Introduction to version control with Git

Day 1: Concepts and a basic workflow

Selina Baldauf

November 11, 2024

Who am I?

- 💻 **Scientific programmer** @ theoretical ecology group
- 🎓 PhD in **dryland ecology** modelling dryland ecohydrology
- 👩‍🏫 **Teaching** R, Git, good scientific practice, …

What do you want to learn here?

… use Git and Github effectively

… work with others on projects

… roll back things (safely)

… make my code publicly available on GitHub

… write good commit messages?

… contribute to an open source project

…

Aims of the workshop

Git is very powerful …

… but can also be confusing in the beginning.

🎯 Learn simple Git workflows in **theory and practice** that you can **immediately apply** to your research projects.

xkcd on Git

Topics

Today 2 - 4 pm

Introduction to **Git concepts** and a **simple workflow** for your **individual projects**

Tomorrow 2 - 4 pm

Collaborate using Git and GitHub

Next Monday 2 - 3 pm

Q&A session and/or more advanced topics

Until then: work with Git on your own projects

Organization

- Material is all online
  - View and download slides, tasks and more from there
  - Will stay online after the workshop
- Certificate of attendance from the graduate center
- All questions and comments are welcome
- Feedback is welcome (Evaluation at the end of the workshop)
- If possible, please turn on your camera

Before we start

Did anyone have problems with the workshop preparation?

- Install Git
- Install GitHub Desktop
- Get a GitHub account and connect it with GitHub Desktop

Let’s get started

Why version control?

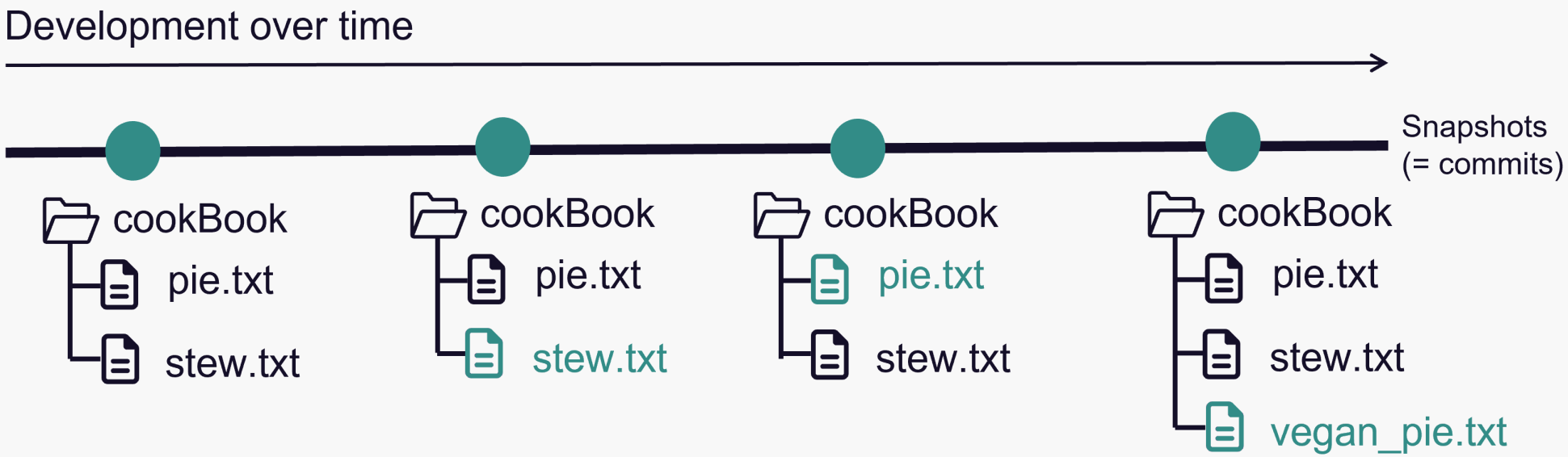
Two examples in which proper version control can be a time/stress saver

Version control with Git

- **Complete** and **long-term** history of every file in your project
- **Open source and free** to use version control software
- Quasi **standard** for software development
- A whole universe of **other software and services** around it

Version control with Git

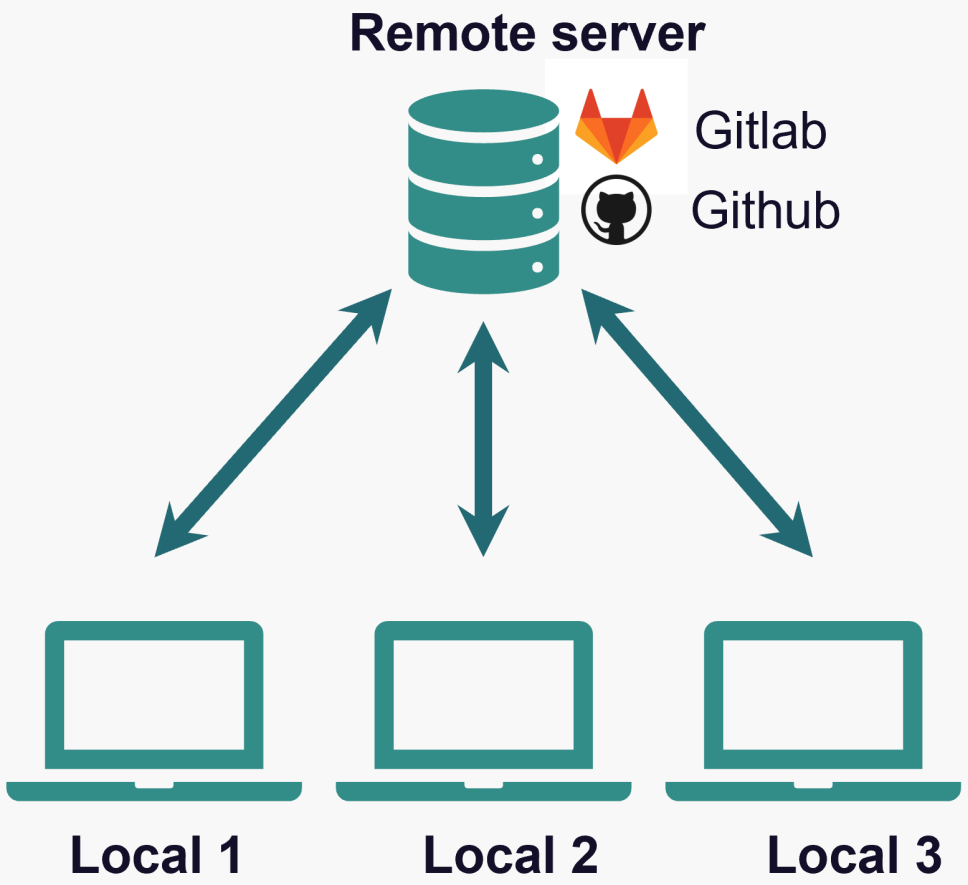
- For projects with mainly text files (e.g. code, markdown files, …)
- Basic idea: Take snapshots (**commits**) of your project over time



- A project version controlled with Git is a Git **repository** (**repo**)

Version control with Git

Git is a **distributed version control system**



- Idea: many *local* repositories synced via one *remote* repo
- Everyone has a complete copy of the repo

How to use Git

After you installed it there are different ways to interact with the software.

How to use Git - Terminal

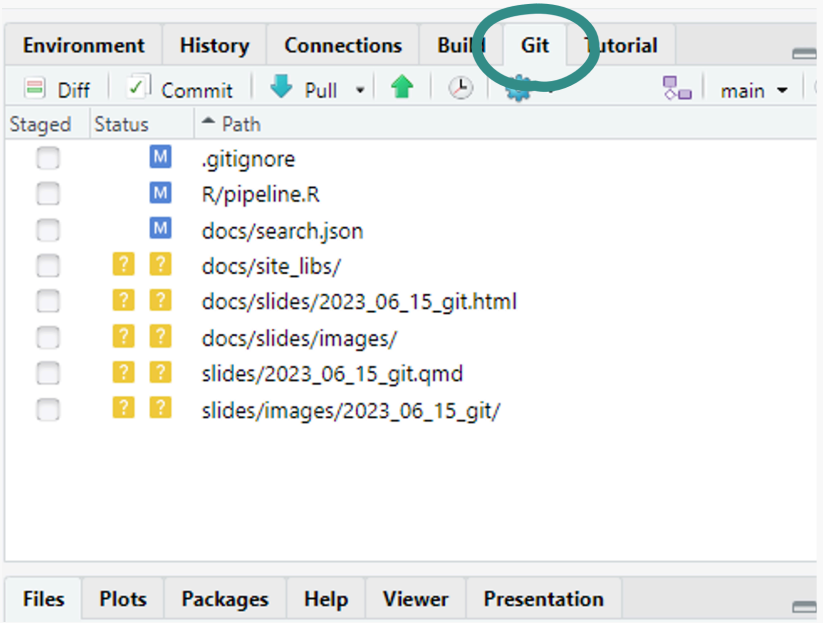
Using Git from the terminal

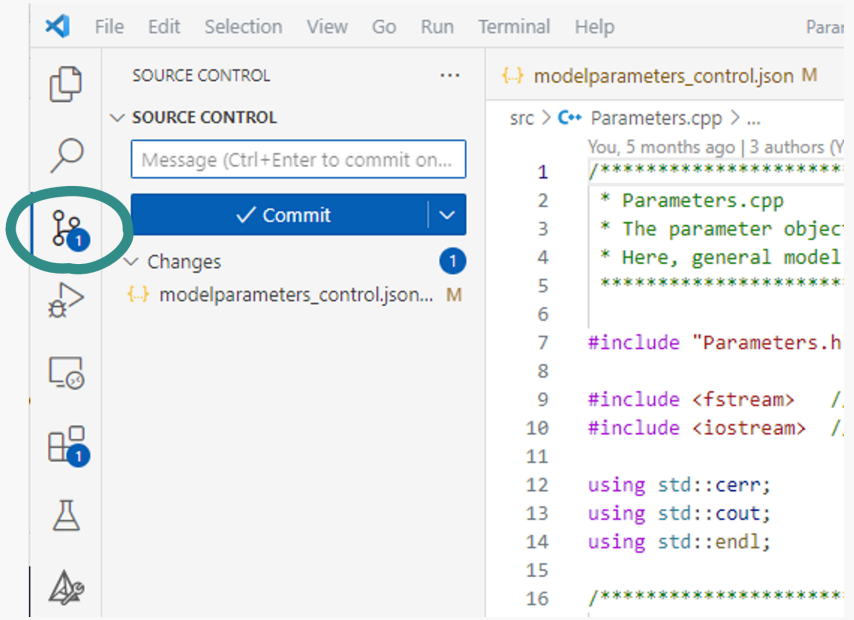
```
Selina_User@DESKTOP-G0RM7MS MINGW64 ~/Files_Selina
$ cd Repos/02_workshops/first_git_project/

Selina_User@DESKTOP-G0RM7MS MINGW64 ~/Files_Selina/Repos/02_workshops/first_git_
project
$ git init
Initialized empty Git repository in C:/Users/Selina_User/Files_Selina/Repos/02_w
orkshops/first_git_project/.git/

Selina_User@DESKTOP-G0RM7MS MINGW64 ~/Files_Selina/Repos/02_workshops/first_git_
project (master)
$
```

- \+ Most control
- \+ A lot of help/answers online
- − You need to use terminal 😱

How to use Git - Integrated GUIs

A Git GUI is integrated in most (all?) IDEs, e.g. R Studio, VS Code





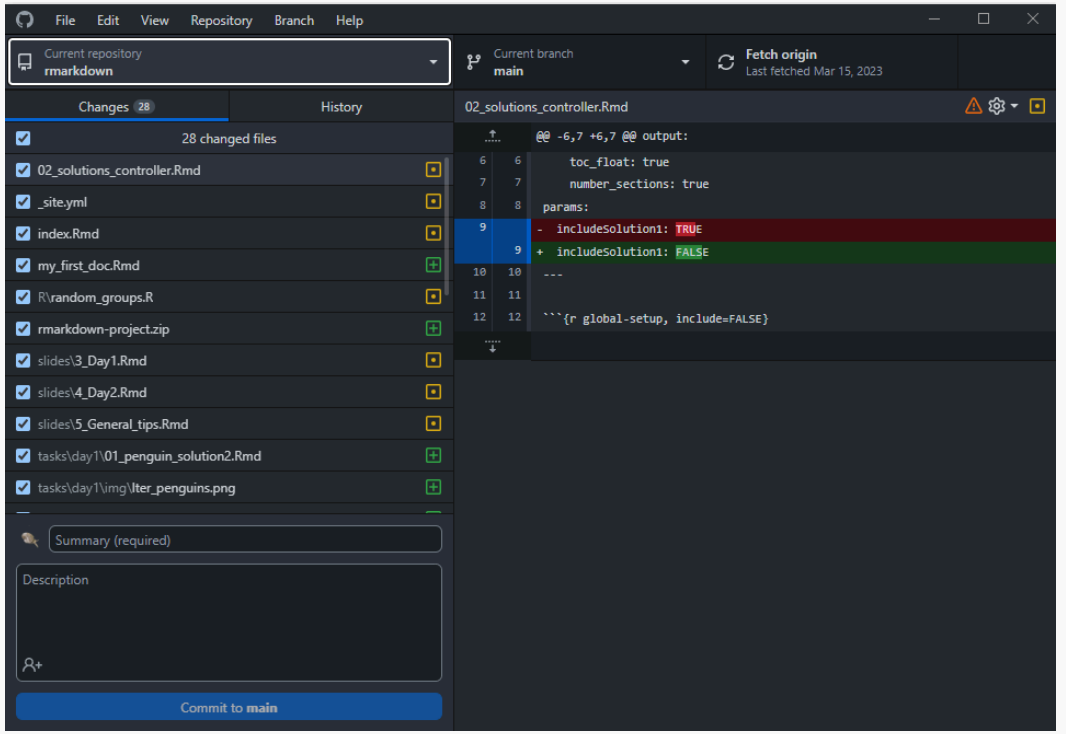
\+ Easy and intuitive

\+ Stay inside IDE

− Different for every program

How to use Git - Standalone GUIs

Standalone Git GUI software, e.g. GitHub Desktop, Source Tree, ...



+ Easy and intuitive
+ Use for all projects

− Switch programs to use Git

How to use Git

Which one to choose?

- Depends on experience and taste
- You can mix methods because they are all interfaces to the same Git

- We will use GitHub Desktop
  - Beginner-friendly, intuitive and convenient
  - Nice integration with GitHub

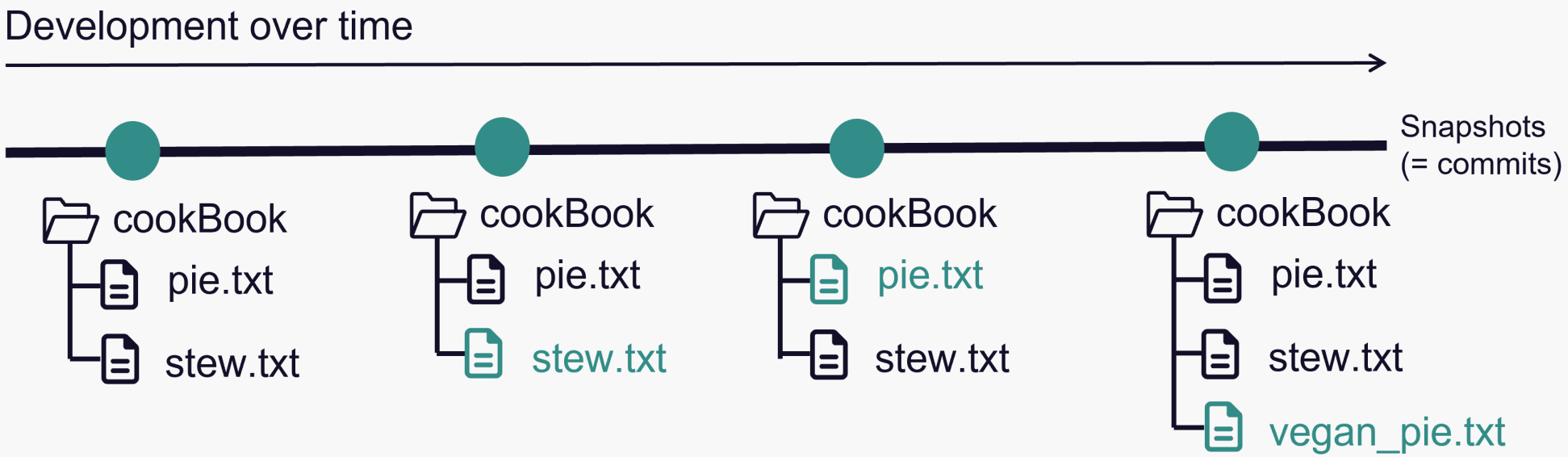
> Tip
>
> Have a look at the website where you find **How-To guides for the other methods** as well.

The basic Git workflow

> `git init`, `git add`, `git commit`, `git push`

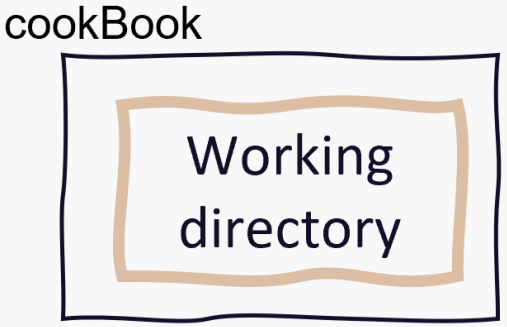
Example

A cook book project to collect all my favorite recipes.

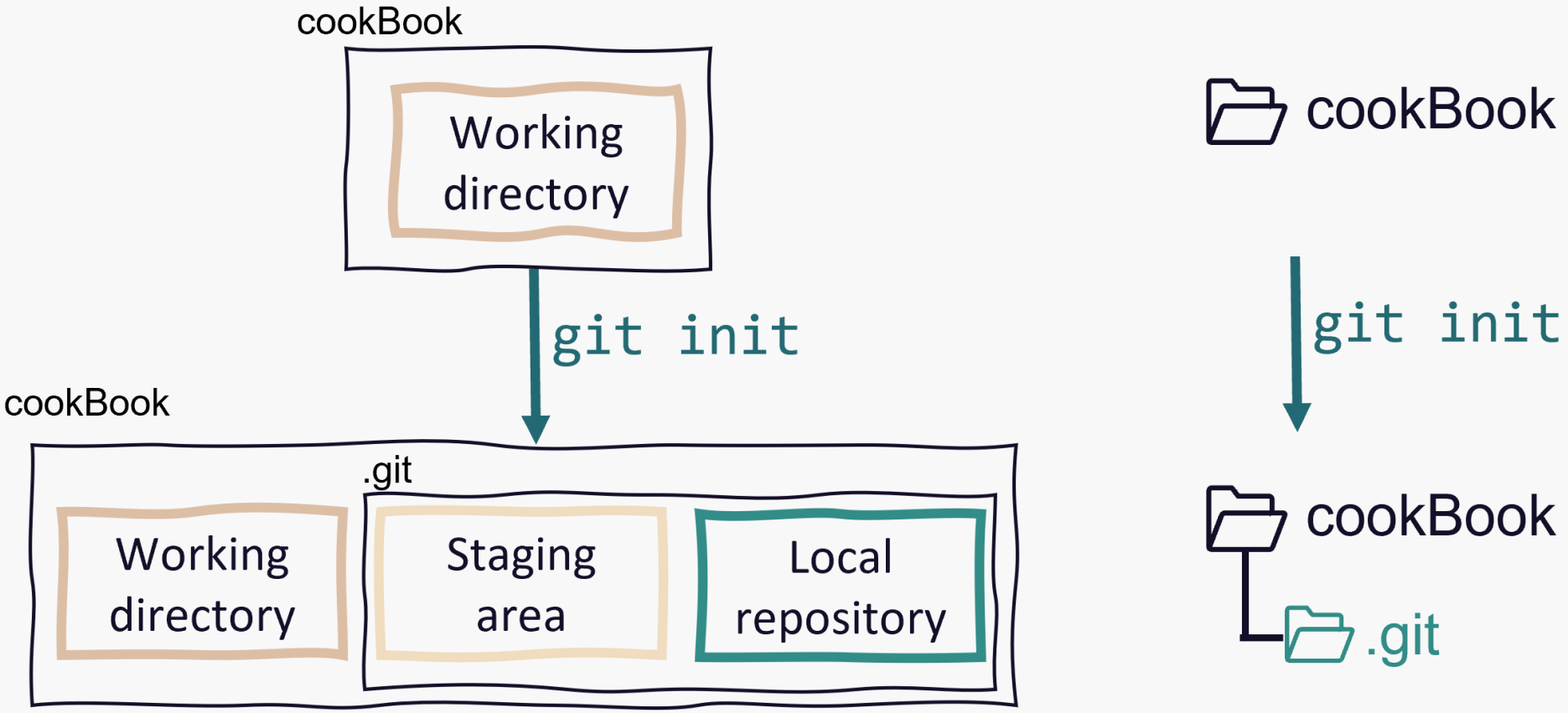


In real life this would be e.g. a data analysis project, your thesis in LaTex, a software project, …

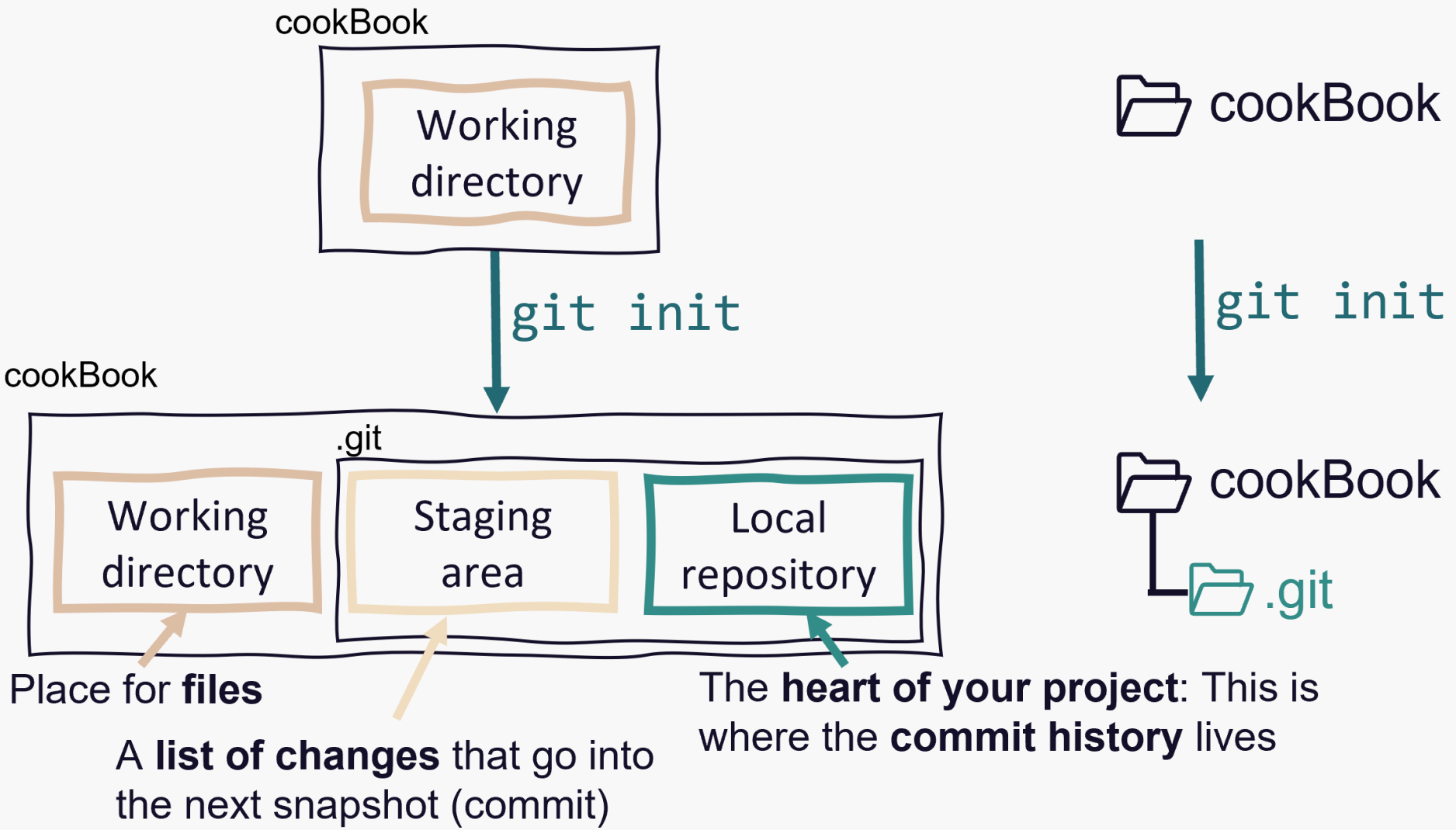
Step 1: Initialize a Git repository



Step 1: Initialize a Git repository

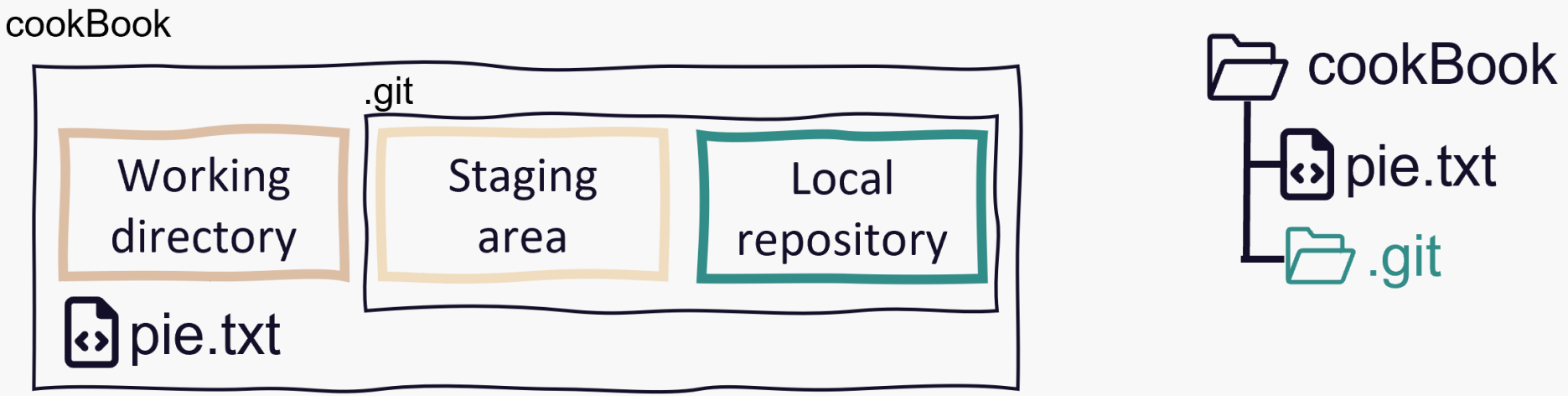


Step 1: Initialize a Git repository



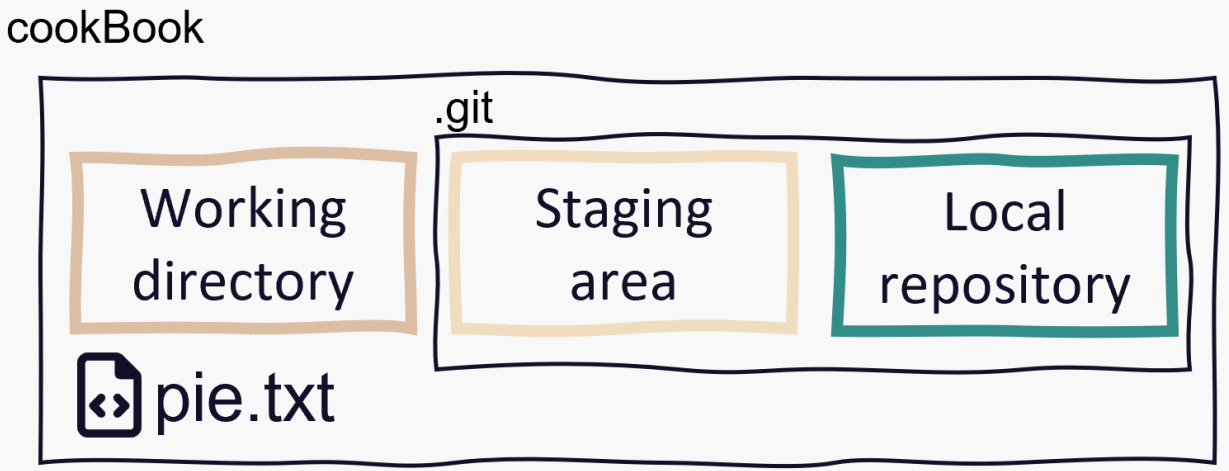
Step 2: Add and modify files

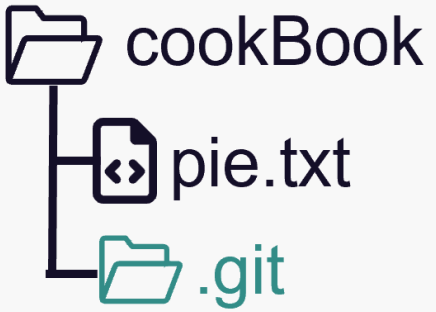
Git detects any changes in the working directory



Step 2: Stage changes

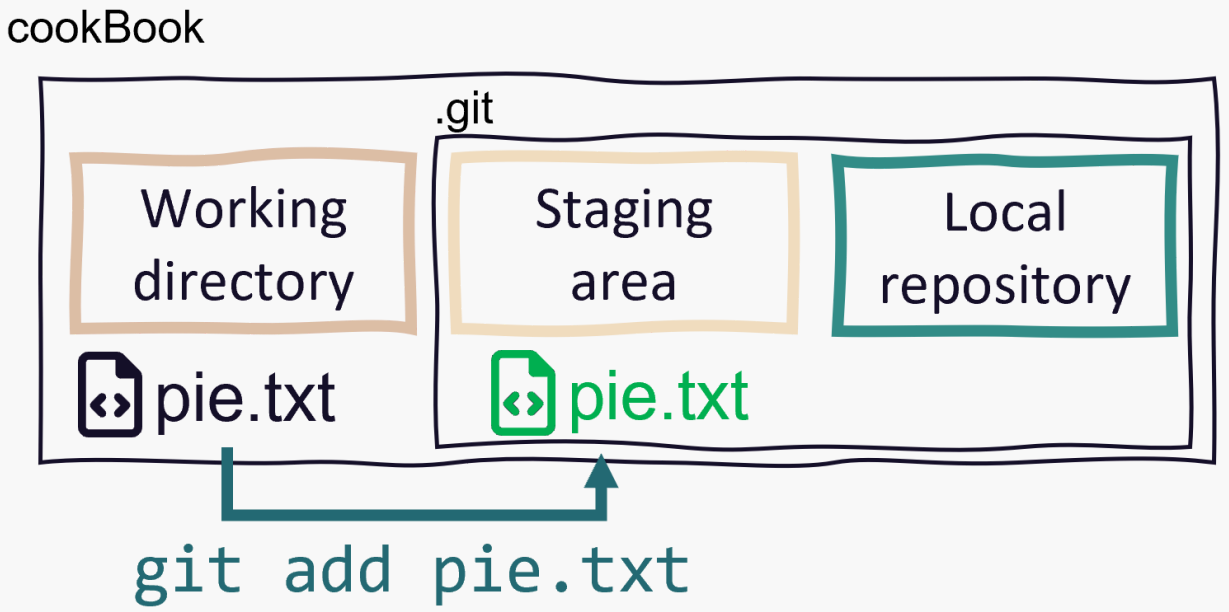
Staging a file means to **list it for the next commit**.

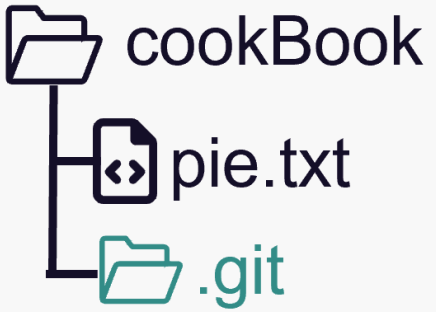




Step 2: Stage changes

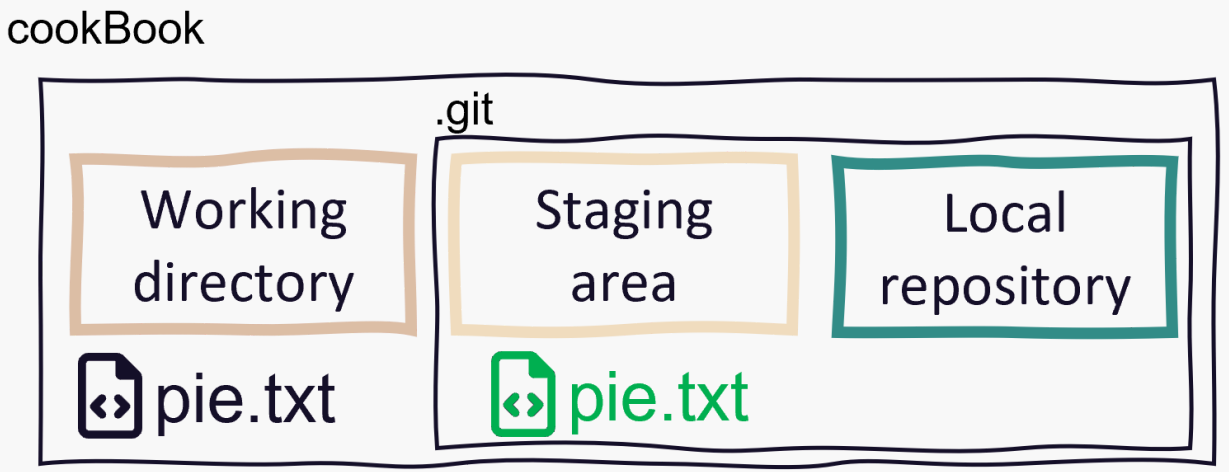
Staging a file means to **list it for the next commit**.

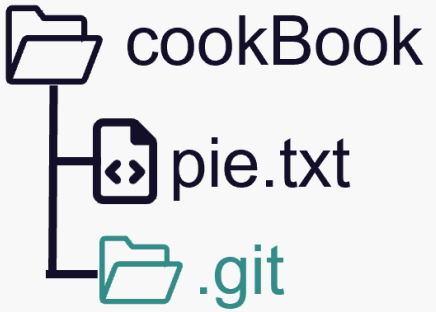




Step 3: Commit changes

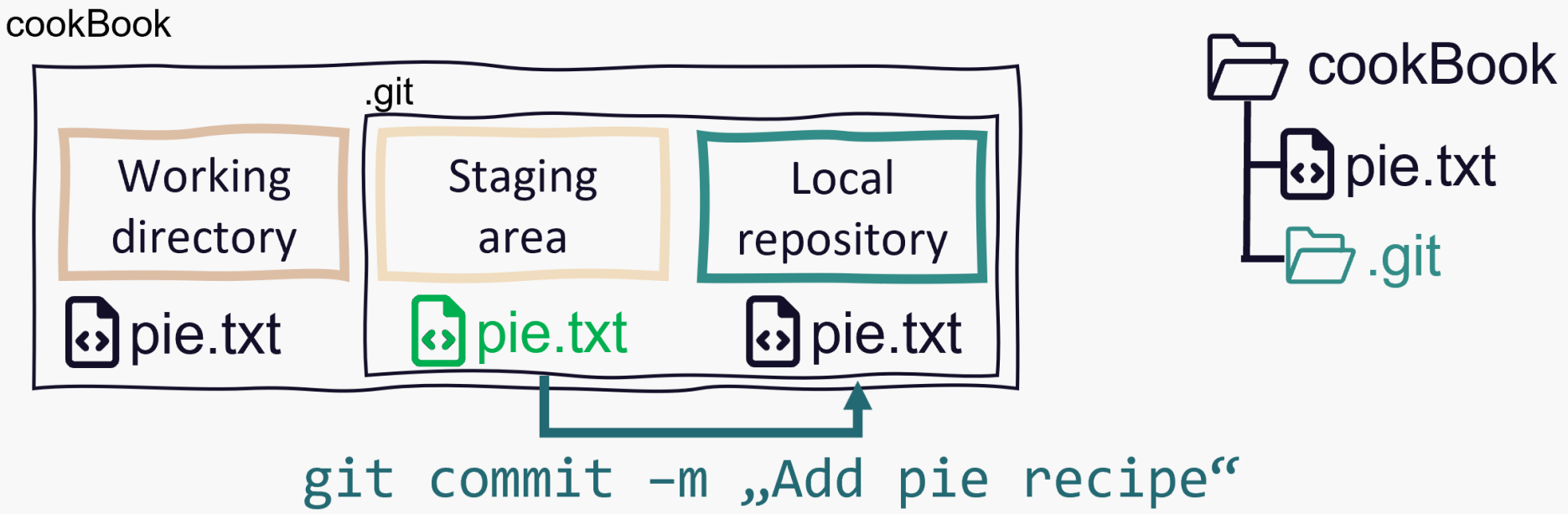
Commits are the snapshots of your project state





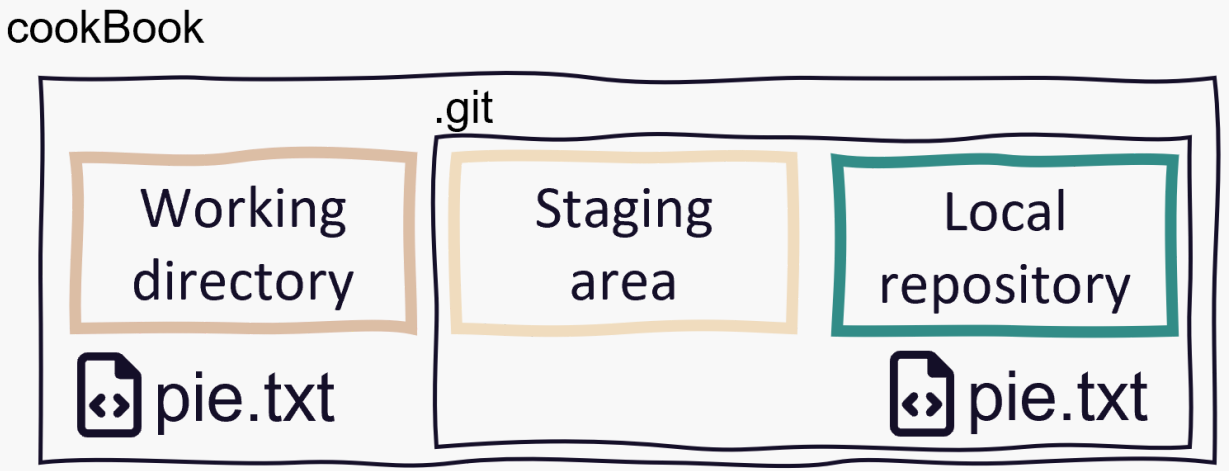
Step 3: Commit changes

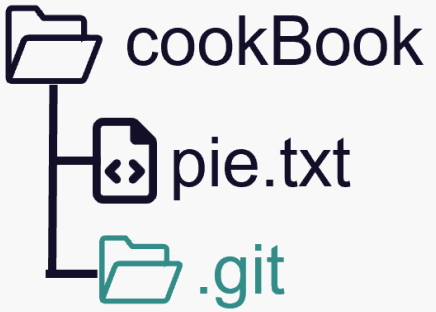
Commits are the snapshots of your project state



Step 3: Commit changes

Changes are part of Git history and staging area is clear again





How to write good commit messages?

xkcd on commit messages

How to write good commit messages?

✔

```
Add pie recipe

This is my favorite pie in the world.
The recipe comes from my grandfather and
he learned it from his neighbor.
```

✘

```
added a file.

This is really good.
```

See here for more details but some general rules:

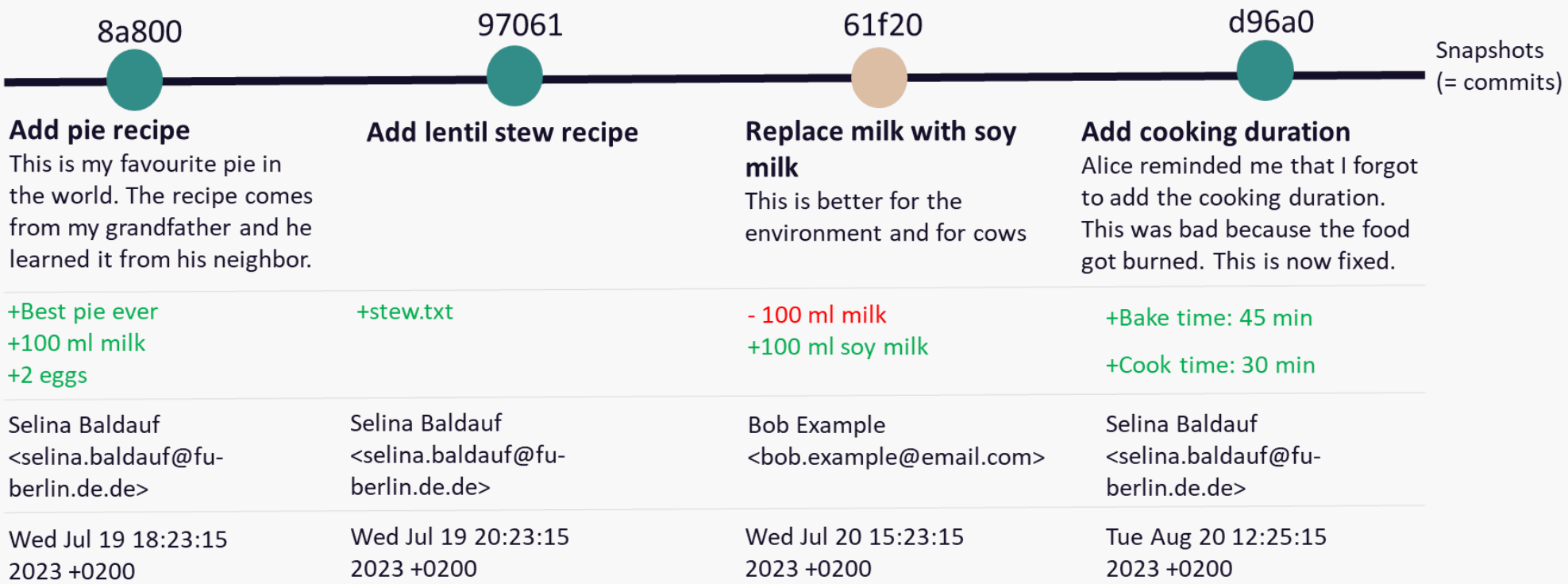
1. Limit summary line to 50 characters
2. Capitalize summary line
3. Do not end summary line with period
4. Use imperative mood in the subject line
5. Use the *Description* to explain **what** and **why**, not **how**

Now you

> Start your own cook book
> Complete Task 1 “Local repo” (10 - 15 min)

Stay in the meeting for the task.
Ask if you are stuck.
Turn down/off volume if you are disturbed.

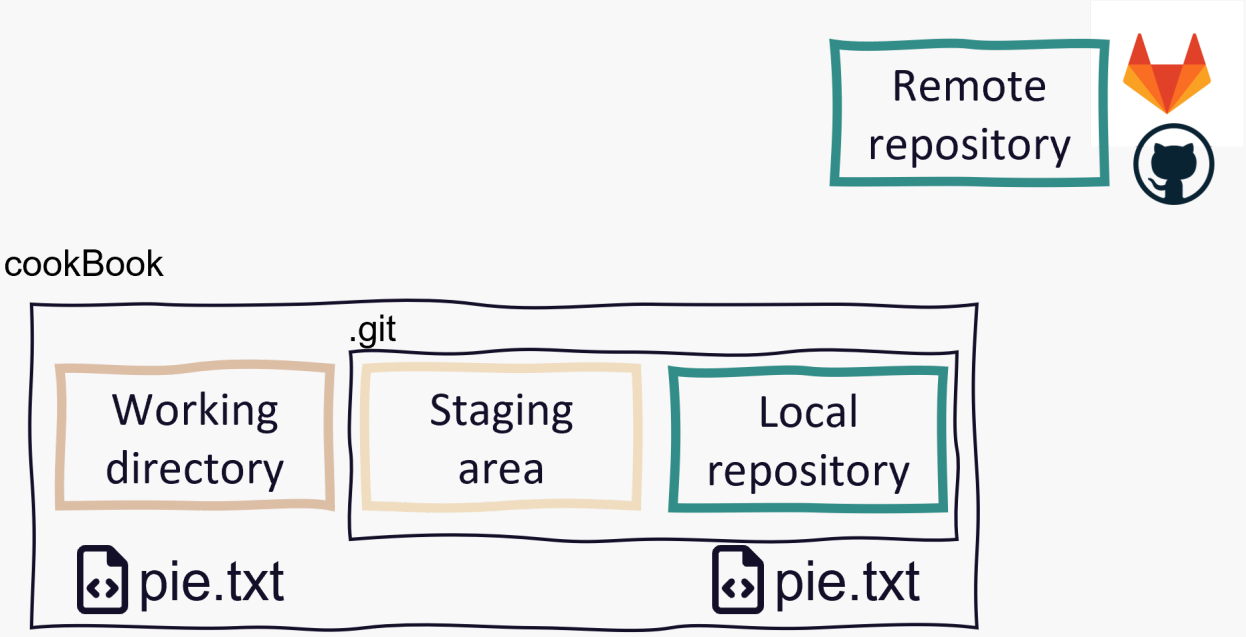
The commit history



Step 4: Share changes with the remote repo

Use remote repos (on a server) to **synchronize**, **share** and **collaborate**

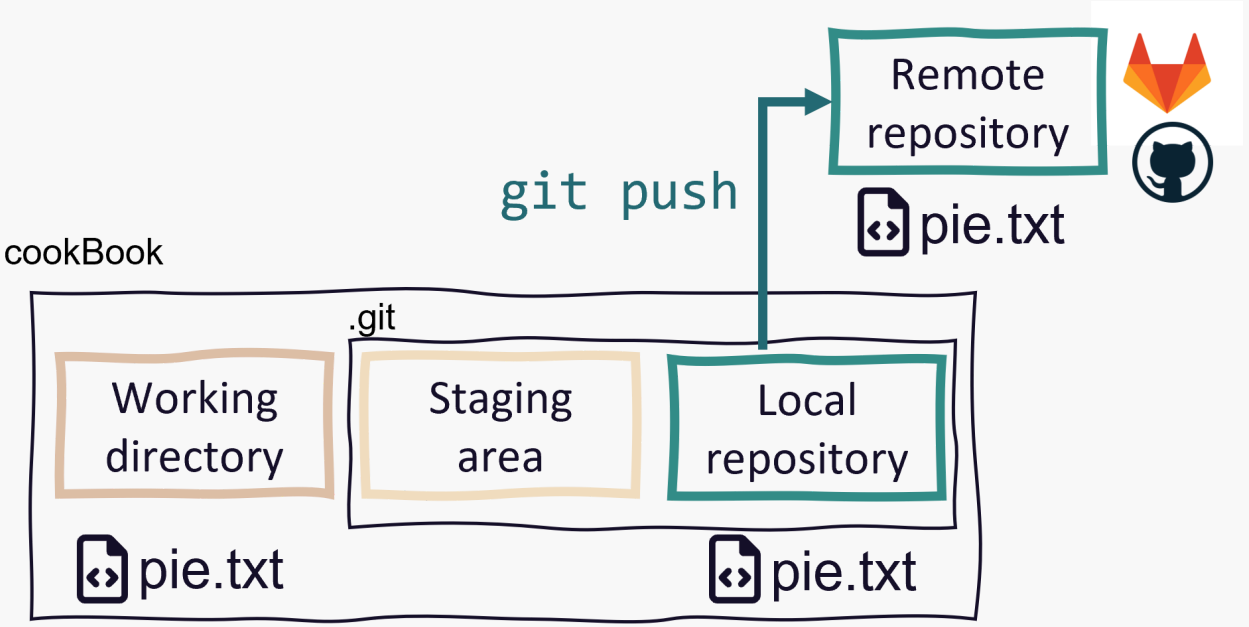
- Remote repos can be **private** (you + collaborators) or **public** (visible to anyone)



Step 4: Share changes with the remote repo

Use remote repos (on a server) to **synchronize**, **share** and **collaborate**

- Remote repos can be **private** (you + collaborators) or **public** (visible to anyone)



Different remote repositories

- There are **commercial** and **self-hosted** options for your remote repositories
  - Commercial: GitHub, Gitlab, Bitbucket, …
  - Self-hosted: Gitlab (maybe at your institution?)
- Please be aware of your institutional guidelines
  - Servers outside EU
  - Privacy rules might apply depending on type of data

Make your repositories public

GitHub/Gitlab are a good way to publish and share your work.

Advantages of publishing your code

- Others can build on your work
- Citations
- Reproducibility
- Get feedback

Make your repositories public

You can increase the quality/complexity of your repo by

- Adding a nice README.md file
- Connecting the repo with Zenodo to get a DOI
- ...

If you are interested, browse some nice GitHub repositories for inspiration (e.g. Git training repository, Computational notebooks, Repo to publish code from a manuscript)

Now you

> Publish your cook book on GitHub
> Checkout your repositories GitHub page (**Repository -> View on GitHub**) 5 min

Summary of the basic steps

- `git init`: Initialize a git repository
  - Adds a `.git` folder to your working directory
- `git add`: Add files to the staging area
  - This marks the files as being part of the next commit
- `git commit`: Take a snapshot of your current project version
  - Includes time stamp, commit message and information on the person who did the commit
- `git push`: Push new commits to the remote repository
  - Sync your local project version with the remote e.g. on GitHub

Undo things

> `git revert`

Revert changes

- Use `git revert` to revert specific commits
- This does not delete the commit, it creates a **new commit that undoes a previous commit**
  - It's a safe way to undo commited changes

Now you

> Revert a commit from your cook book

Thanks for your attention

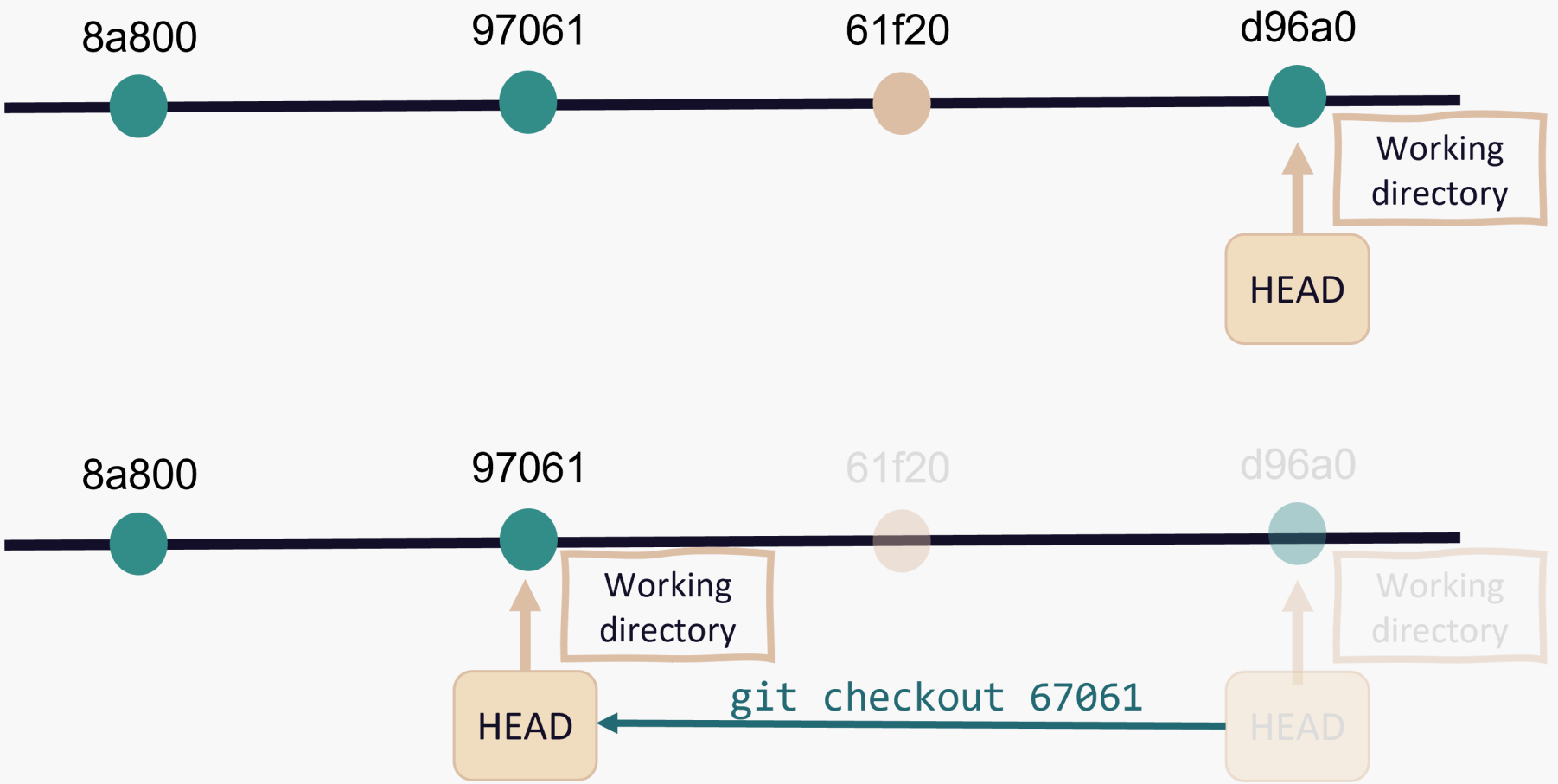
Questions?

Go back in time

> `git checkout`

Checkout a previous commit

Take your work space back in time temporarily with `git checkout`



Ignoring files with `.gitignore`

Ignore files with `.gitignore`

- Useful to ignore e.g.
  - Compiled code and build directories
  - Log files
  - Hidden system files
  - Personal IDE config files
  - …

Ignore files with `.gitignore`

- Create a file with the name `.gitignore` in working directory
- Add all files and directories you want to ignore to the `.gitignore` file

Example

```
*.html    # ignore all .html files
*.pdf     # ignore all .pdf files

debug.log # ignore the file debug.log

build/    # ignore all files in subdirectory build
```

See here for more ignore patterns that you can use.

Preparation for tomorrow

- Tomorow we have teams of 2
- Collaborate on the cook book of your team mate

Now please

1. Enter your GitHub Account Name and the link to your repo here
2. Look for the GitHub Name of your team mate and add them as a collaborator to your repository
3. Accept the invitation of your team mate to their repository

- You will get an Email or you can do it on GitHub