Git and GitHub for code management

The benefits of using collaborative version control for your science projects

GitHub

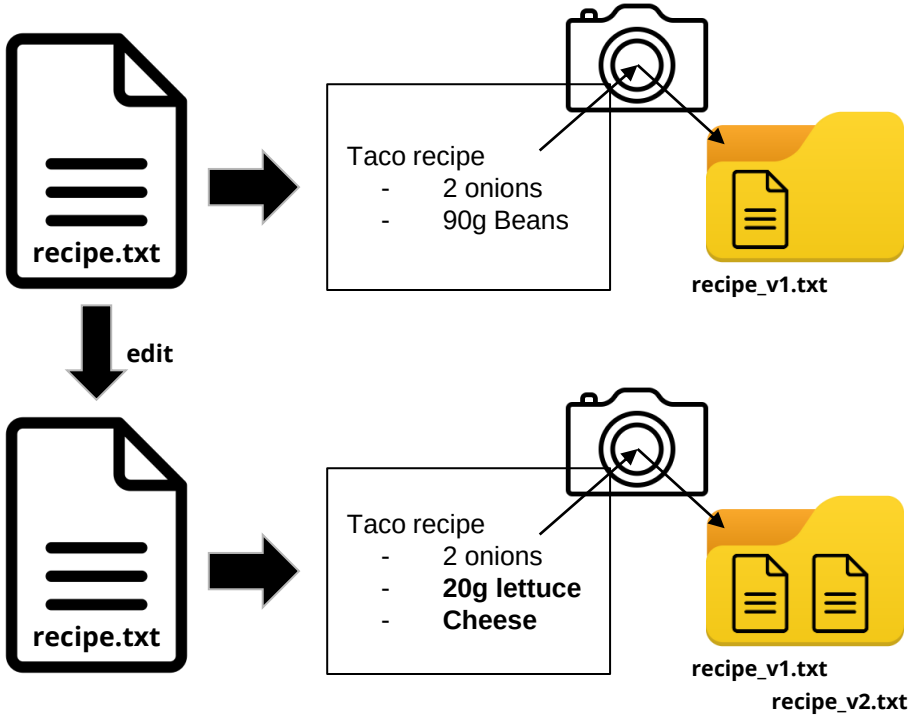
University of Oslo (UiO) - Lasse T. Keetz (NHM, Dept. of Geoscience), Peter Horvath (NHM) - Autumn, 2021

Disclaimer: this overview is largely based on materials from the “coderefinery” project and a coding workshop given at MetOs (UiO). Links given at the end.

What is version control?
Why should I use it?

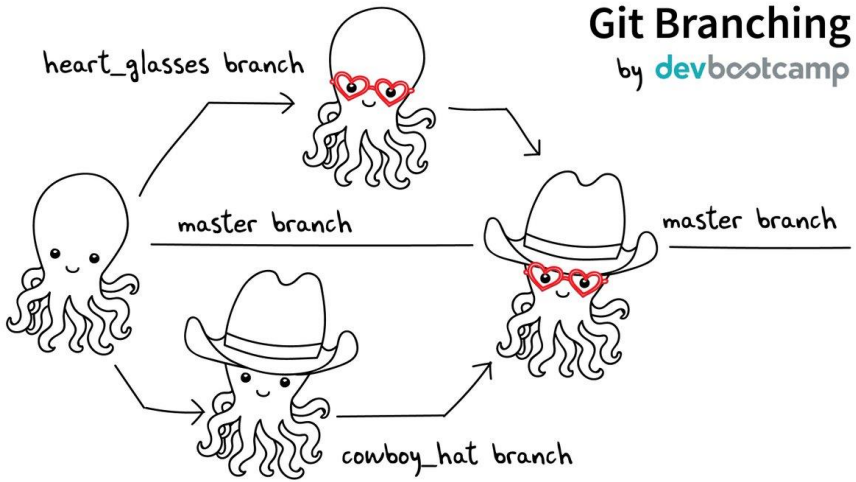
The essence of a version control system

- Records **snapshots** of project files, for example:
  - Scripts (R, python, etc.)
  - Software
  - Documents / manuscripts
  - Data
- Saves the **history** of files, makes it possible to easily restore previous versions
  - No more: “script.R” → “script_final.R” → “script_final_nowforreal.R” :-)
  - Easily return to working versions if you accidentally break something
- Improves reproducibility!



Branching and collaboration

- Branching
  - Efficiently work on **several features** of your code
  - Safely & simultaneously divide tasks between collaborators → merge results
- Example: you have a working script that reads in species distribution data.
  - **Task / person A**: add function to plot species distribution
  - **Task / person B**: add statistical analysis
- Improves:
  - “Can you please send me the latest version of your code?”
  - “I will finish my work and then you can start with your changes!”

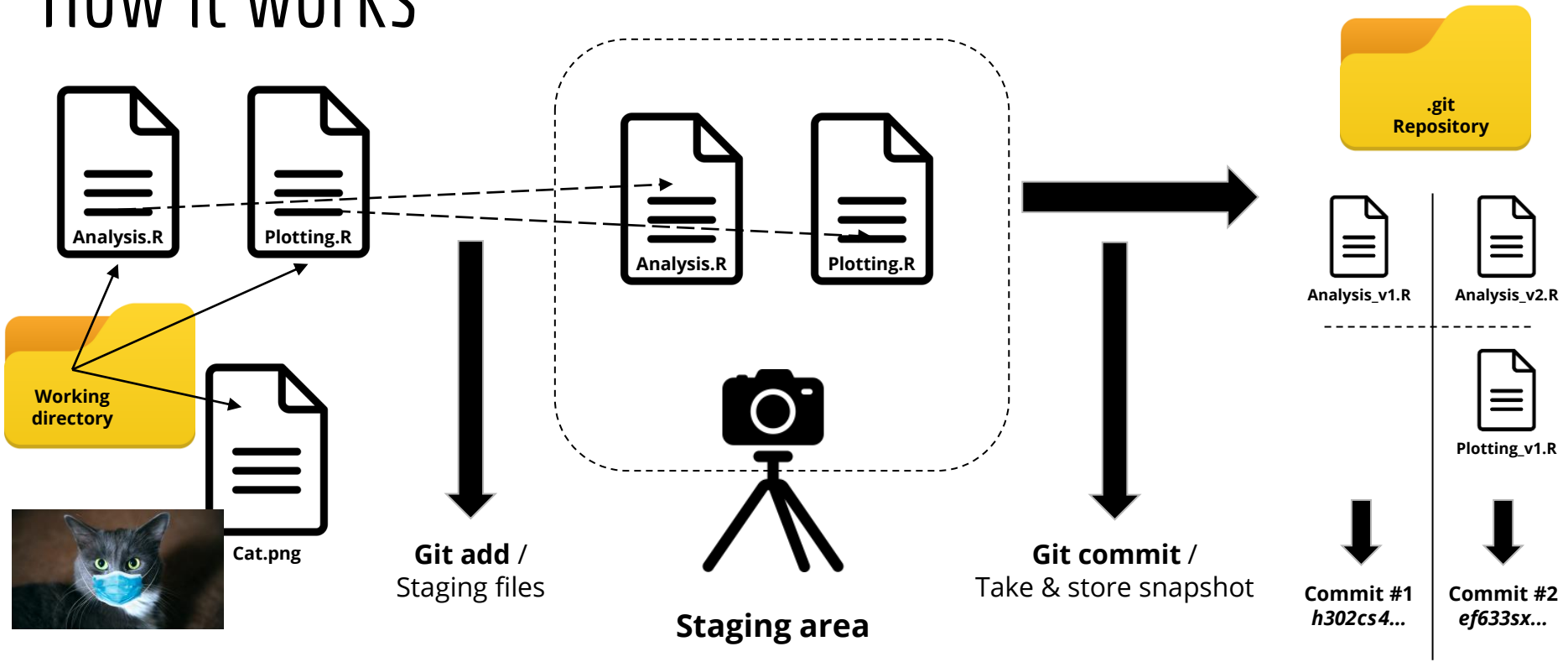


What is Git?
How does it work?

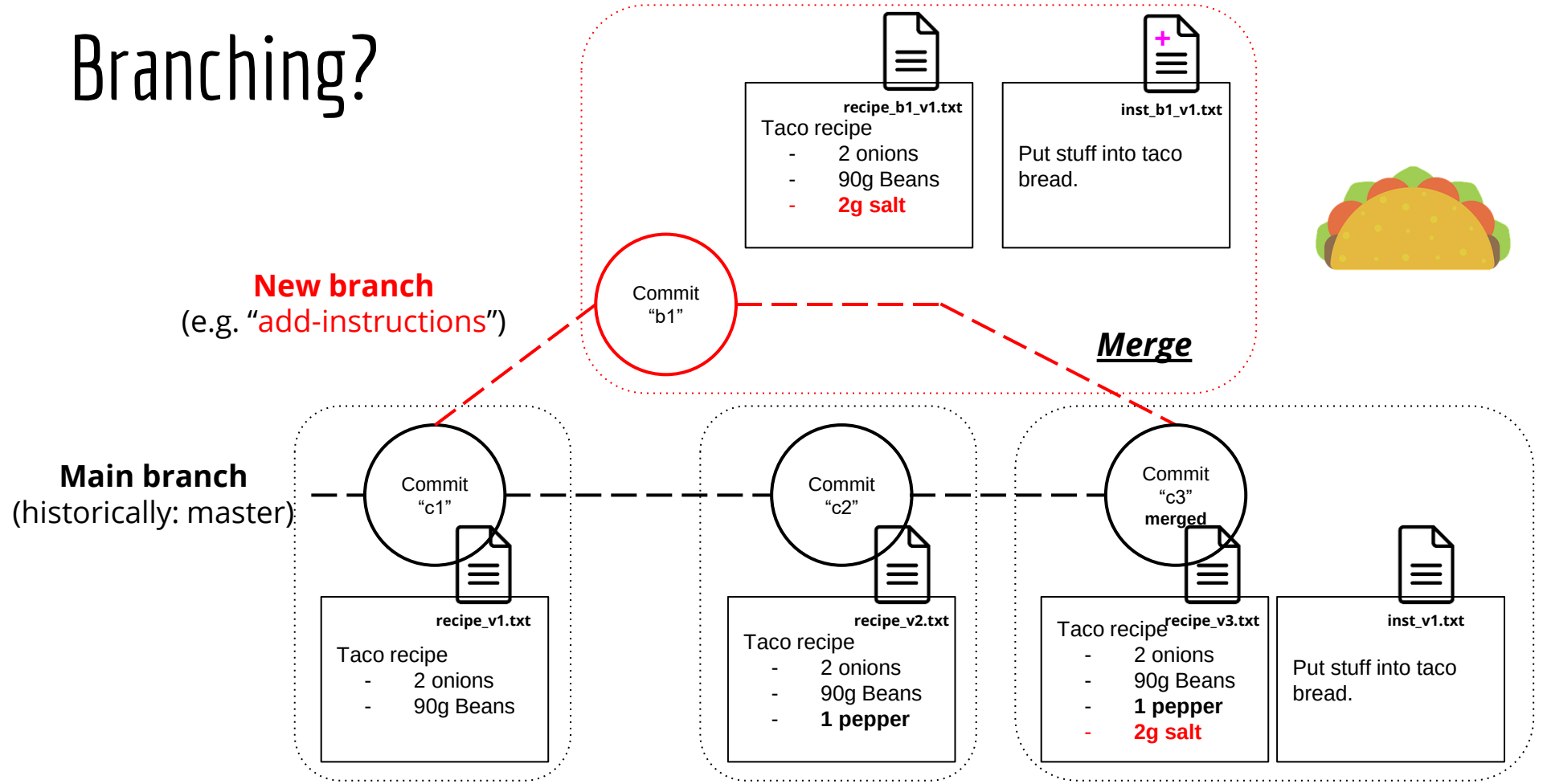
Git basics 

- Git is a widely used **version control system** (there are others! E.g. SVN)
- Provides tools to take & store snapshots and to manage the changes
- Creates **local** hidden **folder** (“.git”) containing the repository → not to be confused with working directory!
    - You could delete the repository (history of changes) and still keep your local files
- Does not record anything unless you specifically ask it to!

How it works


Analysis.R
Plotting.R
Working directory
Cat.png
Git add /
Staging files
Analysis.R
Plotting.R
Staging area
Git commit /
Take & store snapshot
.git
Repository
Analysis_v1.R
Analysis_v2.R
Plotting_v1.R
Commit #1
h302cs4...
Commit #2
ef633sx...

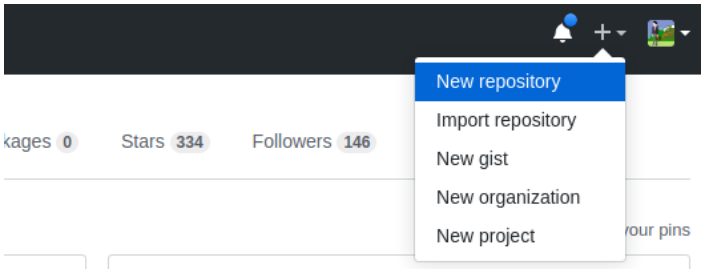
Branching?


recipe_b1_v1.txt
Taco recipe
- 2 onions
- 90g Beans
- 2g salt
inst_b1_v1.txt
Put stuff into taco bread.
New branch
(e.g. "add-instructions")
Commit "b1"
Merge
Main branch
(historically: master)
Commit "c1"
recipe_v1.txt
Taco recipe
- 2 onions
- 90g Beans
Commit "c2"
recipe_v2.txt
Taco recipe
- 2 onions
- 90g Beans
- 1 pepper
Commit "c3" merged
recipe_v3.txt
Taco recipe
- 2 onions
- 90g Beans
- 1 pepper
- 2g salt
inst_v1.txt
Put stuff into taco bread.

What is GitHub?
How does it work?

What is GitHub, why use it?

- Service to host your repositories online (**remote**) → online backups!
- For-profit (Microsoft), but no limit on number of repos in free version
- Enables collaboration and easy distribution of projects (only need to share link!)
- Nice web interface with lots of handy tools
  - E.g., Code reviews → nice way to get feedback for specific lines of code



How to get started

1. Create user account!

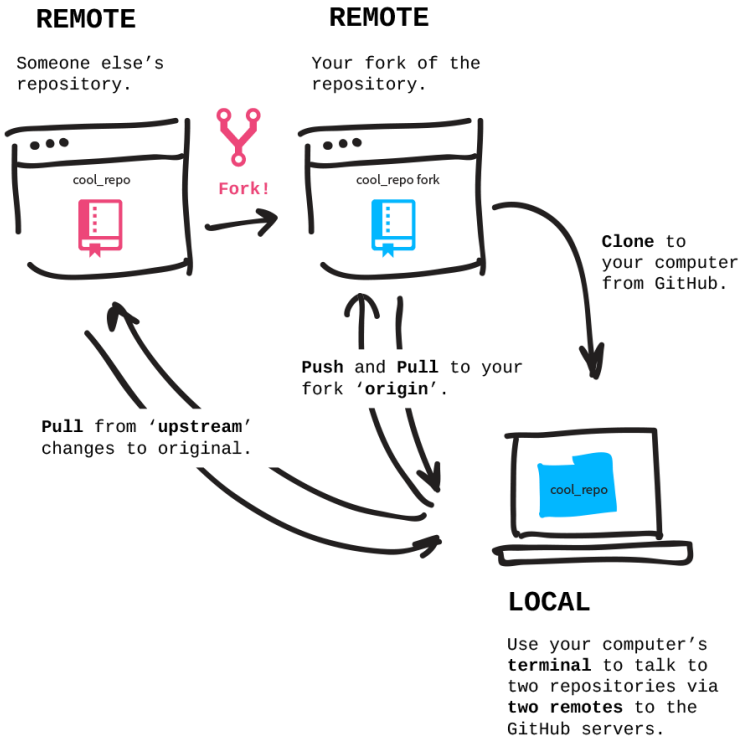
To create your own Git repository on GitHub, you can either…

- Create a new repo in web interface
- Upload an existing repo (**push**)

To work on an existing project, you can…

- Copy repo to your machine (**clone**)
- Copy (“scan”) an existing online remote repo to your own GitHub account (**fork**)

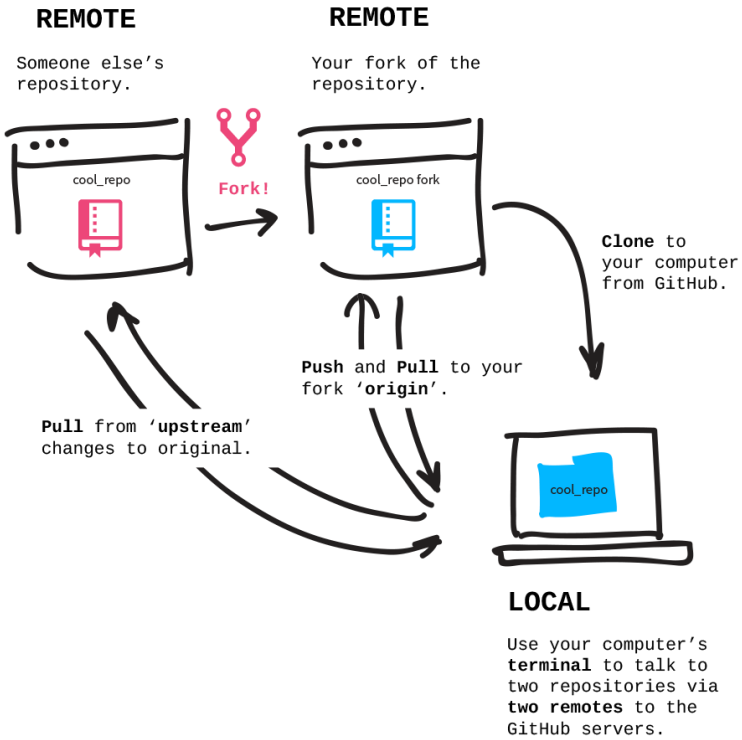
Remember: copying the state (commit) of a repository always includes the history up to that point and it will not update unless you specifically give the order!



How to sync local and remote

- Get latest online version to your machine (already cloned project)
  → **Pull** (technically: **fetch** + **merge**)
- When you have changed / added code and want to save progress online
  → **Push** [to desired remote and branch]

- Basic idea analogous to using shared Google Drive folder, for example! But optimized for code management.



Recap: most important vocabulary

General git:

- **Git init -** initialize a new repository (create hidden ".git" folder in empty directory)
- **Git add** - stage files ("put in snapshot frame") you want to add/change in repository
- **Git commit** - take and store snapshot of staged files with unique ID
- **Git branch** - independent development line containing all files+histories of parent at state of creation
- **Git merge** - combine two states of development, either from local or remote branches
- **Git tag** - Pointer to a specific commit that contains descriptive version number/name after reaching a "milestone" (e.g. finished data analysis workflow. Examples: *thesis-printed*, *SDMproject_v1.0.0*)
- **Git log** - list history of commits
- **Git status** - shows if working state is clean, e.g. check if files were changed but not added/committed

Online functionalities:

- **Git clone** - copy a repository from an online URL into your machine
- **Git fork** - copy ("scan") an entire existing repo into your own GitHub account, creates separate entity
- **Git pull** - retrieve and merge (see slide) latest changes from remote repository
- **Git fetch** - check for and display updates from remote repository, but without merging
- **Git push** - push your local changes to remote repository (merge upstream)
- **Pull request** - ask a remote host to incorporate (i.e., "pull") your local changes

Takk!

Links for additional materials

- Coderefinery workshop for version control with Git
- MetOs (Dept. of Geosciences, UiO) workshop on collaborative Git and good coding practice
- Coderefinery workshop on using Conda for package and virtual environment management (important to make sure your code is really reproducible!)
- Coderefinery workshop on using Python for scientific computing

Final thoughts for NHM

- One building block to make sure people use the same version of code
  → E.g. in data labs, version mismatches common problem
- Great tool to share code between supervisors and students, for collaborating students, etc.
  → Avoid “as long as it looks like it’s working, it should be fine” mentality
  → Correct and understandable code as important as thesis text!
- University GitHub → Separated “copy” of regular GitHub, needs special access rights
  → Avoid unless you have sensitive data (harder to collaborate & cannot access anymore if you leave UiO)