

Introduction to version control with Git

Day 1: Concepts and a basic workflow

Selina Baldauf

September 17, 2023

Who am I?

Scientific programmer @ theoretical ecology group

Who are you?

Aims of the workshop

Git is a huge topic and Git is very powerful.

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

Enough time to for practice and questions.

Topics

Today

- Introduction to Git concepts
- Simple Git workflow for your own projects

Tomorrow

- Collaborate using Git and Github

Next Monday

- Q&A
- Advanced topics
- Until then: work with Git on your own projects

Organization

- Today and tomorrow:  2.30 - 4 p.m.
- Next Monday:  2.30 - 3.30 p.m.
- Material is all [online](#)

Before we start

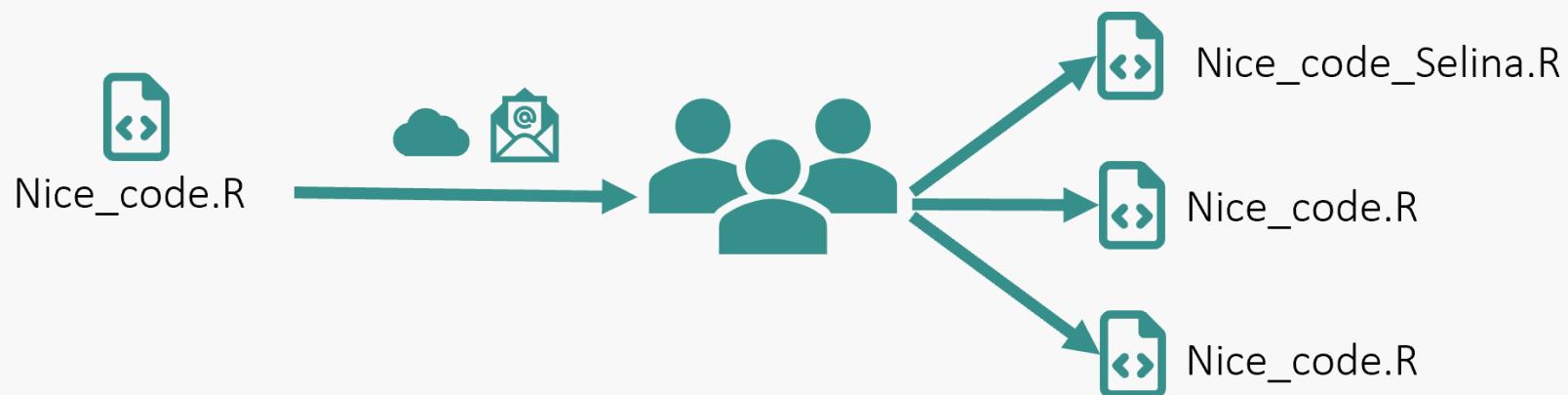
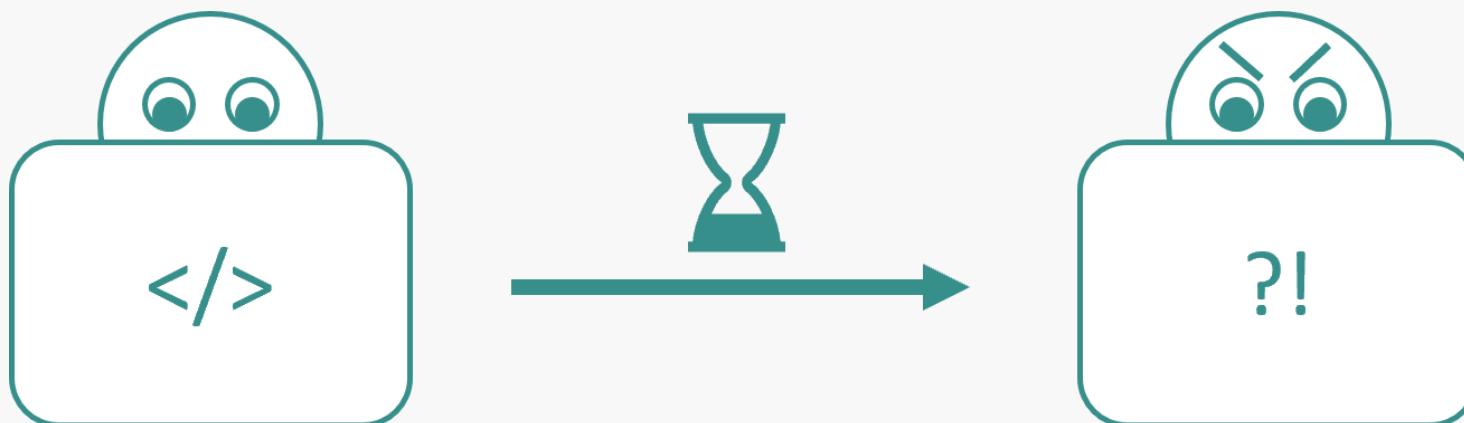
Did anyone have problems with the workshop preparation?

- 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



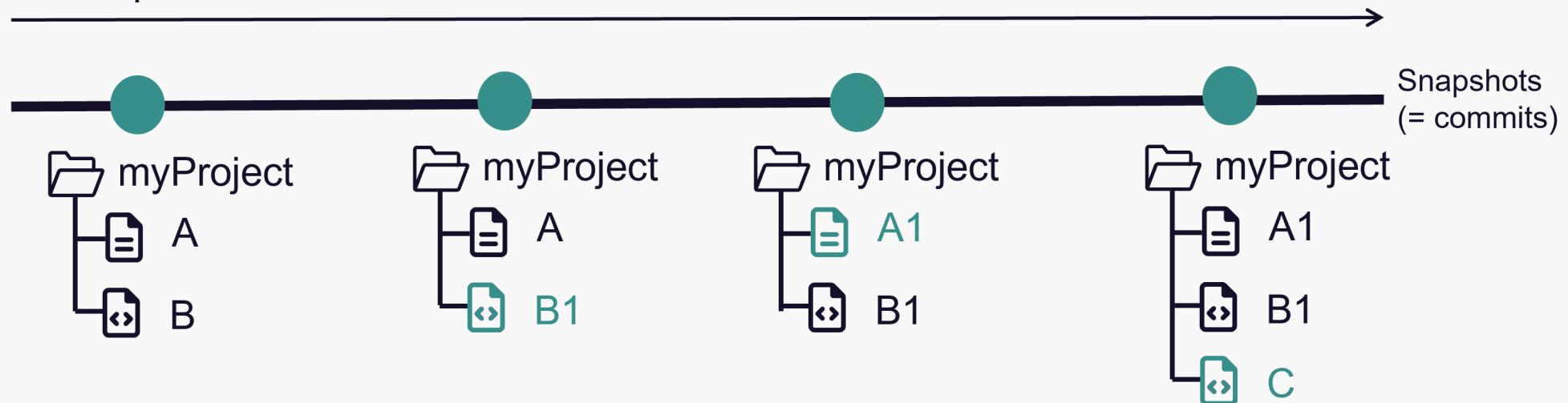
Version control with Git

- 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

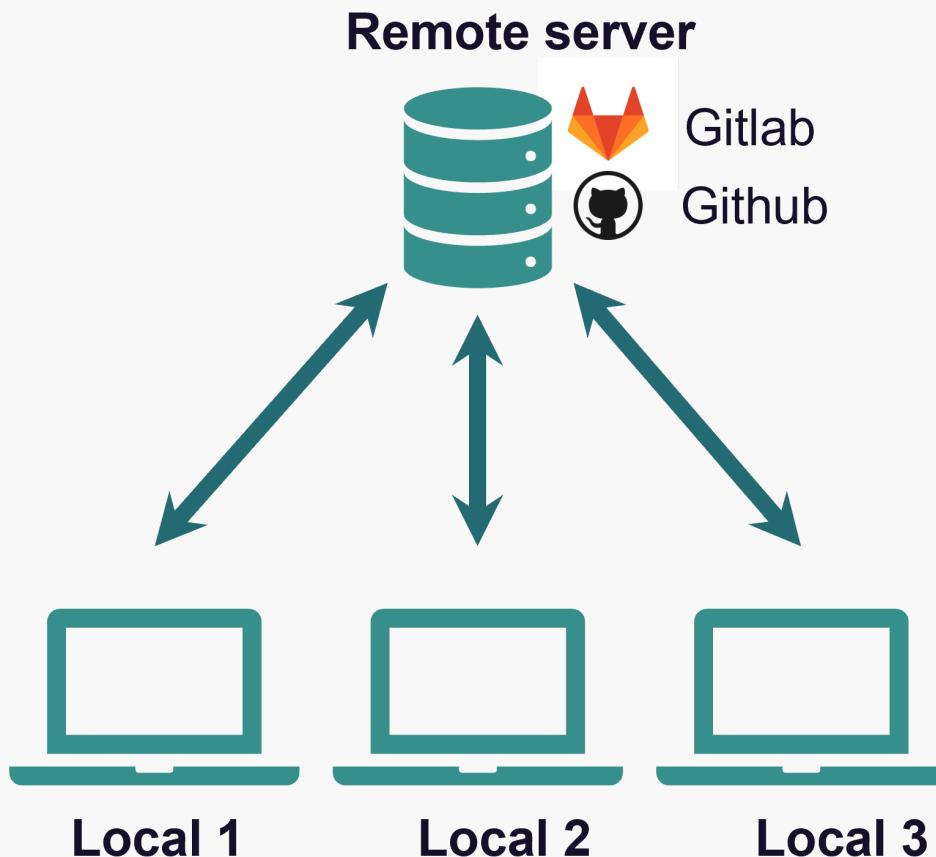
- For projects with mainly text files (e.g. code, markdown files, ...)
- Basic idea: Take snapshots (**commits**) of your project over time
- A project version controlled with Git is a Git **repository** (**repo**)

Development over time



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/

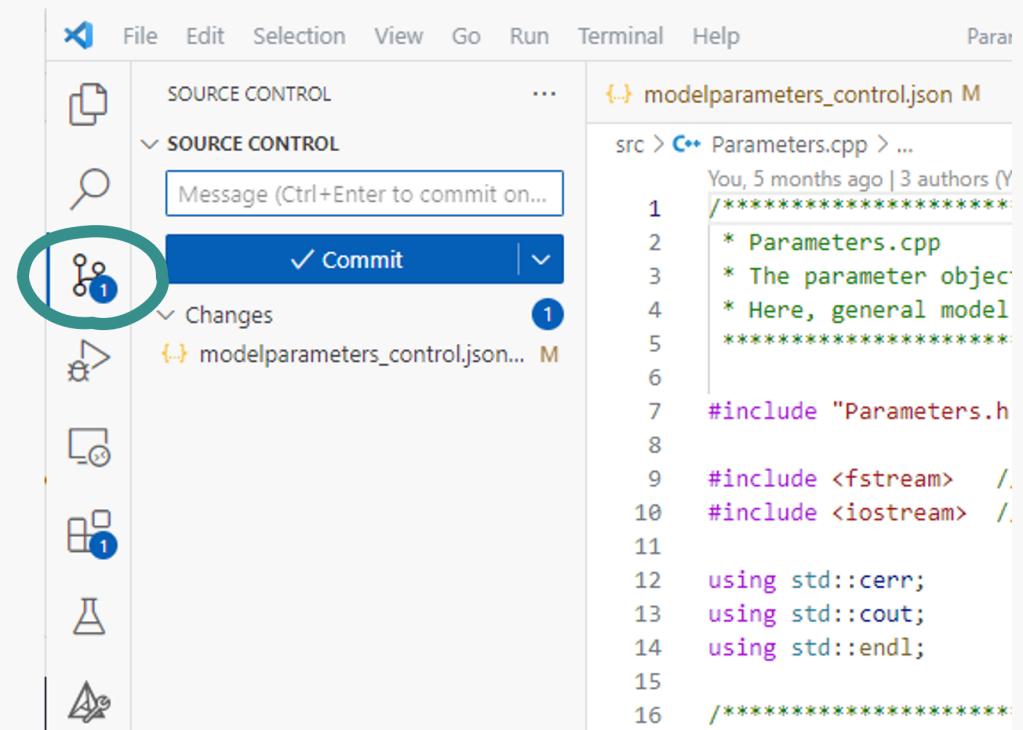
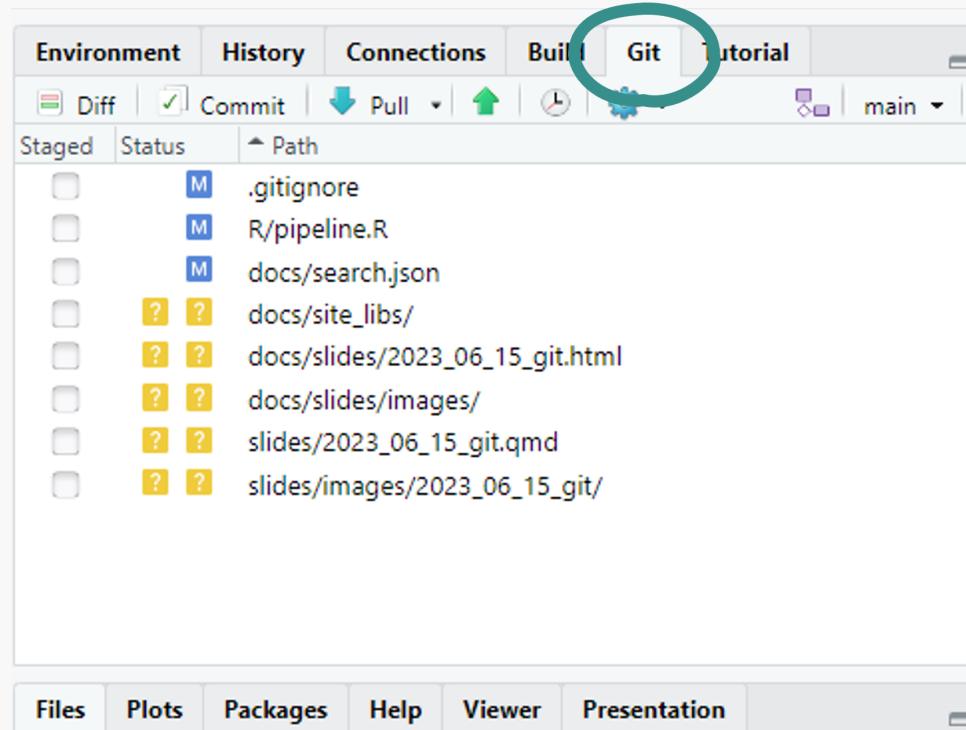
selina_user@DESKTOP-G0RM7MS MINGW64 ~/Files_Selina/Repos/02_workshops/first_git_
project
$ git init
Initialized empty Git repository in C:/Users/selina_user/Files_Selina/Repos/02_w
orkshops/first_git_project/.git/
selina_user@DESKTOP-G0RM7MS MINGW64 ~/Files_Selina/Repos/02_workshops/first_git_
project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

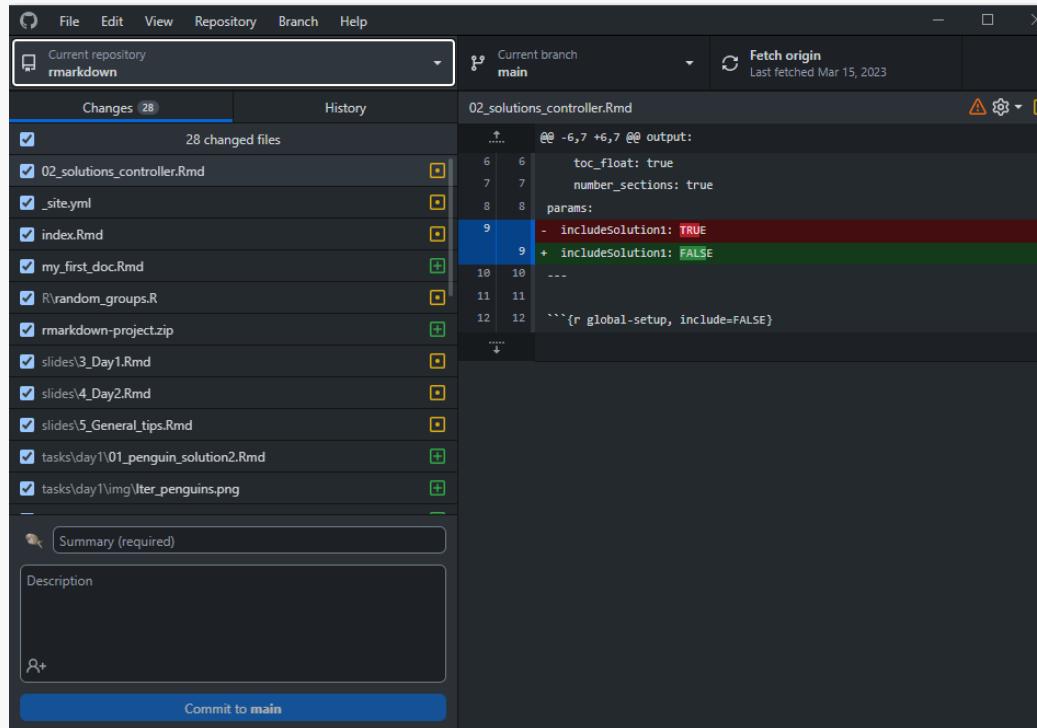


- + Easy and intuitive
- + Stay inside IDE

- Different for every program

How to use Git - Standalone GUIs

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



+ Easy and intuitive

+ Nice integration with Github

- 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 and intuitive
 - Very 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

Step 0: An empty project

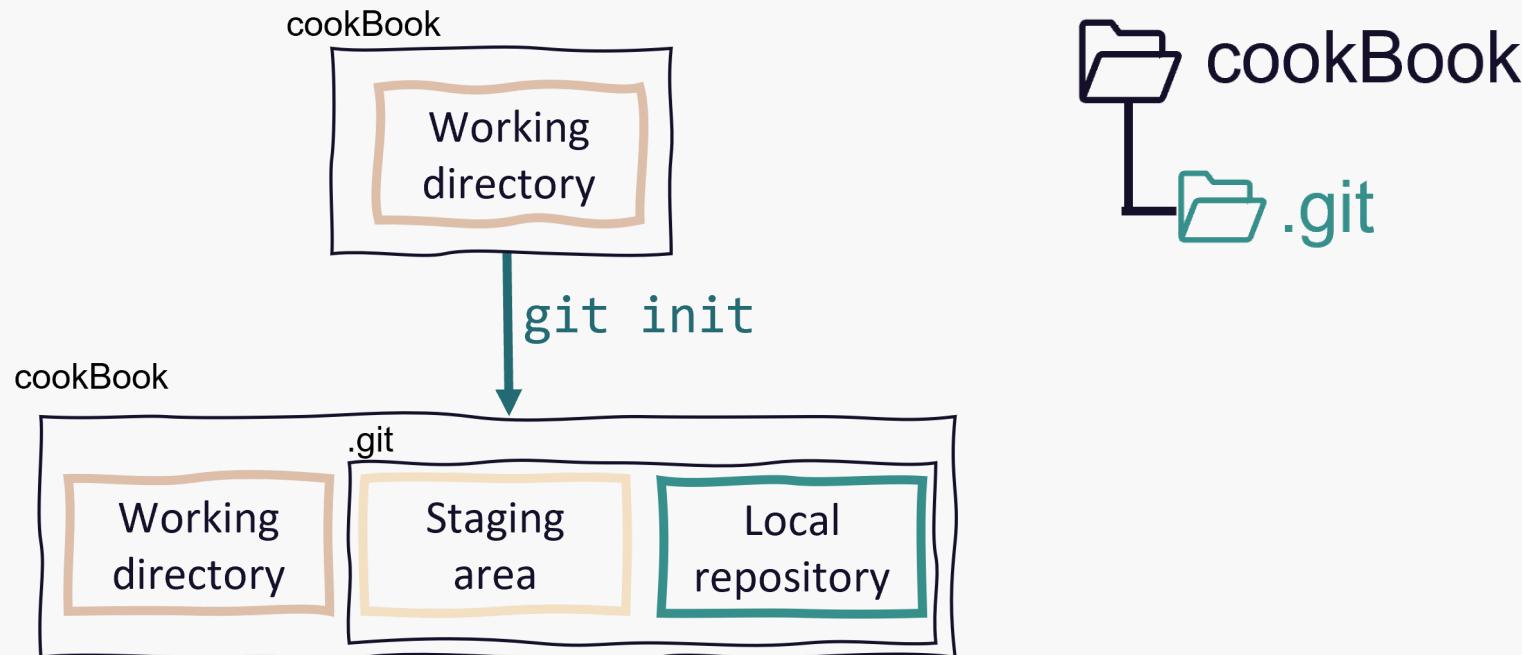
Example: A cookBook project to collect all our favorite recipes

cookBook



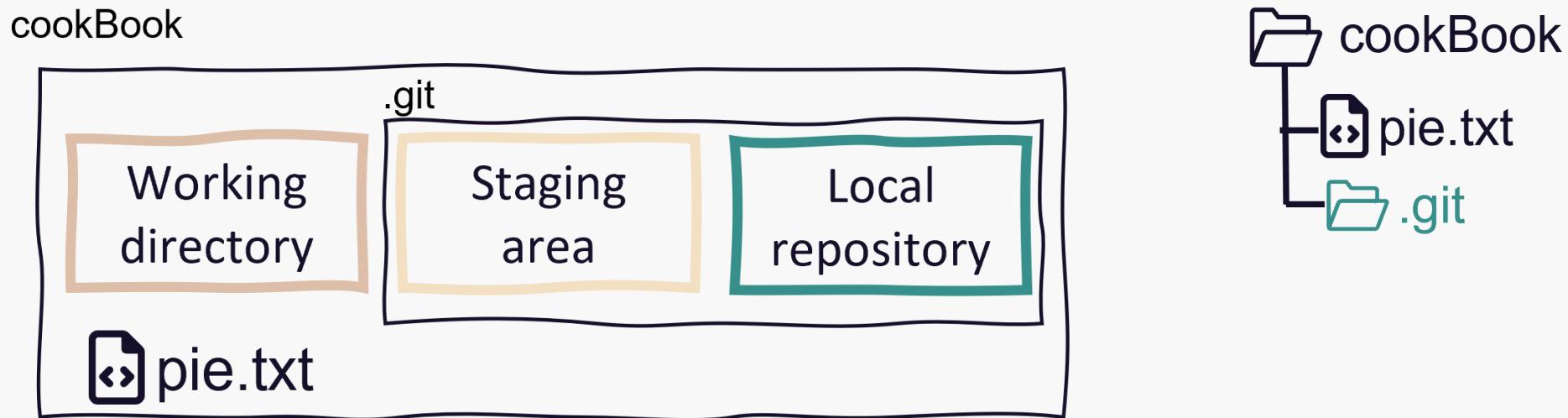
Step 1: Initialize a Git repository

- Adds a (hidden) `.git` folder to your project
 - You don't need to interact with this folder directly



Step 2: Modify files and stage changes

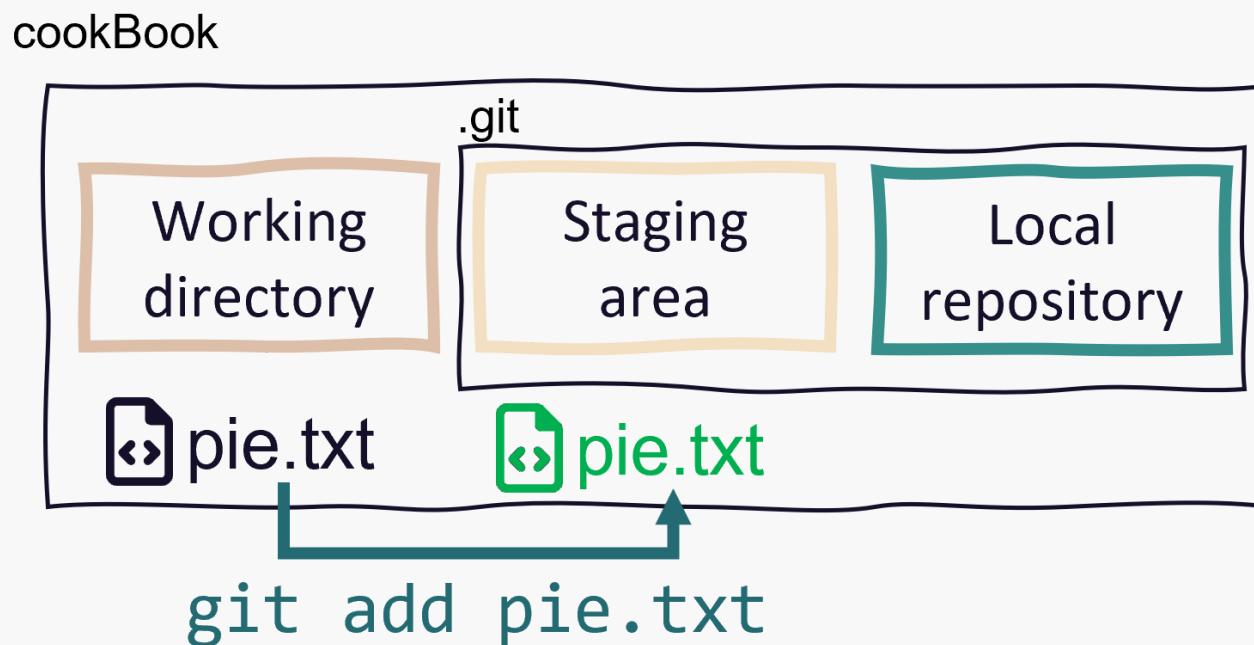
Git detects any changes in the working directory



Step 2: Modify files and stage changes

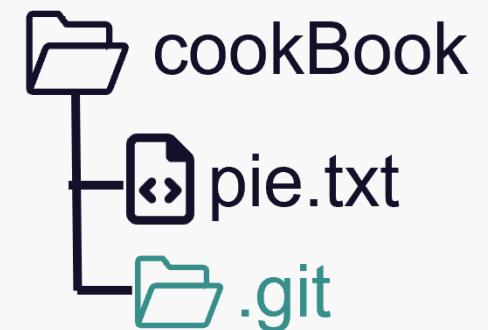
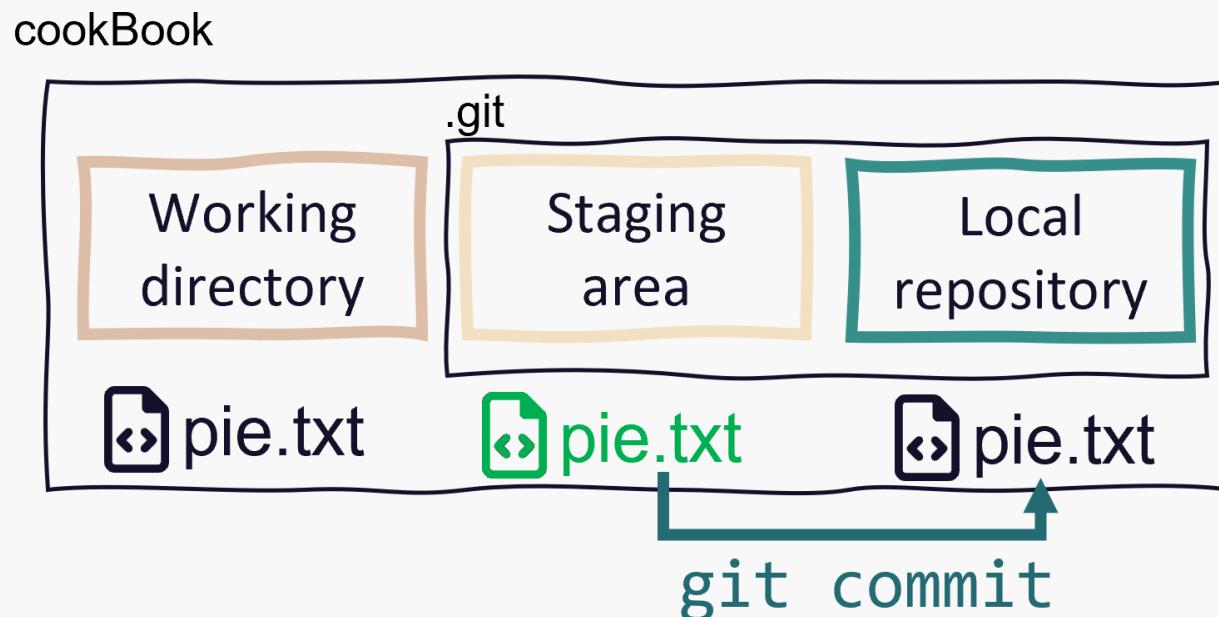
Stage file to be part of the next commit (snapshot)

- In the terminal use `git add`
- In GUIs just a check box



Step 3: Commit changes

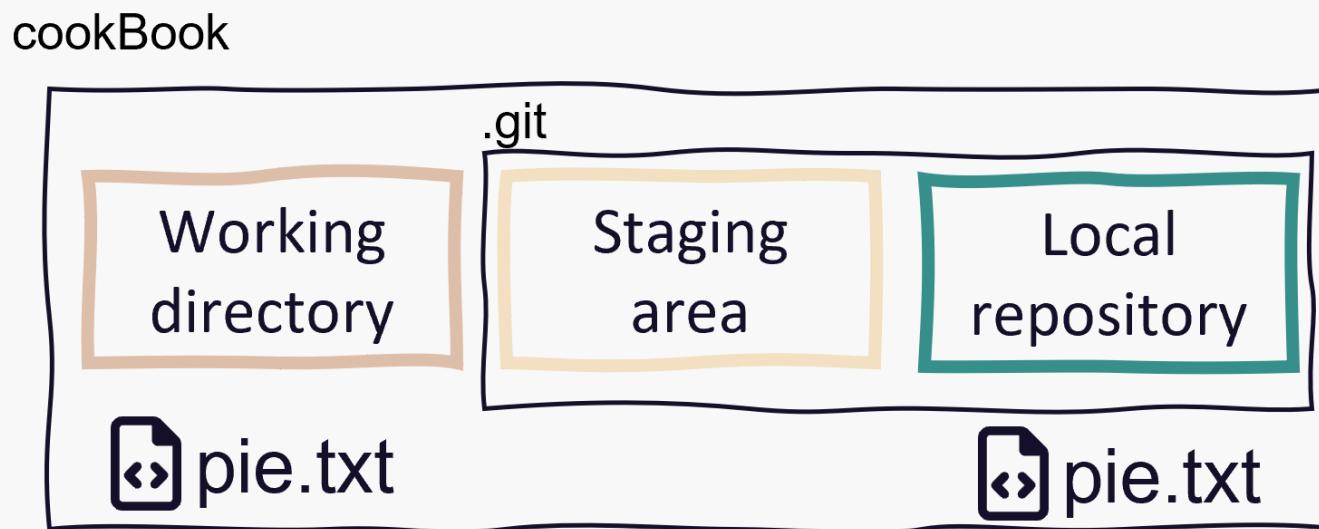
- Commits are the snapshots of your project state



- Collect meaningful bundles of changes in the staging area, then commit

Step 3: Commit changes

- After a commit, the staging area is clear again
- Changes are now part of the project's Git history



Now you

Start your own cook book Complete task 1 “Local repo” (10 min)

Stay in the meeting for the task.

Ask questions if you are stuck.

Turn down/off volume if you are disturbed.

How to write good commit messages?

	COMMENT	DATE
O	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
O	ENABLED CONFIG FILE PARSING	9 HOURS AGO
O	MISC BUGFIXES	5 HOURS AGO
O	CODE ADDITIONS/EDITS	4 HOURS AGO
O	MORE CODE	4 HOURS AGO
O	HERE HAVE CODE	4 HOURS AGO
O	AAAAAAA	3 HOURS AGO
O	ADKFJSLKDFJSOKLFJ	3 HOURS AGO
O	MY HANDS ARE TYPING WORDS	2 HOURS AGO
O	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

xkcd on commit messages

How to write good commit messages?

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**

How to write good commit messages?



Add pie recipe

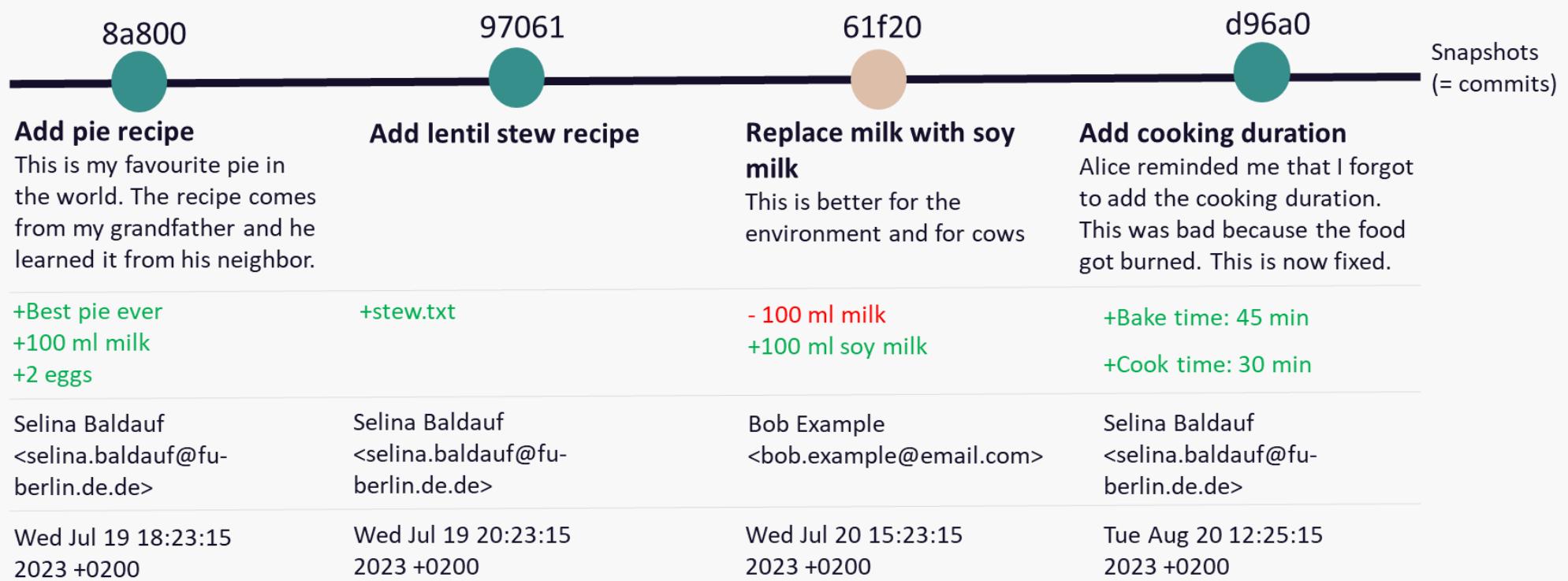
This is my favorite pie in the world.
The recipe comes from my grandfather and
he learned it from his neighbor.



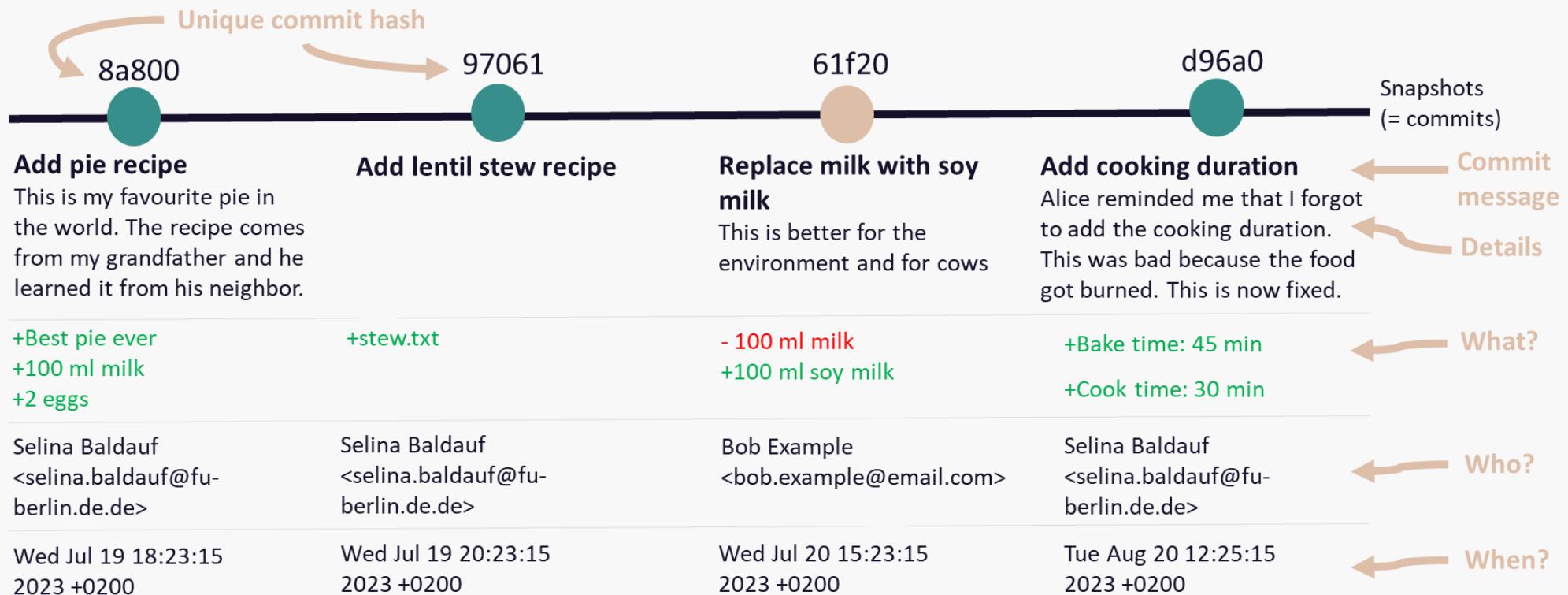
added a file.

This is a a really good recipe that I li

The commit history

				Snapshots (= commits)
8a800	97061	61f20	d96a0	
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	

The commit history



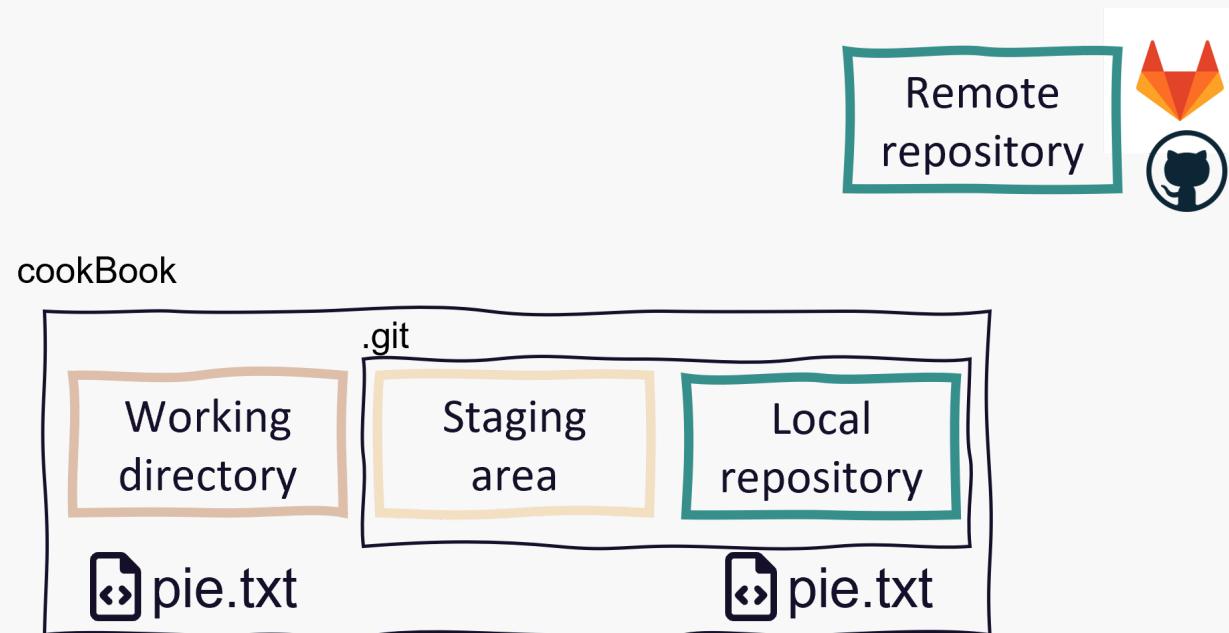
The commit history

A screenshot of a Git commit history interface. At the top, there's a navigation bar with File, Edit, View, Repository, Branch, and Help. Below that, it shows the current repository as 'notebook_demo' and the current branch as 'main'. A green oval highlights the 'History' tab in the top navigation bar. The main area displays a commit titled 'Fix typo' by Selina Baldauf on Aug 30, 2023. The commit details show a file named '05_Ze...\\Modelling cycle.md' with a diff view. The diff shows changes from line 12 to 18. Lines 12 and 13 are unchanged. Line 14 is unchanged. Line 15 has a red background and contains the text '-The modelling cycle describes the stages of model development.' followed by a red 'dd'. Line 16 is unchanged. Line 17 has a green background and contains the text '+The modelling cycle describes the stages of model development.'. Line 18 is unchanged. The commit message also includes '@@ -12,7 +12,7 @@ tags: idea'.

Line	Line	Change	Text
12	12	>	[[Good modelling practice]]
13	13	>	[[TRACE protocol]]
14	14		
15	15	-	The modelling cycle describes the stages of model development. dd
16	16		
17	17	+	The modelling cycle describes the stages of model development.
18	18		

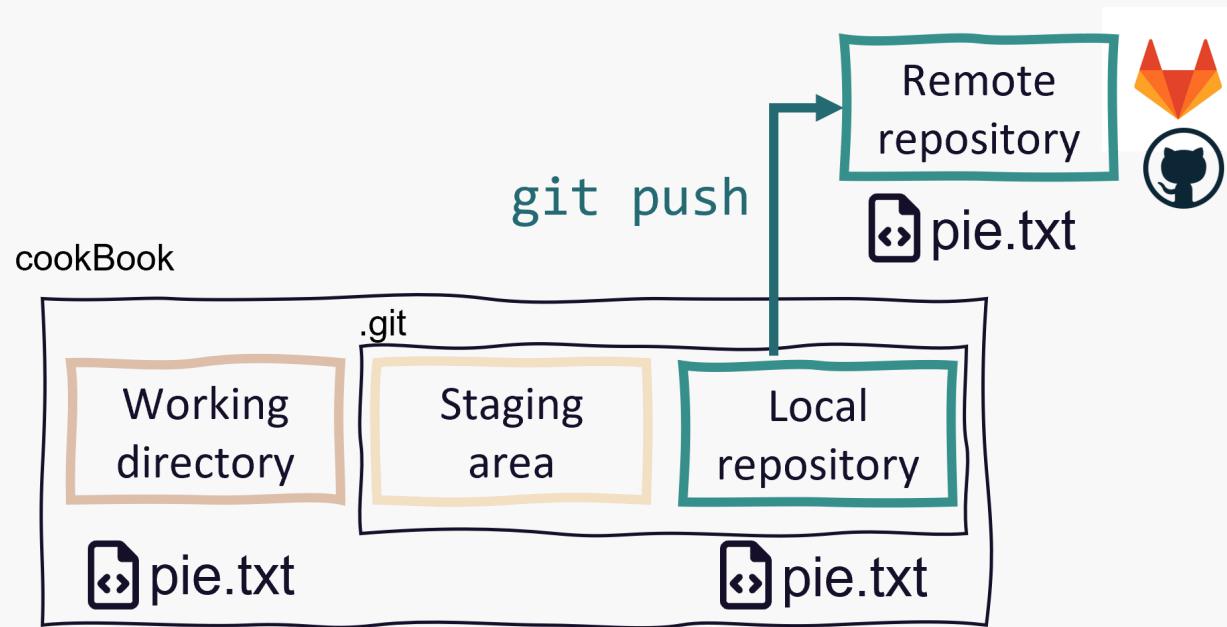
Step 4: Create and connect a remote repo

- Use remote repos (on a server) to *synchronize, share and collaborate*
- Remote repos can be *private* (you + collaborators) or *public* (visible to anyone)



Step 5: Share changes with the remote repo

- Push your local changes to the remote with `git push`



Now you

| Publish your cook book on Github Complete task 2 “Github” (5 min)

A word on 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

Summary of the basic steps

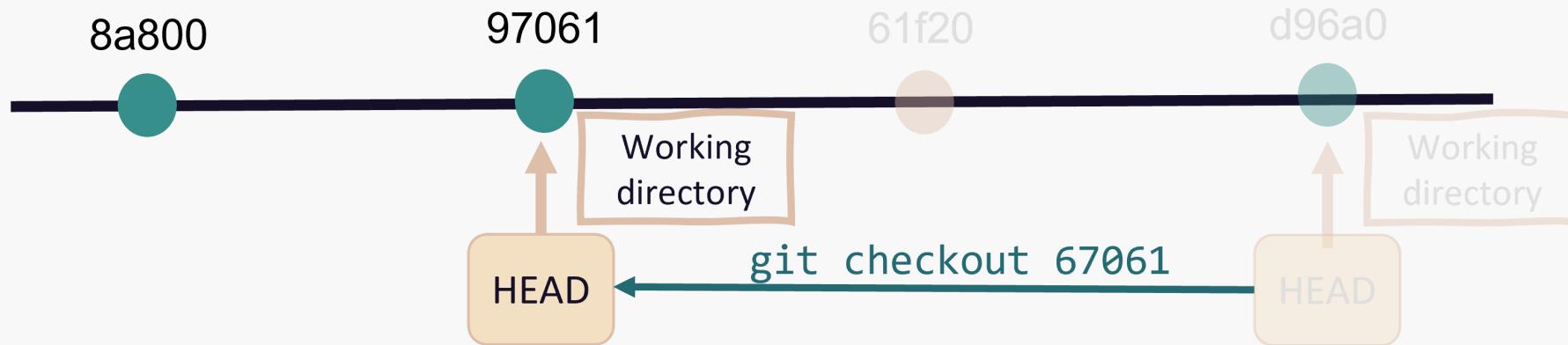
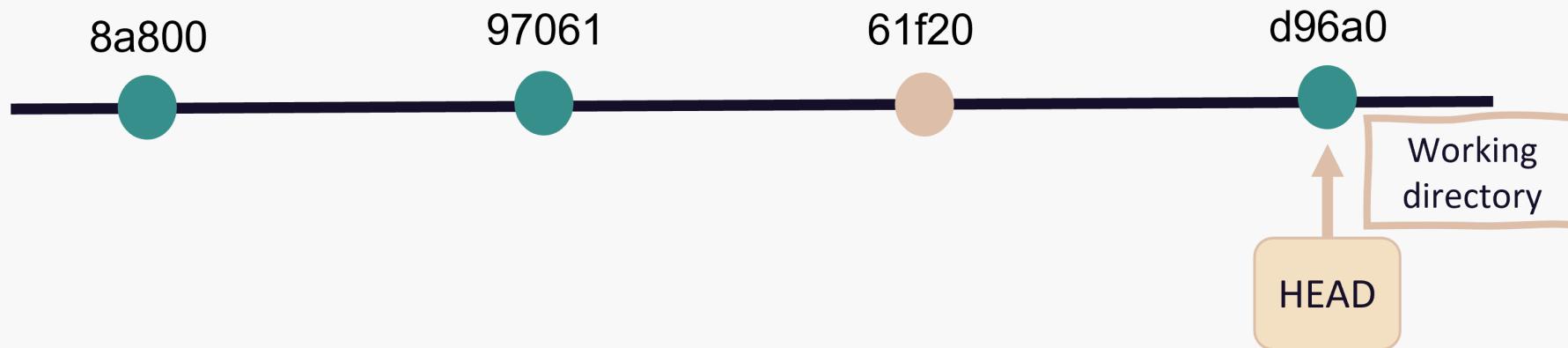
- `git init`: Initialize a git repository
 - Adds a `.git` folder to your working directory
- `git add`: Add files to the staging area
 - This marks the files as being part of the next commit
- `git commit`: Take a snapshot of your current project version
 - Includes time stamp, commit message and information 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

Go back in time with Git

git checkout, git revert

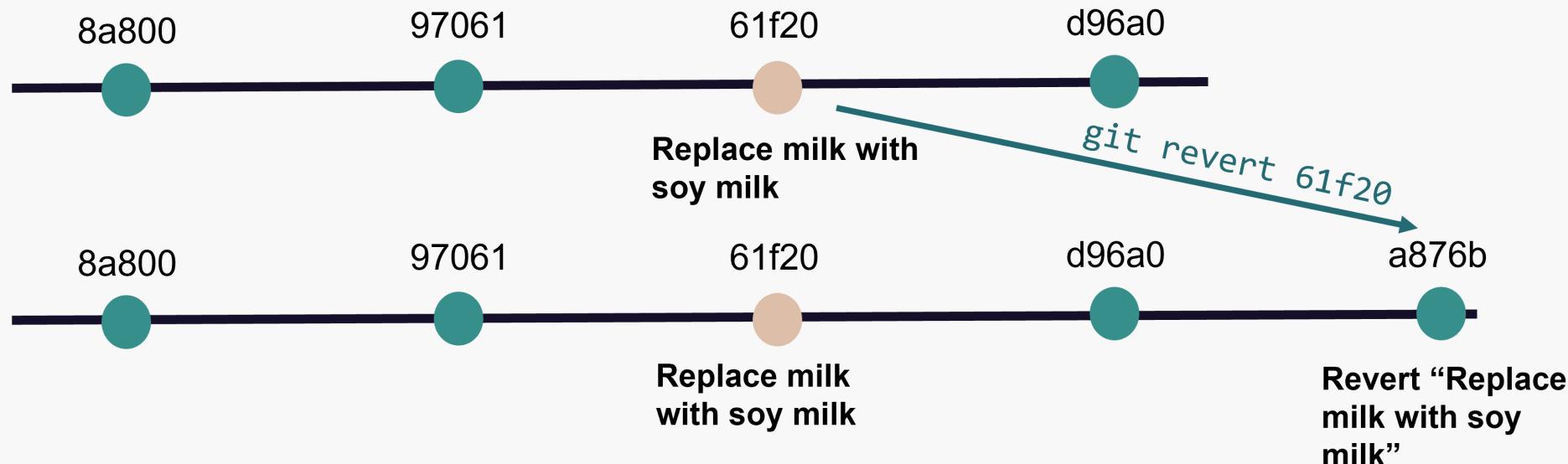
Checkout a previous commit

- Bring your work space back in time temporarily with `git checkout`



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 committed changes



Now you

Revert a part of your recipe Complete the task 3 “Time travel” (5 min)

Publish your repositories

Publish your repositories

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

Publish your repositories

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
- Create a Github pages website alongside your repo
- Encourage people to write issues if they find problems
- ...

Ignoring 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
- 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.

Thanks for your attention

Questions?

Preparation for tomorrow

- Tomorrow we collaborate on our cookbooks in teams of 2
 - Enter your Github Account Name and the link to your repo [here](#)
- Look for the Github Name of your partner and add them as a collaborator to your repository
- Accept the invitation of your partner to their repository
 - You will get an Email or you can do it on Github

