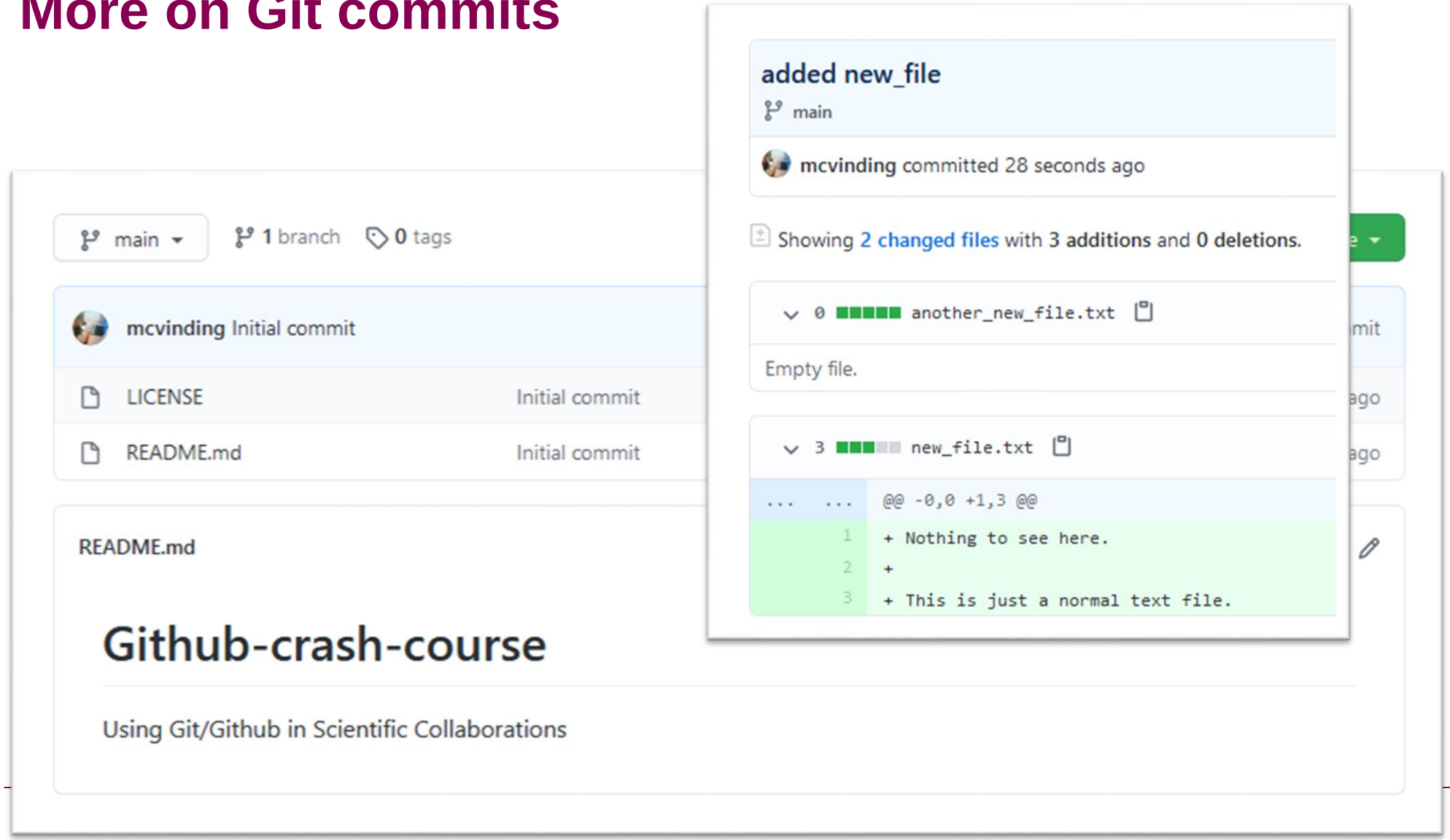
Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Karolinska Institutet

More on Git commits

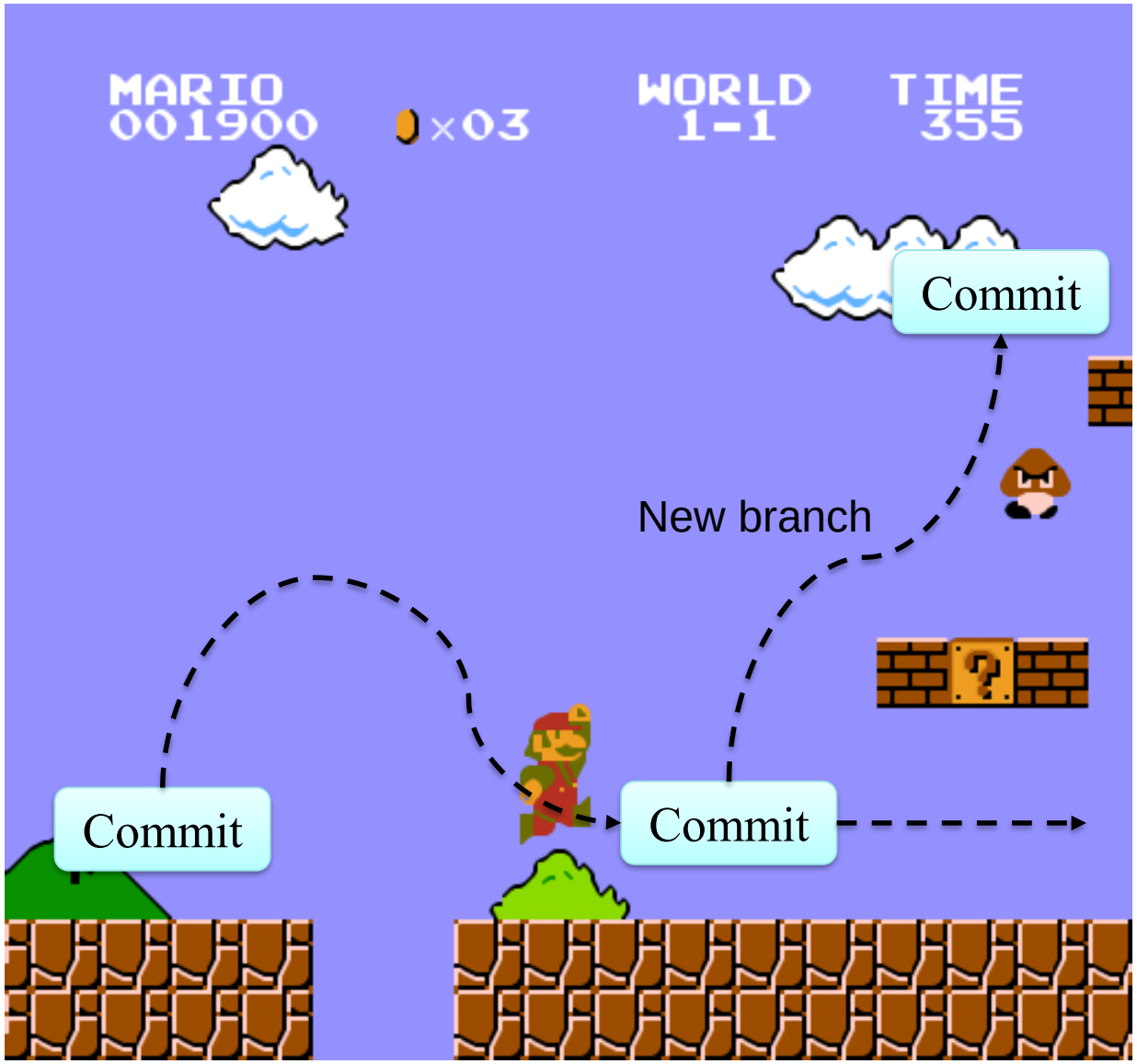


When to use branches

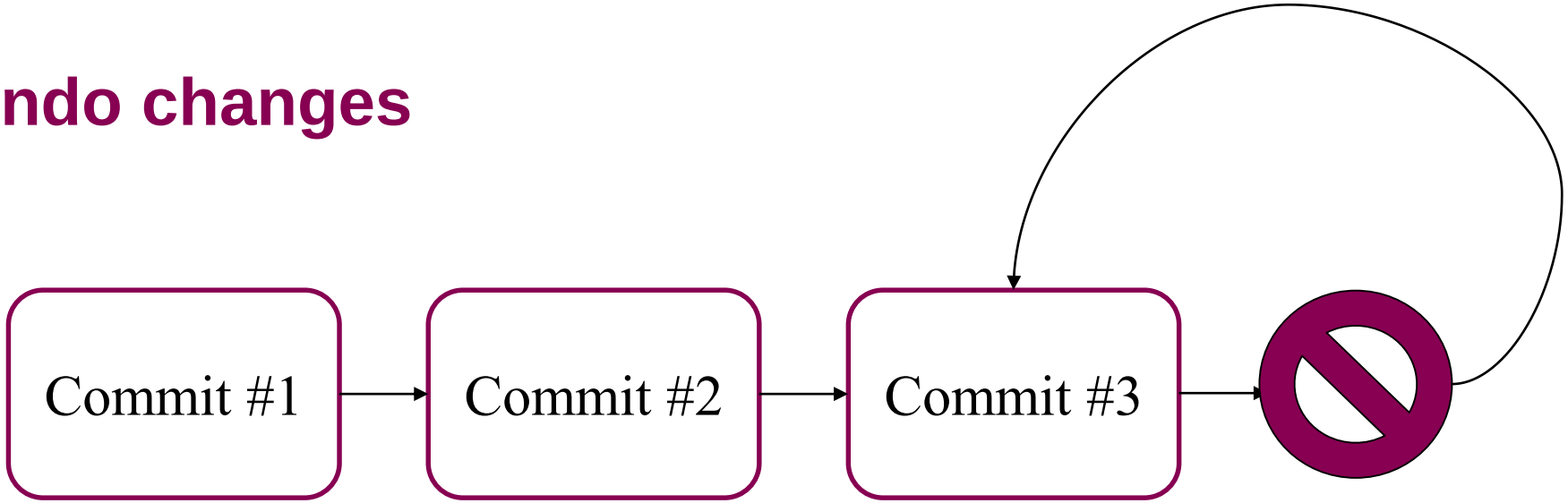
- When implementing entirely new features / analysis parts in existing code
- Testing out others code (and don't want to risk ruining it)

When not to use branches

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

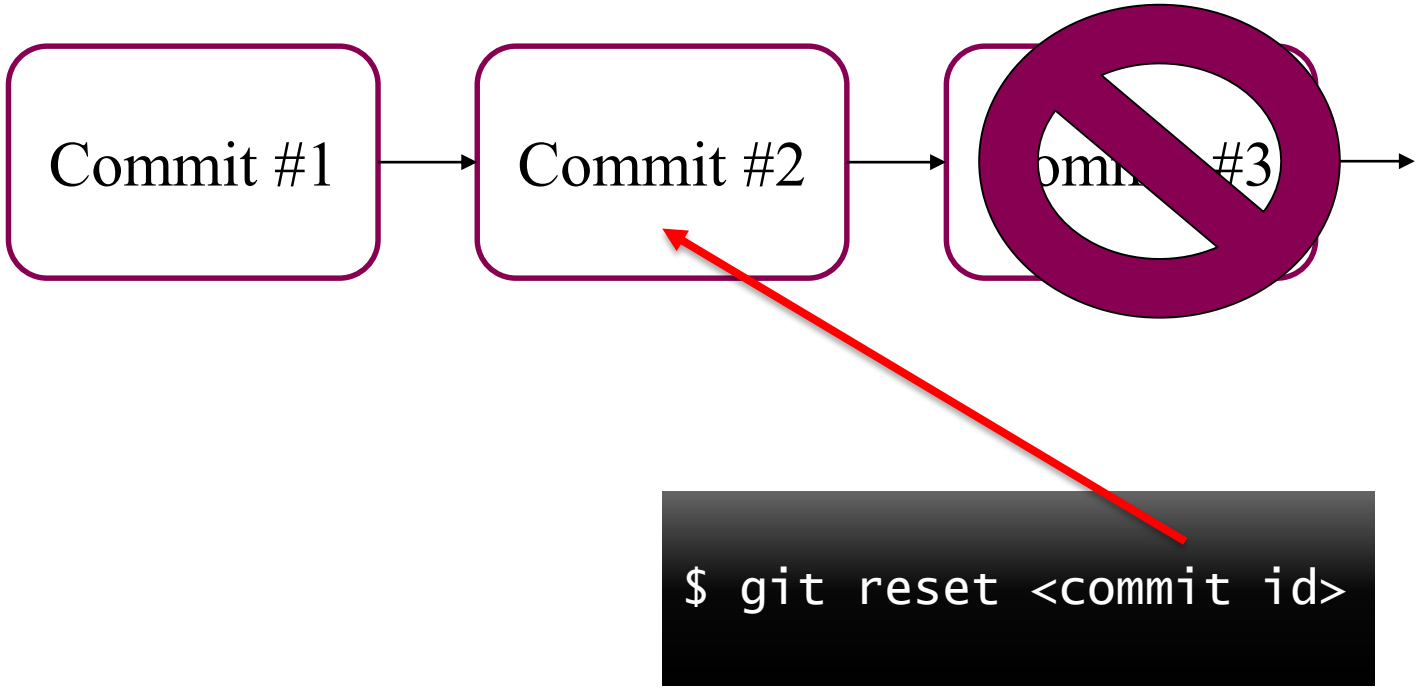

MARIO
001900
×03
WORLD
1-1
TIME
355
Commit
New branch
Commit
Commit

Undo changes

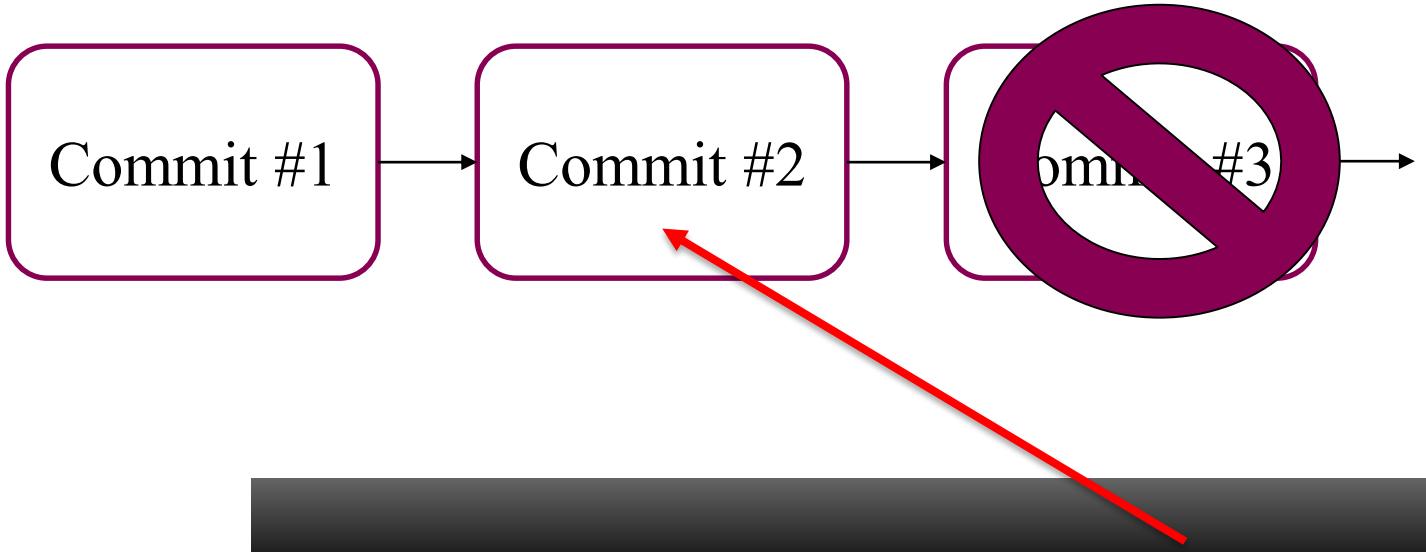



```
$ git reset --hard HEAD
```

Undo changes



Undo changes




```
$ git reset 4fa2ea669e898f2ba0549f50eada713ccd65cf89
```

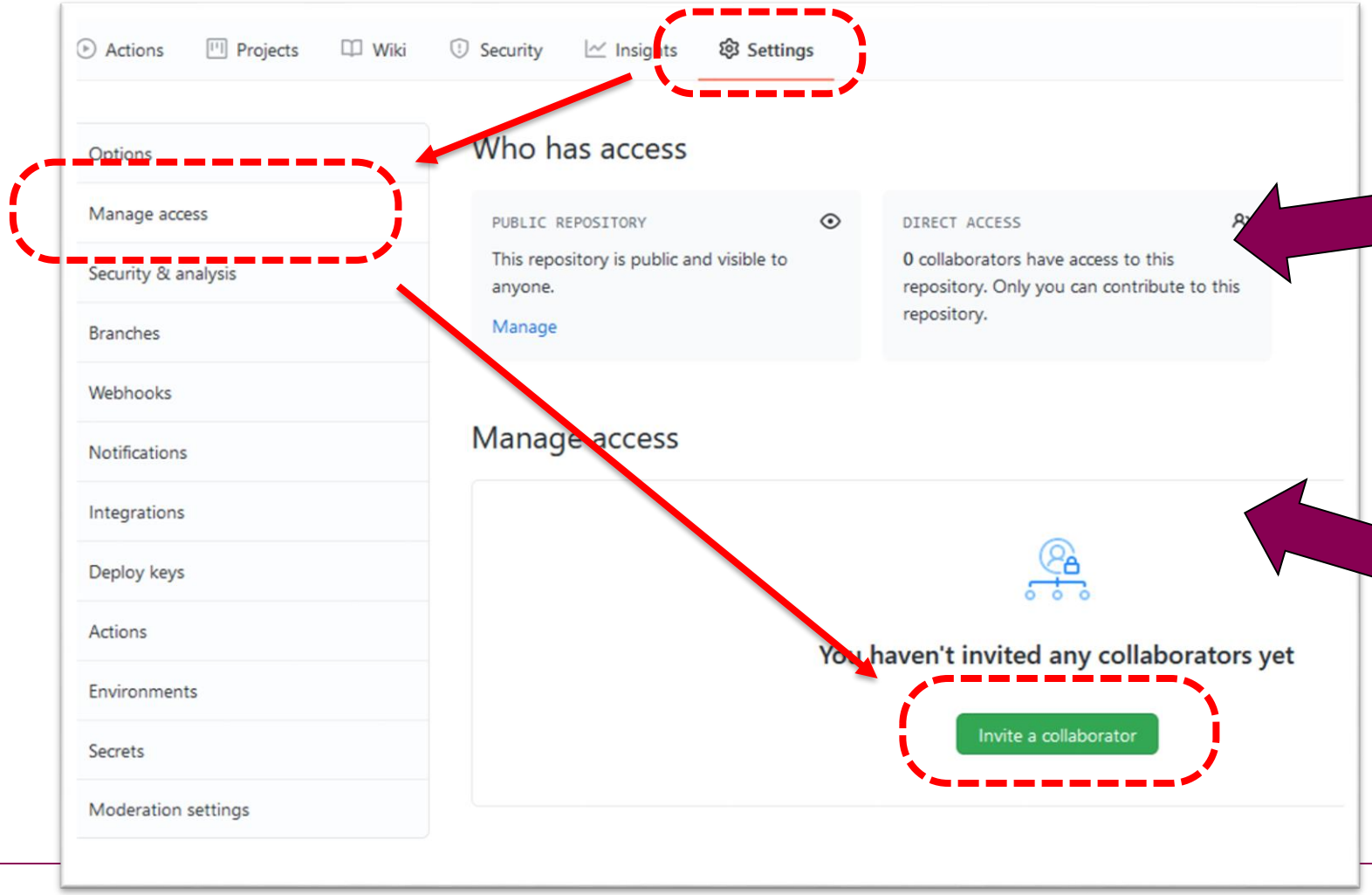
```
$ git log
commit 9fb4a7cbe74a87abccb5fd3fb1d04e529f3417bb (HEAD ->
main, origin/main)
Author: mcvinding <mikkel.vinding@gmail.com>
Date:   Mon Jan 20 16:33:56 2021 +0100

    added new_file

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

    Initial commit
```

Karolinska Institutet

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

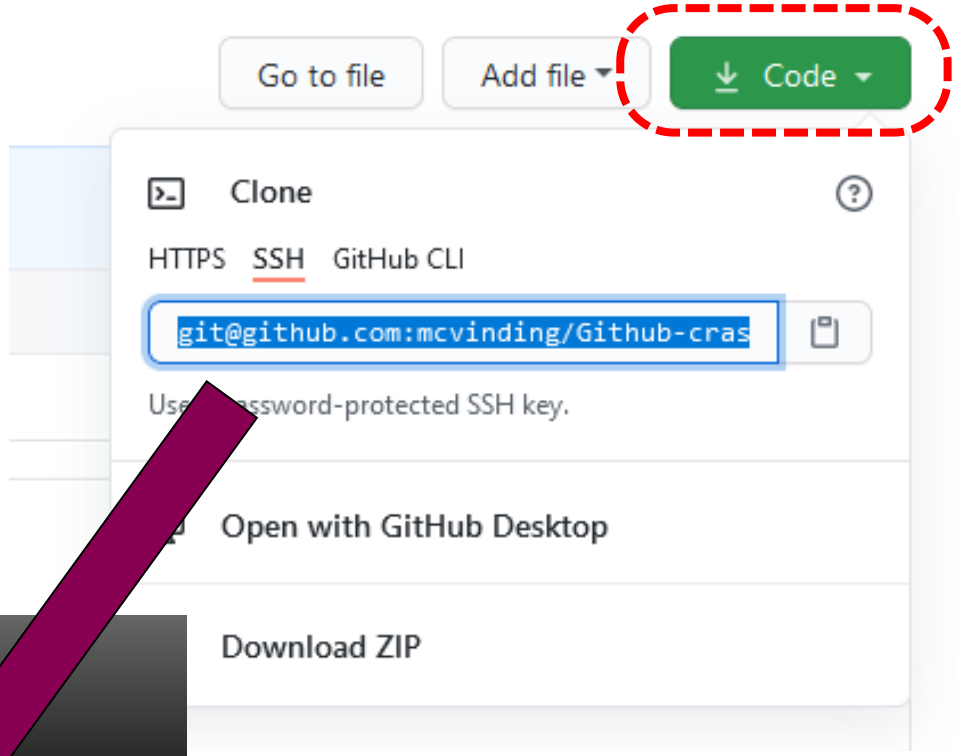
GITHUB COLLABORATIONS

Adding collaborators



Setting up collaboration

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

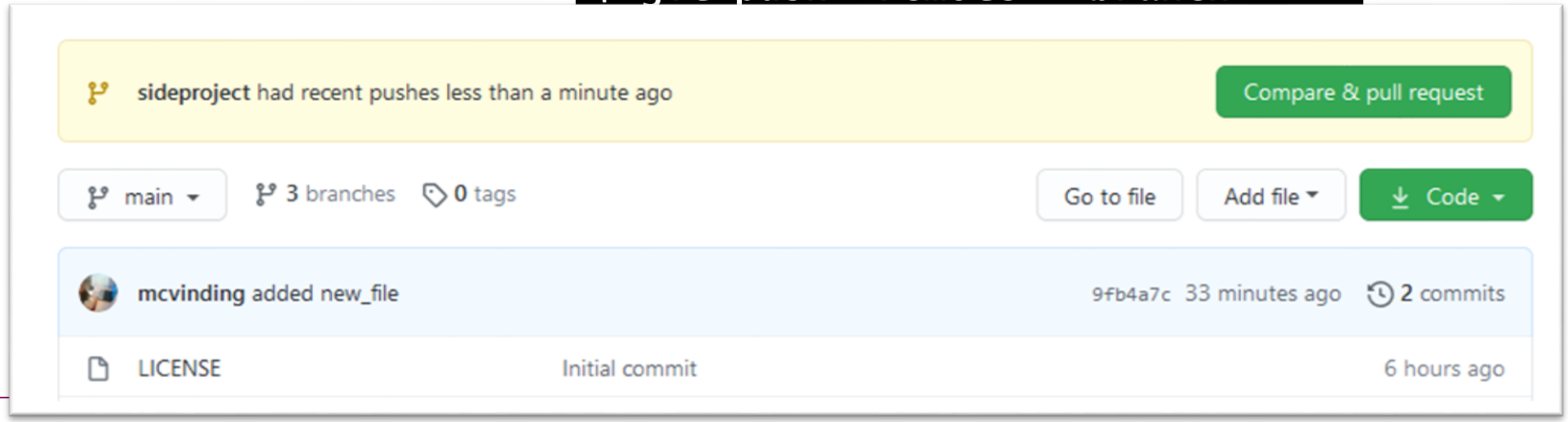


```
$ git init

$ 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>
```