Introduction to version control with Git

Day 1: Concepts and individual Git workflow

Selina Baldauf

September 27, 2025

Welcome

- 🎓 Ecology/Theoretical Ecology
- 💻 **Scientific programmer** @ Freie Universität Berlin
- 👩‍🏫 **Teaching** R, Git, good scientific practice, …

Reach out

✉ selina.baldauf@fu-berlin.de @selina-b

Before we start

I ❤️ questions

Did anyone have problems with the workshop preparation?

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

Workshop overview

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

- Day 1: Git concepts and individual Git workflow
- Day 2: Collaborative Git workflow

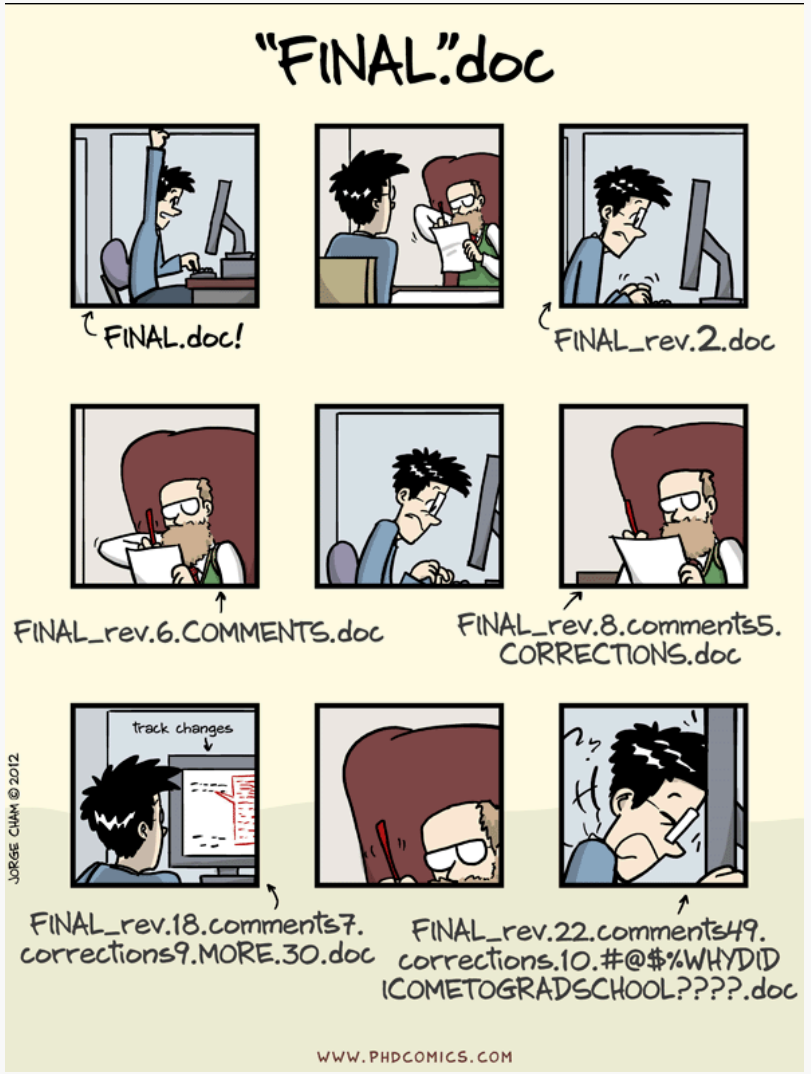
Organization

Material is all online

- View and download slides
- Tasks for Hands-on
- Step by step guides for different workflows
- Will stay online after the workshop

Let’s get started

It's never final



Jorge Cham (PhD comics)

Requirements for good version control

- [ ] Complete and long-term history of every file in your project
- [ ] Safe (e.g. no accidental loss of versions)
- [ ] Easy to use
- [ ] Document all changes: what, why, who, when
- [ ] Allow for collaboration
- [ ] Online version for backup and sharing
- [ ] Offline version for working on the project

Requirements for good version control

- [x] Complete and long-term history of every file in your project
- [x] Safe (e.g. no accidental loss of versions)
- [x] Easy to use
- [x] Document all changes: what, why, who, when
- [x] Allow for collaboration
- [x] Online version for backup and sharing
- [x] Offline version for working on the project

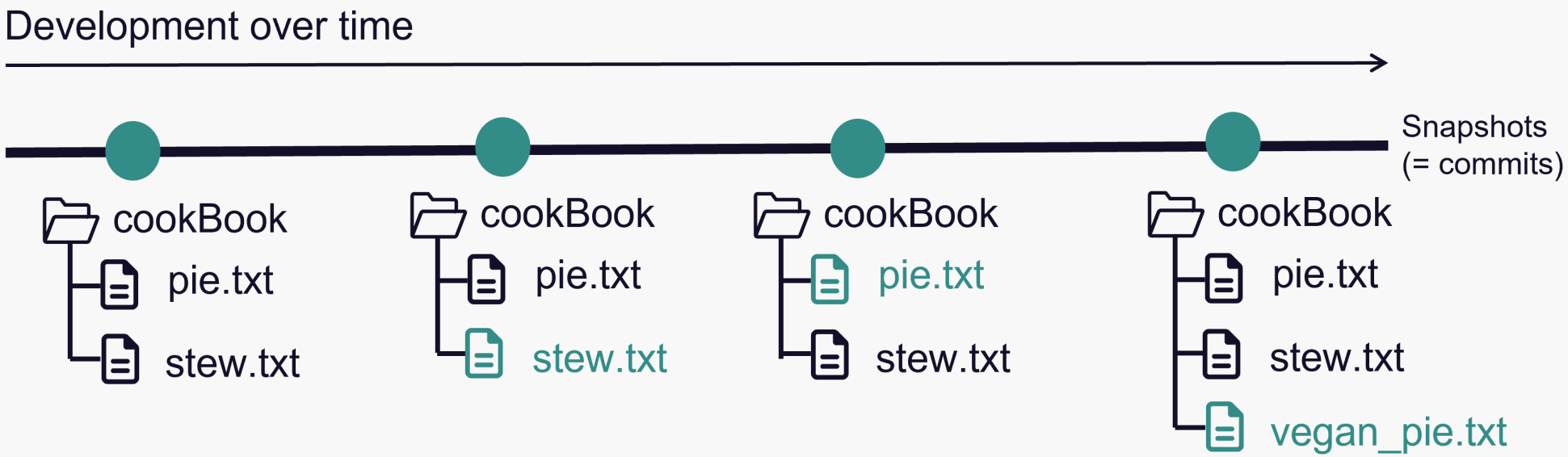
Git checks all the boxes!

What is Git?

- **Open source and free** to use version control software
- Quasi **standard** for software development
- A whole universe of **other software and services** built on top of it

What is Git?

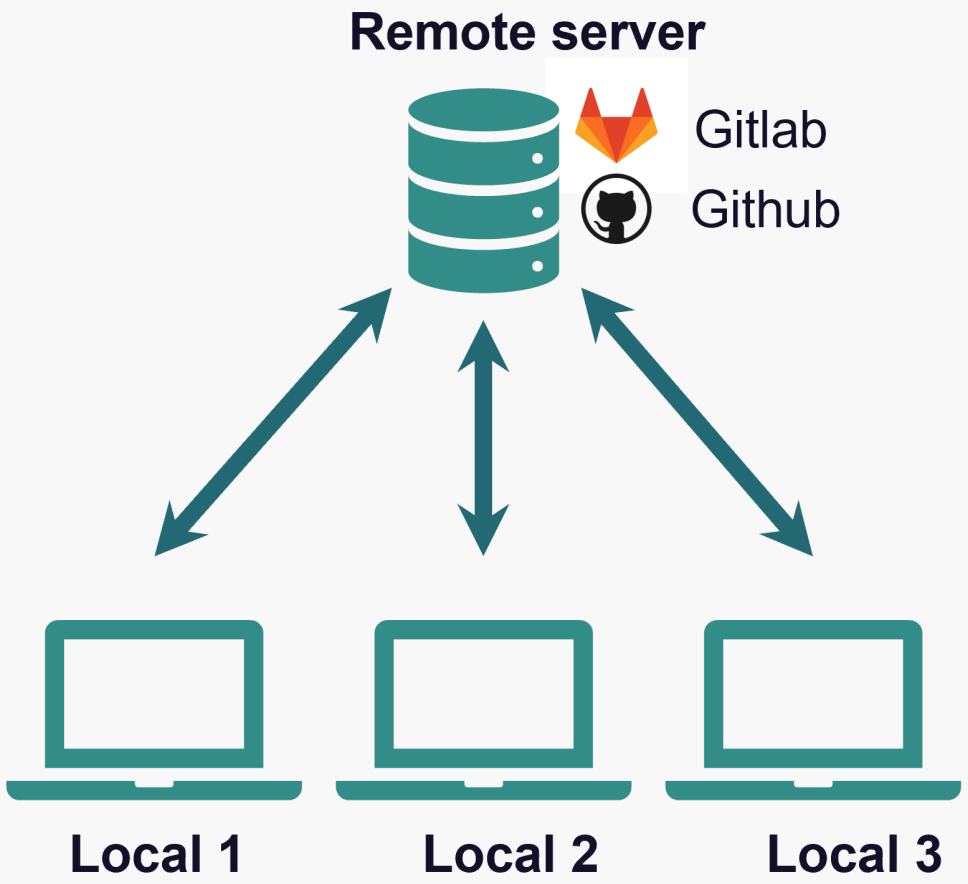
- For projects with **mainly text files** (e.g. code, markdown files, …)
- Basic idea: Take snapshots (**commits**) of your project over time



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

What is 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

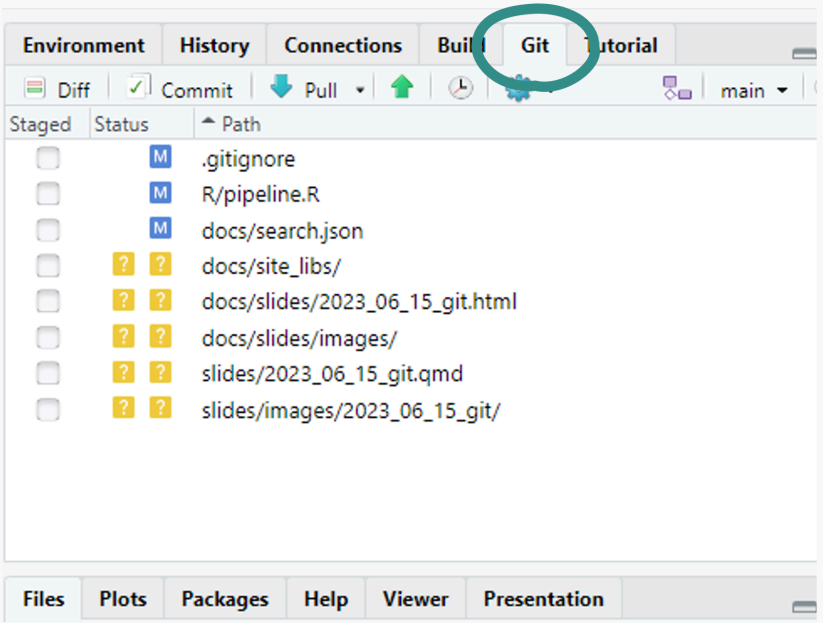
Using Git from the terminal

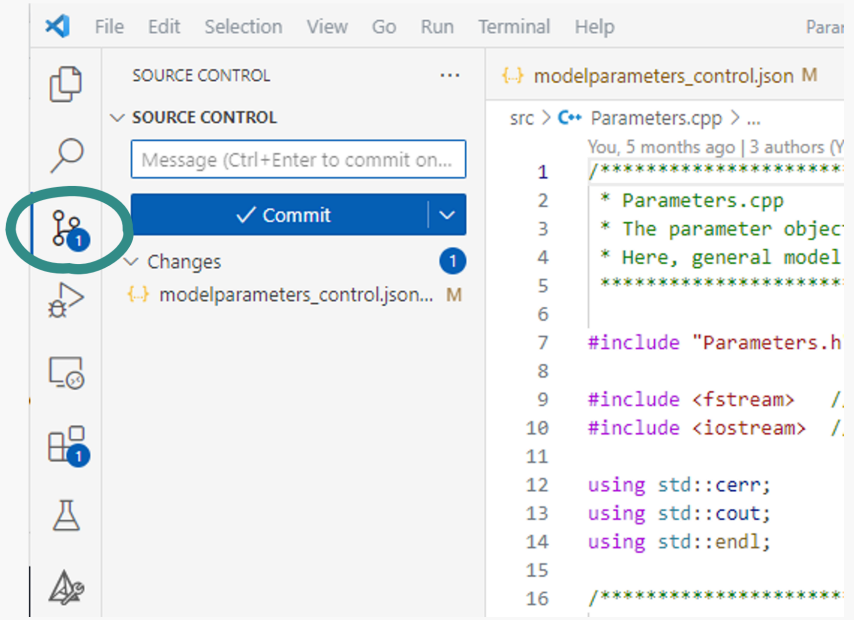
```
Selina_User@DESKTOP-G0RM7MS MINGW64 ~/Files_Selina
$ cd Repos/02_workshops/first_git_project/

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





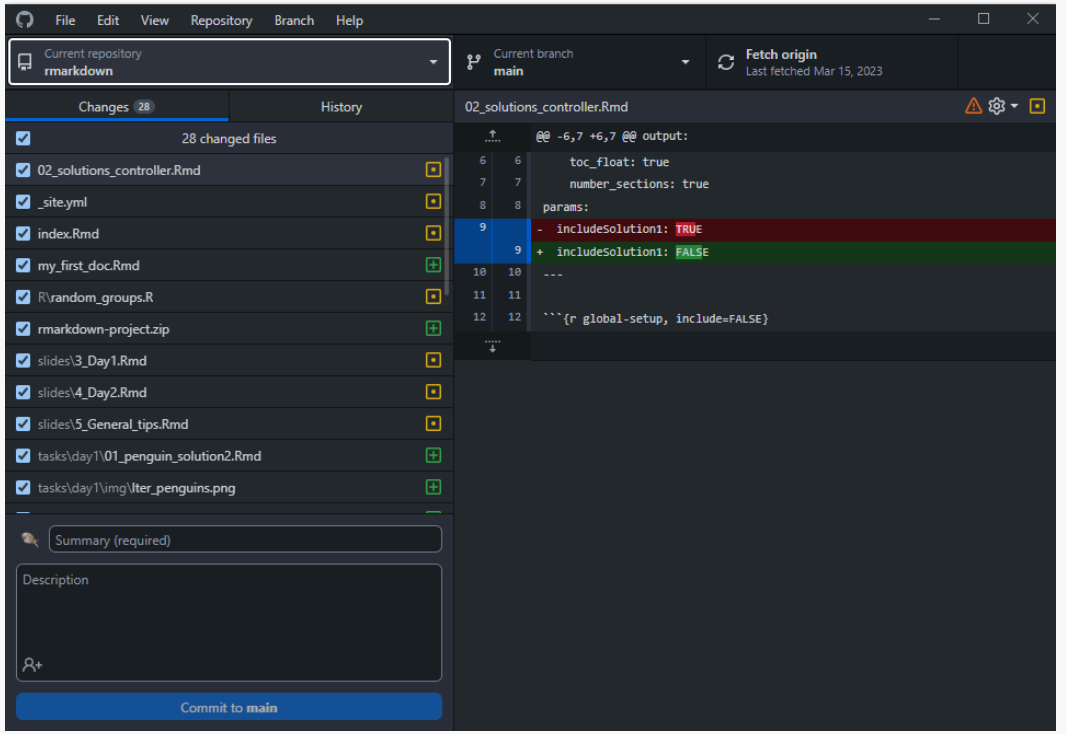
\+ Easy and intuitive

\+ 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?

- Depends on experience and taste
- You can mix methods because they are all interfaces to the same Git
- We will use the GitHub Desktop GUI
  - Beginner-friendly, intuitive and convenient
  - Nice integration with GitHub

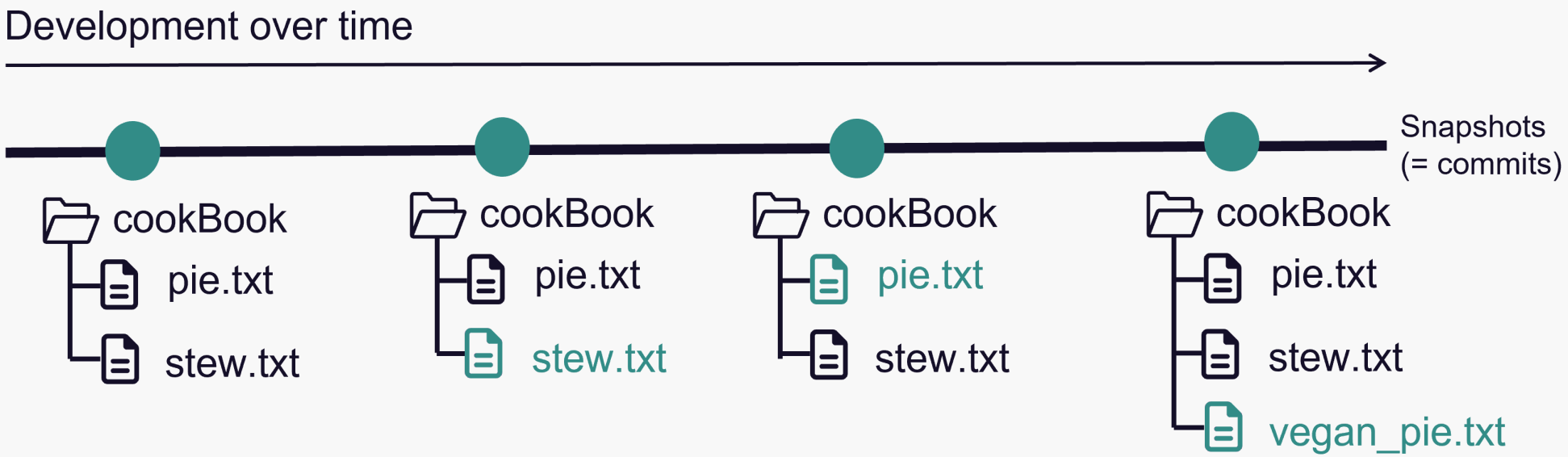
> **Tip**
>
> Have a look at the website where you find **How-To guides for the other methods** as well.

The basic Git workflow

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

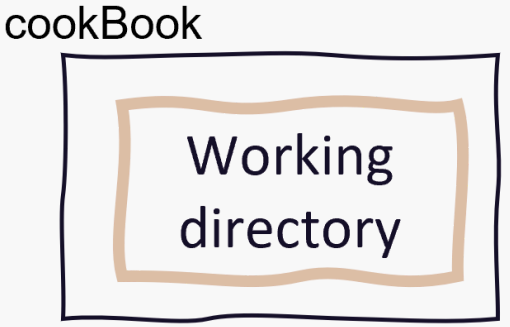
Example

A cook book project to collect all my favorite recipes.

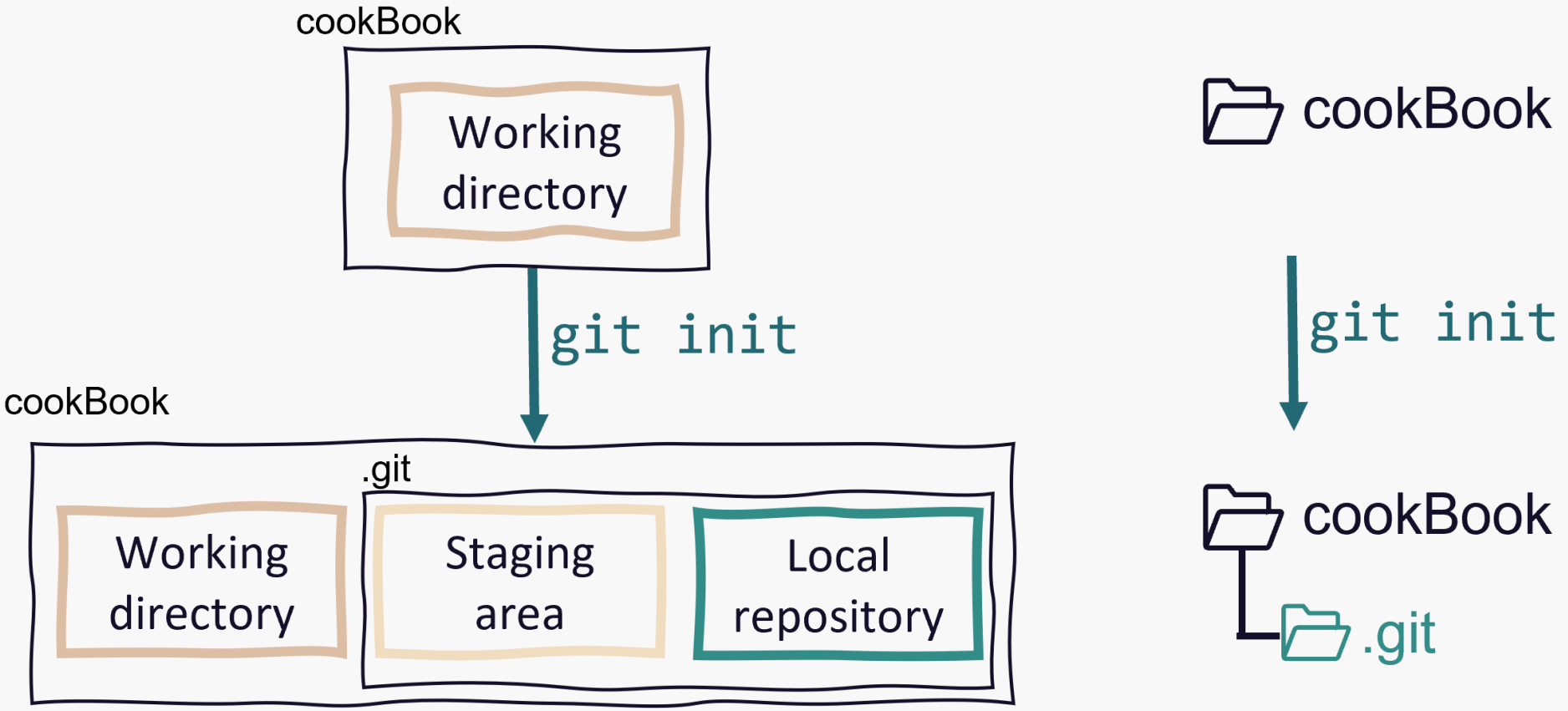


In real life this would be e.g. a data analysis project, your thesis in LaTex, a software project, …

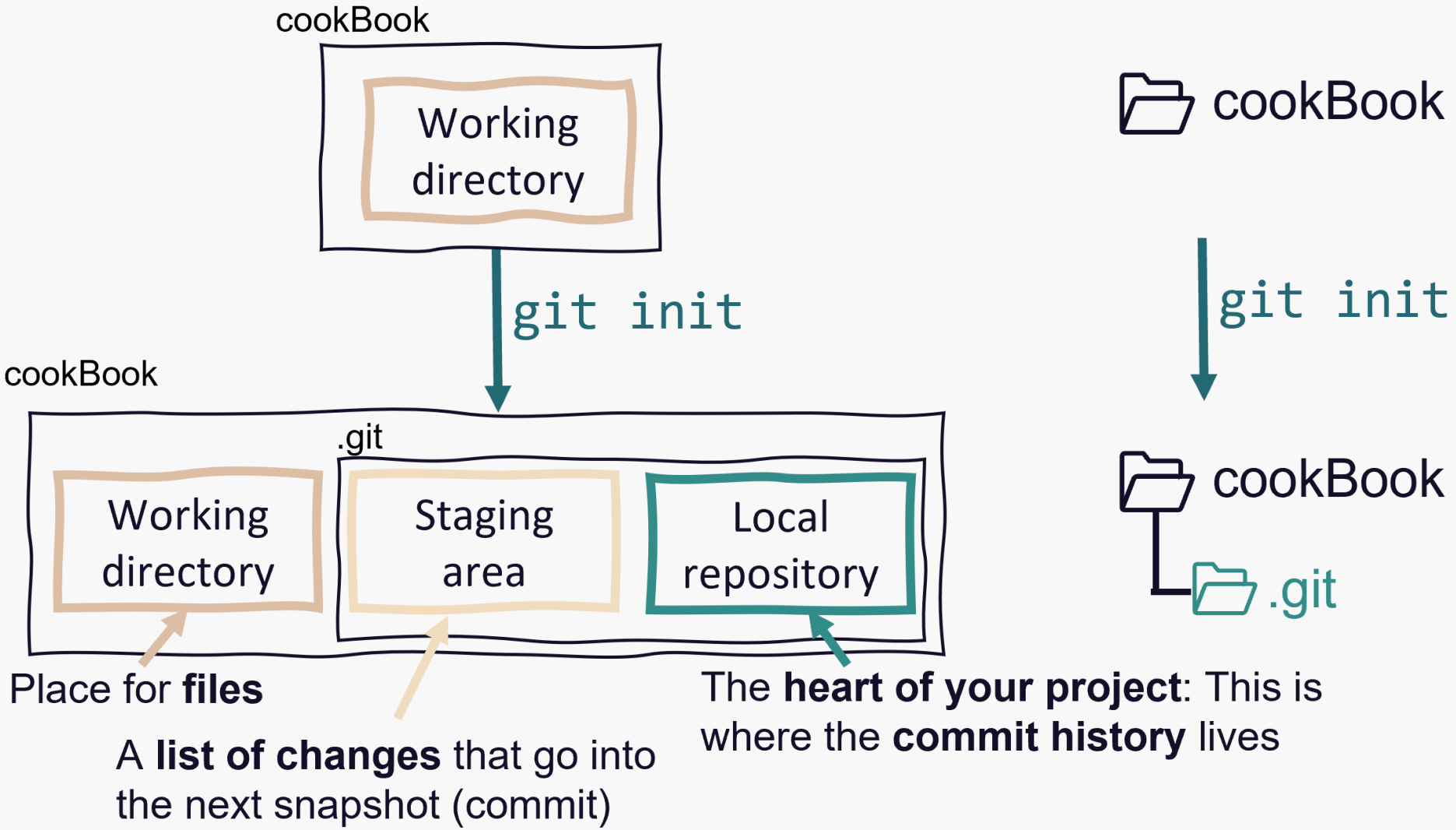
Step 1: Initialize a Git repository



Step 1: Initialize a Git repository

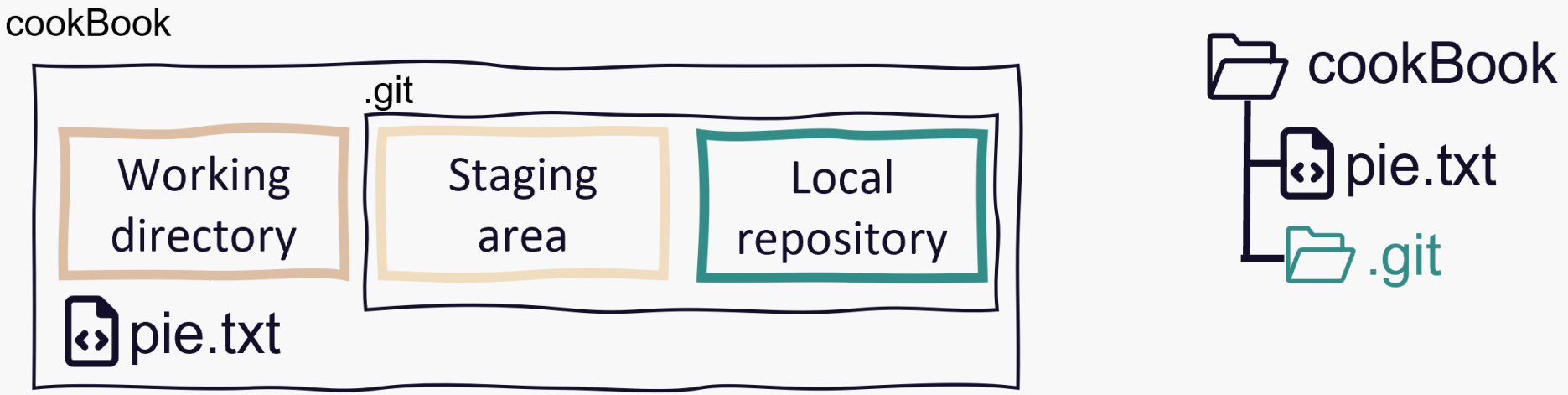


Step 1: Initialize a Git repository



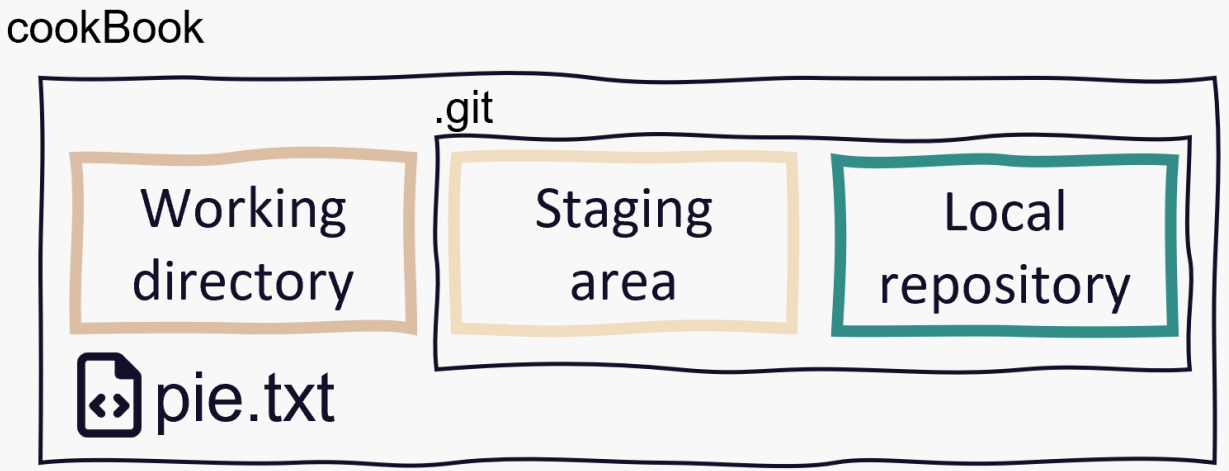
Step 2: Add and modify files

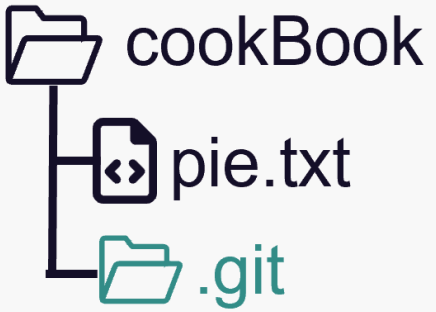
Git detects any changes in the working directory



Step 2: Stage changes

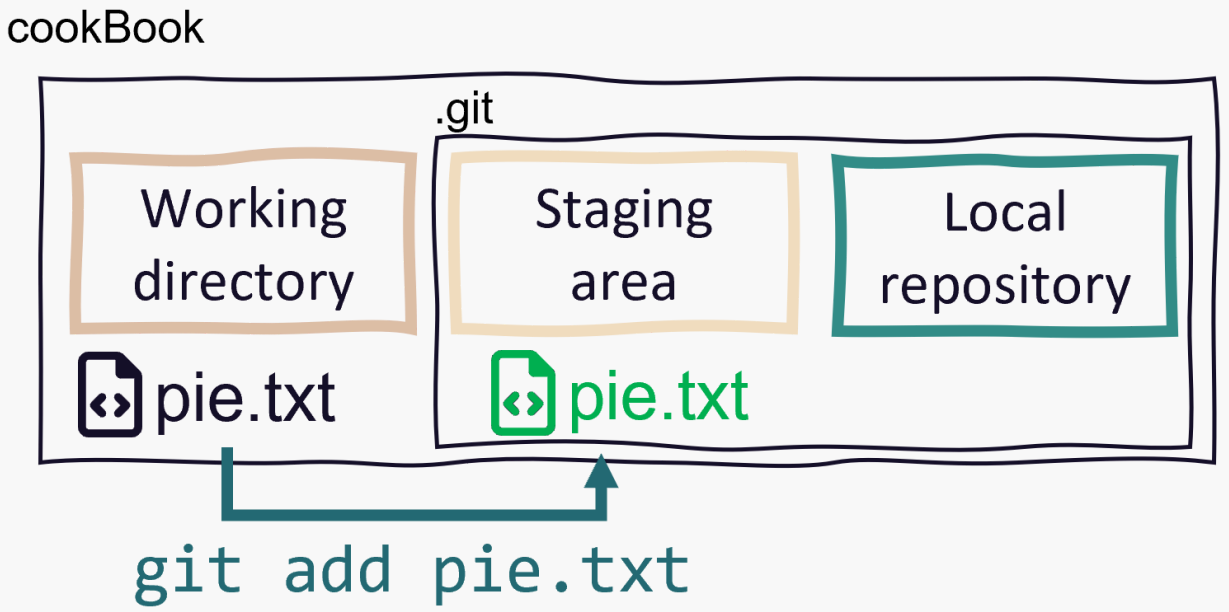
Staging a file means to **list it for the next commit**.

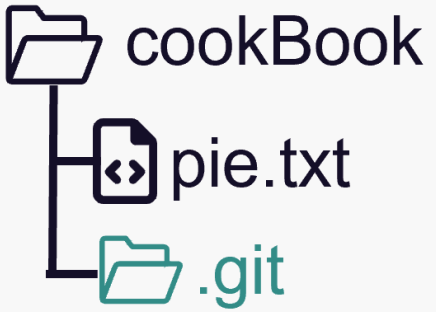




Step 2: Stage changes

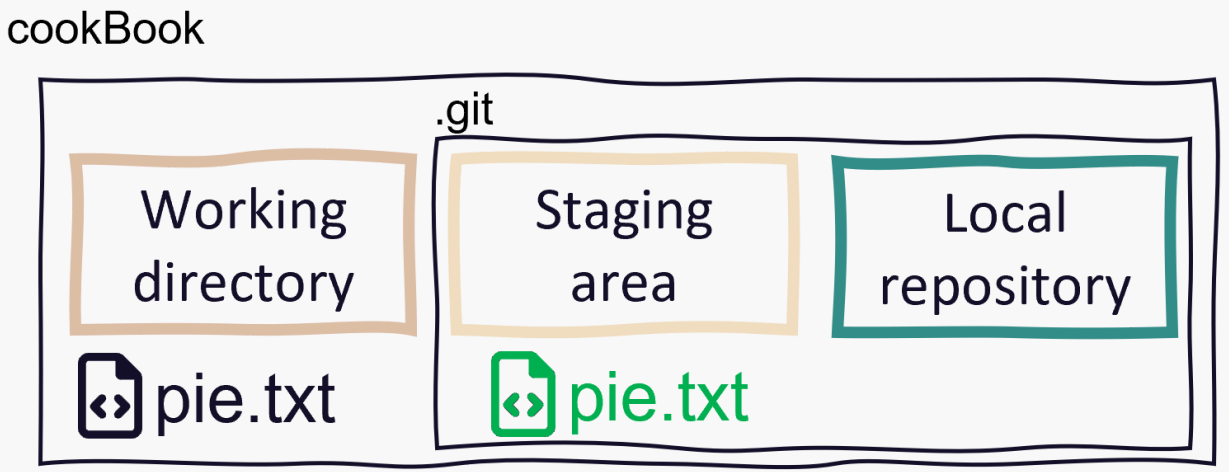
Staging a file means to **list it for the next commit**.

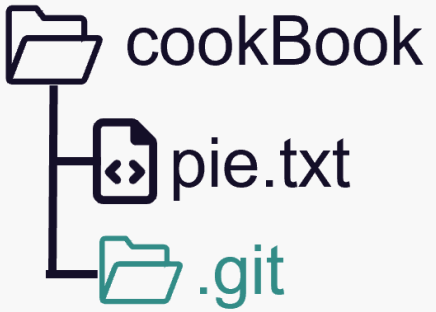




Step 3: Commit changes

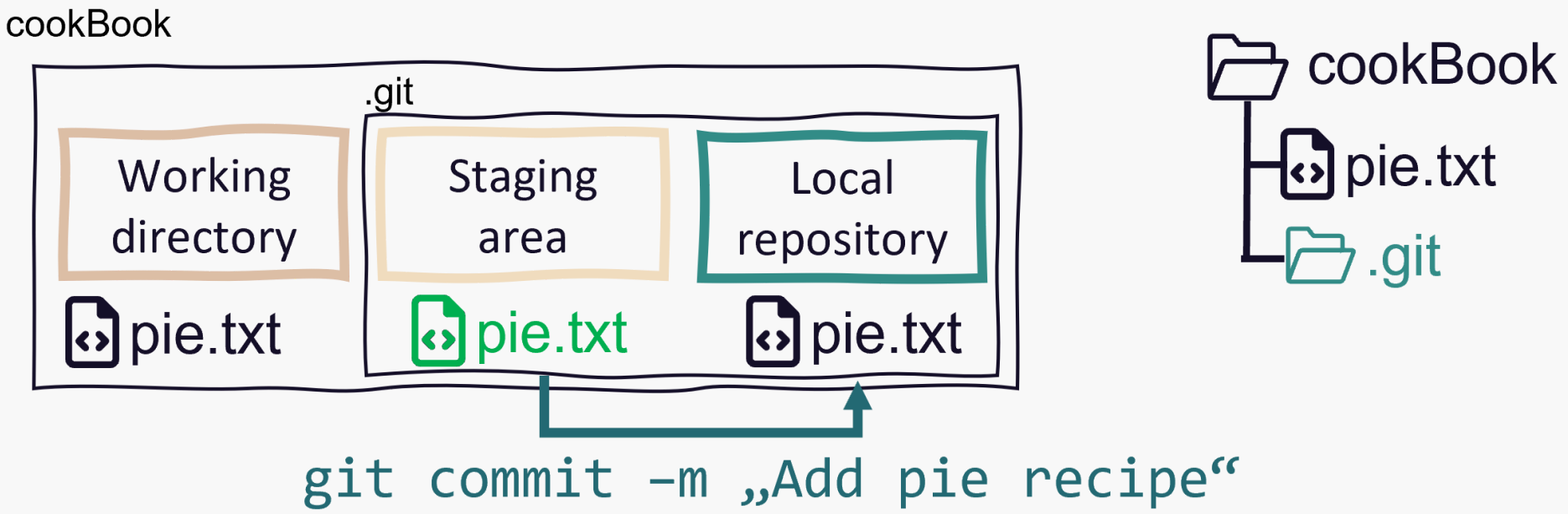
Commits are the snapshots of your project state





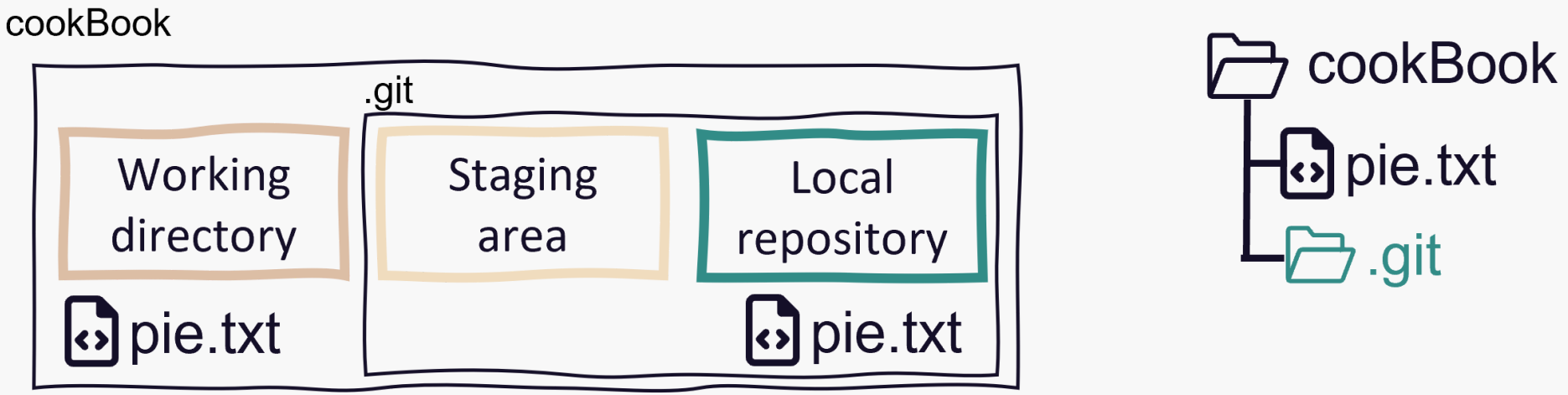
Step 3: Commit changes

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**

The commit history

Add pie recipe This is my favourite pie in the world. The recipe comes from my grandfather and he learned it from his neighbor.	**Add lentil stew recipe**	**Replace milk with soy milk** This is better for the environment and for cows	**Add cooking duration** Alice reminded me that I forgot to add the cooking duration. This was bad because the food got burned. This is now fixed.
+Best pie ever +100 ml milk +2 eggs	+stew.txt	- 100 ml milk +100 ml soy milk	+Bake time: 45 min +Cook time: 30 min
Selina Baldauf <selina.baldauf@fu-berlin.de.de>	Selina Baldauf <selina.baldauf@fu-berlin.de.de>	Bob Example <bob.example@email.com>	Selina Baldauf <selina.baldauf@fu-berlin.de.de>
Wed Jul 19 18:23:15 2023 +0200	Wed Jul 19 20:23:15 2023 +0200	Wed Jul 20 15:23:15 2023 +0200	Tue Aug 20 12:25:15 2023 +0200

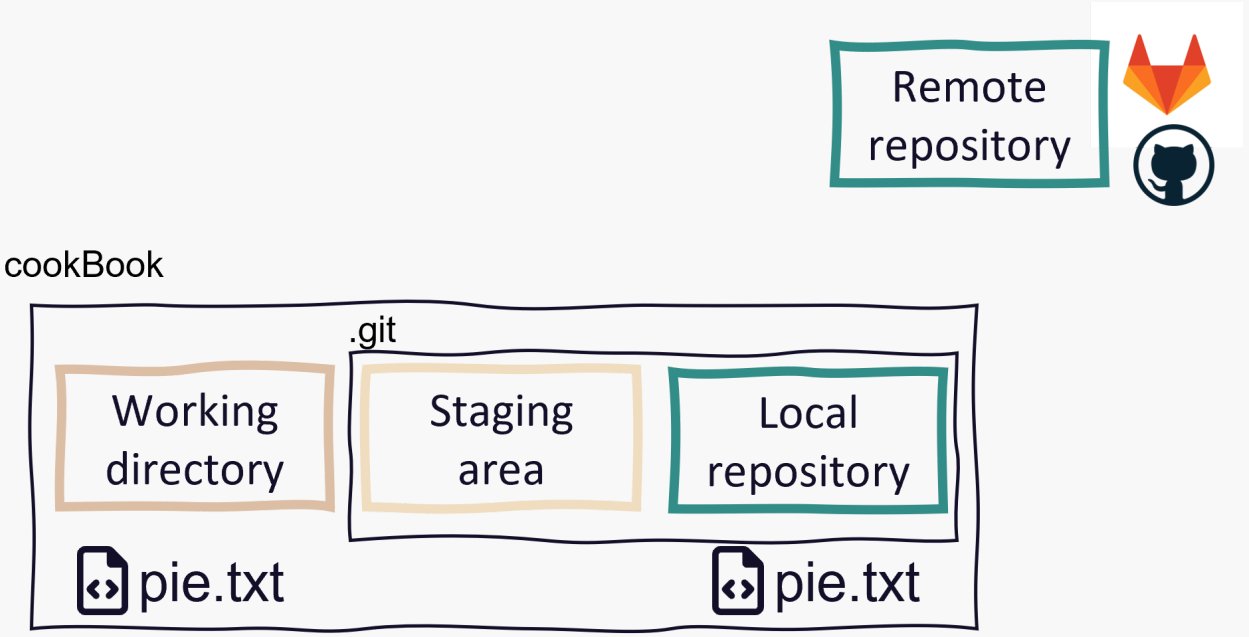
Now you (10 min)

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

> Start your own cook book
> Complete Task 1 “Local repo”

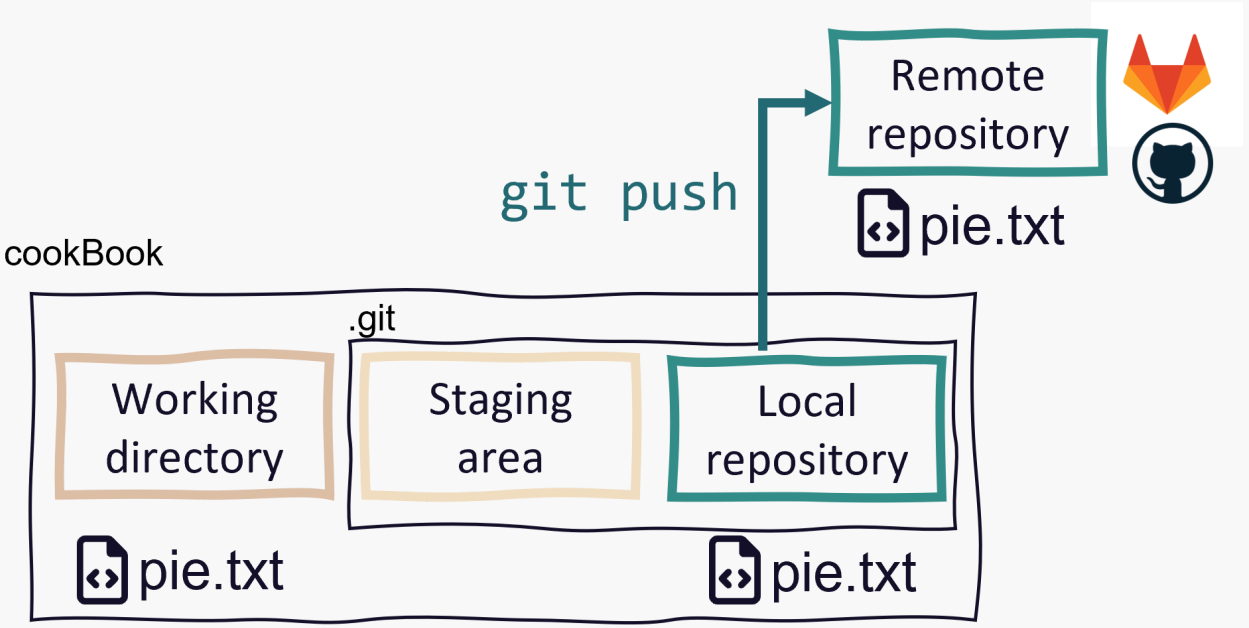
Step 4: Share changes with the remote repo

Use remote repos to **synchronize**, **share** and **collaborate** (can be public or private/for collaborators only)



Step 4: Share changes with the remote repo

Use remote repos to **synchronize**, **share** and **collaborate** (can be public or private/for collaborators only)



Publishing your projects

- 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 might be outside EU (e.g. GitHub)
  - Privacy rules might apply depending on type of data

Advantages of publishing your projects

- Visibility and Credit
- Portfolio of your work
- Others can build on your work
- Get feedback
- Citations
- Reproducibility

Now you (5 min)

> Publish your cook book on GitHub
> Complete Task 2 “GitHub”

Preparation for tomorrow

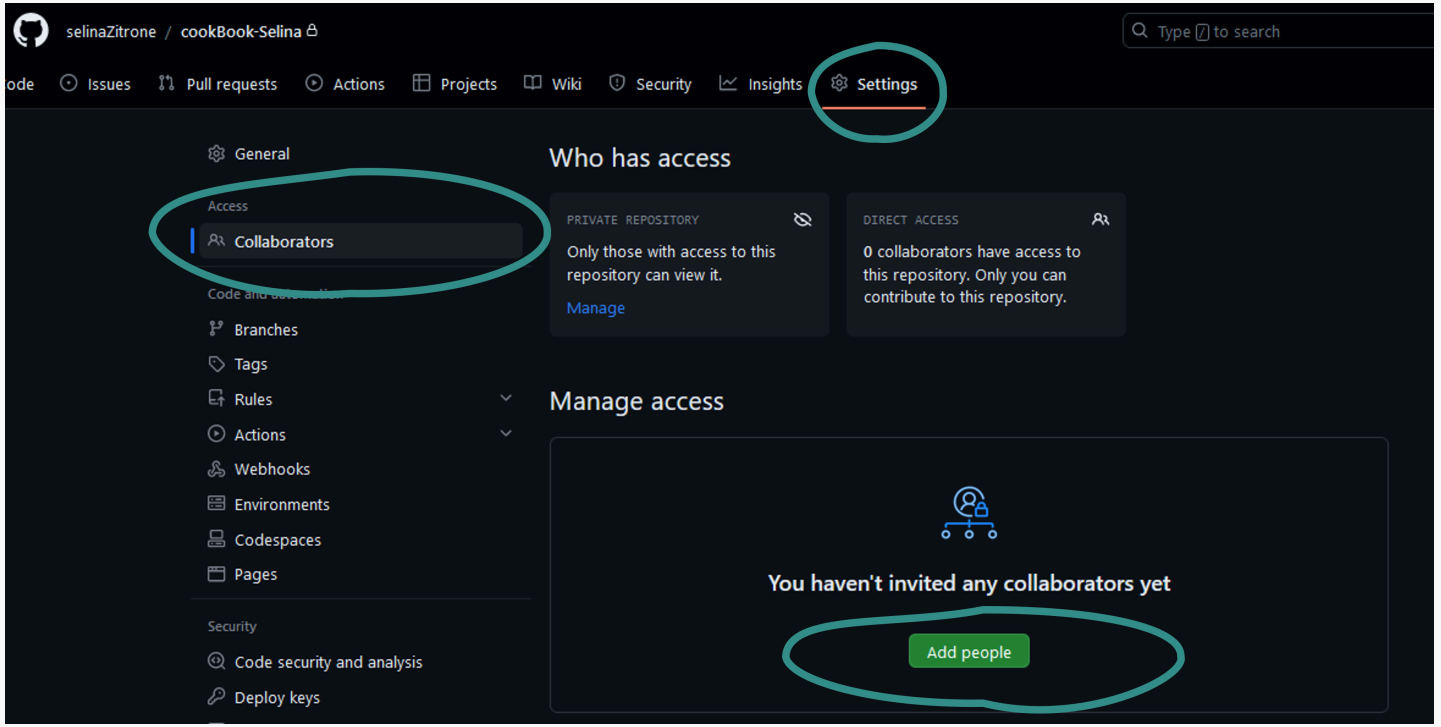
- Tomorrow we have teams of 2
- Collaborate on the cook book of your team mate

Preparation for tomorrow

1. Enter your GitHub Account Name and the link to your repo here

Preparation for tomorrow

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



Preparation for tomorrow

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

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`, `git restore`

Restore/Discard uncommitted changes

- You can easily discard uncommitted changes in your working directory
  - In the terminal: `git restore <file>`
  - In GUIs: right-click the file and select **Discard changes**

Revert committed 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 committed changes

Now you (5 min)

> Undo some changes in your cook book

Ignore files with `.gitignore`

Ignore files with `.gitignore`

- Git tracks all files in your working directory
- Often, we have files we do not want tracked
  - Personal notes
  - 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

```
# Ignore single files

my_notes.docx  # ignore the file my_notes.docx
debug.log # ignore the file debug.log


# Ignore files with specific endings

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

# Ignore directories

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

See here for more ignore patterns that you can use.

Now you (5 min)

> Ignore some files in your cook book project

Some good practice tips

Git

- Commit often (small changes that can be described in one commit message)
- Write good commit messages (it becomes a habit)
- Push (at least) daily (backup!)
- Use `.gitignore`
- Don't commit secrets

Publishing

Some essentials that will improve your published repository:

- Add a good README.md file
  - Tell people what your project is about and how to use it
  - Check out the GitHub documentation for formatting options
- Add a LICENSE file
  - Tell others how to use your code
- Add a DOI to your repository (e.g. via Zenodo)

If you are interested, browse some nice GitHub repositories for inspiration (e.g. Computational notebooks guide)

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`