

git/GitHub Introduction



Lei Ma, PhD
Informatics Group



Gregg Thomas, PhD
Informatics Group

We help FAS researchers with bioinformatic analysis



ONE-ON-ONE
CONSULTS



ONGOING
COLLABORATIONS



BIOINFORMATICS
WORKSHOPS

Specialized Services



DATA CONCIERGE



OFFICE HOURS



NEWSLETTER

“Data available
upon request” is
no longer
acceptable,
shouldn't it be
the same for
code?

We annotated the aligned sRNAs using our in-house annotation pipeline ([source code available upon request](#)). The annotations were then resolved hierarchically with the following categories

interrogation high-resolution 3D particle data sets. We describe an efficient algorithm for performing a volume reconstruction of the lithology field defined via particles ([code available upon request from the author](#)). The algorithm generates an Approximate

Don't do this!



Your code is part of your methods: what you include matters

“A differential abundance analysis was conducted using the Wilcoxon Rank Sum Test...These analyses were performed using the R statistical program.”

- ? Did they code their own or use a software?
- ? What parameters were given?
- ? What data cleaning was done beforehand?

“Maaslin2 (version 1.14.1) [[32](#)] was used for differential abundance analysis on a rarefied dataset with TSS normalization, prevalence set to 15%, and cage household set as a random effect.”

- ✓ Software and version are listed and cited
- ✓ Parameters and data cleaning/normalization are listed
- ✓ Could recreate if we read the tutorial for Maaslin2, maybe
- ? Not all parameters are listed, some are assumed
- ? Takes a lot of effort to recreate

Two types of sharing code: for reproduction and for reuse

Code shared for reproduction



Data and materials availability: Raw sequencing reads are available from the NCBI BioProject database (PRJNA814281 and PRJNA979973). Processed data are available from Zenodo ([75](#), [76](#)); source code for the sequencing pipeline, downstream analyses, and figure generation are available from GitHub ([77](#), [78](#)).

[Link](#) to paper by Alejandro Couce et al, Science 2024 “Changing fitness effects of mutations through long-term bacterial evolution”

Code shared for reuse

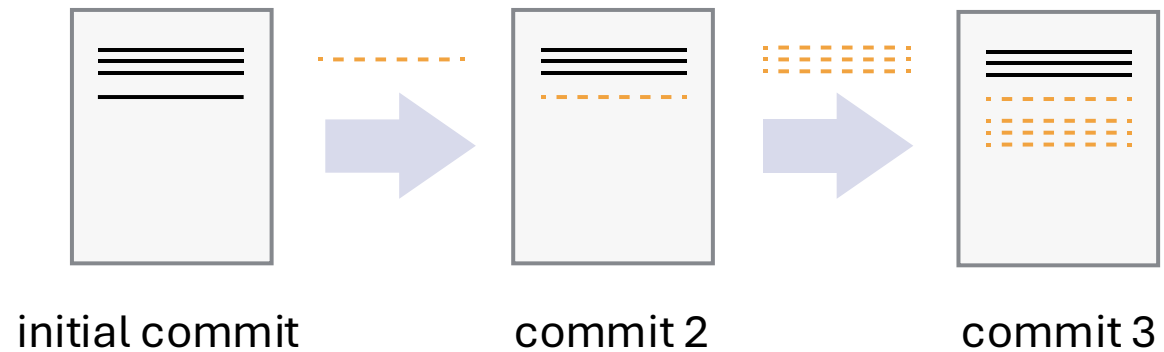
EURYALE is available through a GitHub repository, which can be found at <https://github.com/dalmolingroup/euryale>, with documentation available through <https://dalmolingroup.github.io/euryale/>. MicroView, although executed as part of EURYALE, can be executed stand-alone and has its own source code available in the following repository: <https://github.com/dalmolingroup/microview>.

[Link](#) to paper by Cavalcante et al, IEEE 2024 “EURYALE: A versatile Nextflow pipeline for taxonomic classification and functional annotation of metagenomics data”

Feature	 git	GitHub 	GitHub Desktop
Type	Local version control software	Cloud-based hosting service	Software to help manage GitHub repositories with a user interface
Installation	Downloaded and installed locally	Web-based, no installation needed	Downloaded and installed locally
Cost	Free and open-source	Free and paid tiers available	Free
Dependency	Can work independently	Requires Git to function	Comes with its own copy of git
Ownership	Open-source community; (Maintained by the Linux Foundation)	Owned by Microsoft	Owned by Microsoft
Internet Requirement	Works offline	Requires internet connection	Requires internet connection
Primary Function	Version control and code management	Repository hosting and collaboration	Help manage git/GitHub repos
Access	Local machine only	Accessible from anywhere	Local machine only
Features	Basic version control operations	Additional features like issue tracking, pull requests, wikis	Most features of git & GitHub
Security	Local authentication	Two-factor authentication, access control	Connects to GitHub account

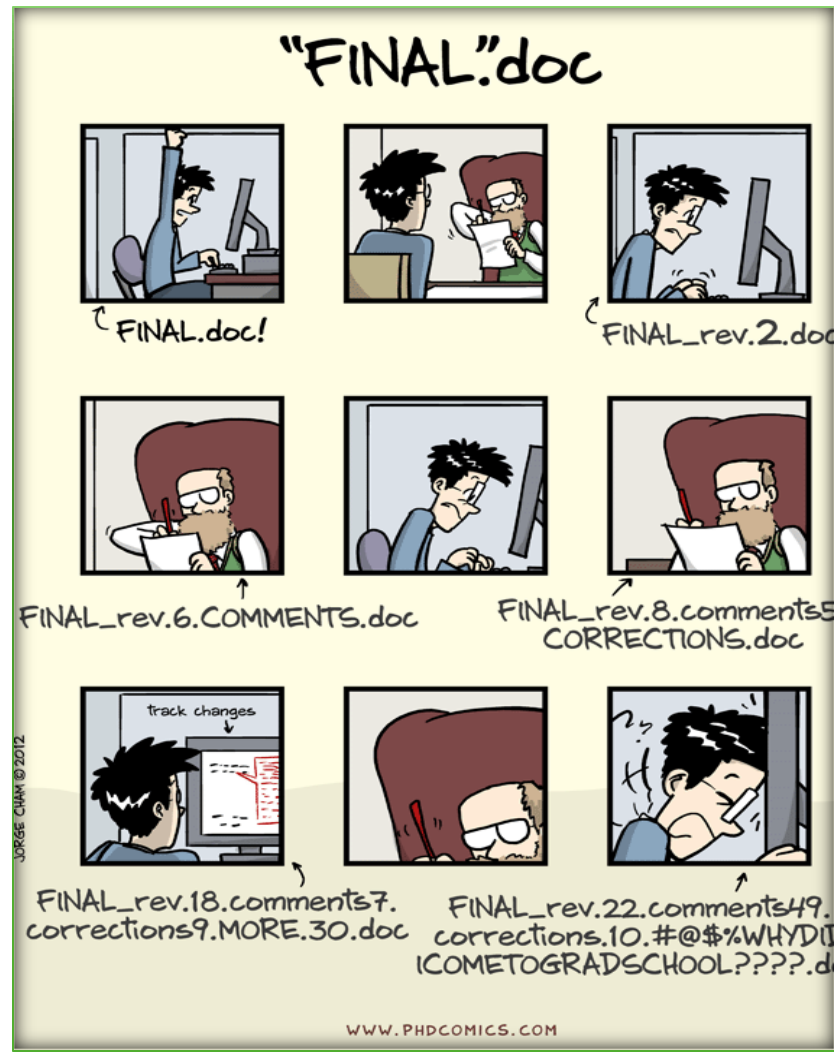
What is git?

It's manual
version control

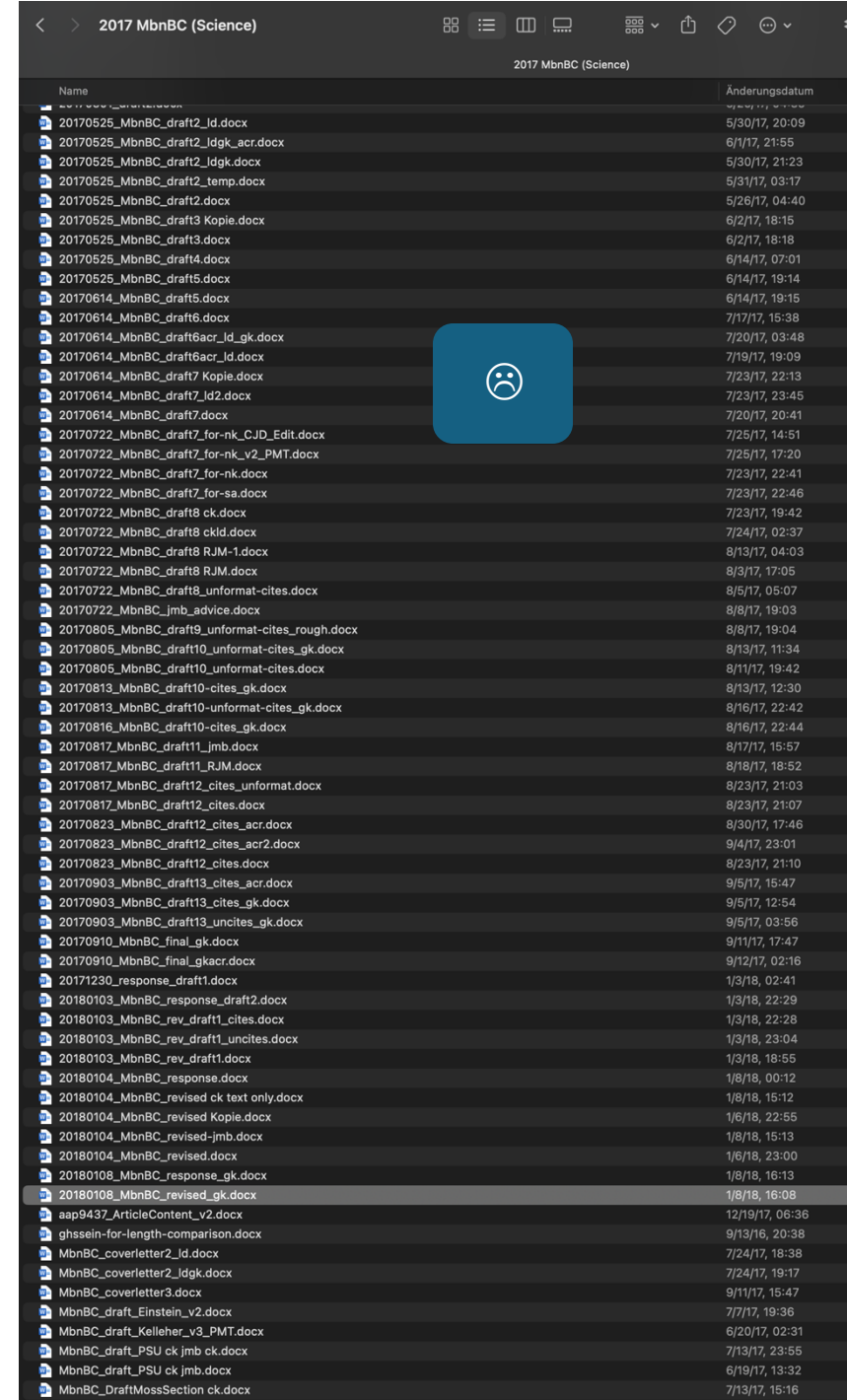


- Allows **manual** snapshotting of directory
 - like Time Machine, Dropbox, Google Docs, but not automatic
- Works **locally** (i.e. does not upload to anywhere on the internet)
- A hidden folder (.git) tracks changes, not copies
- Works **line by line**

Changing file names is not version control



“notFinal.doc” by Jorge Cham, <https://www.phdcomics.com>



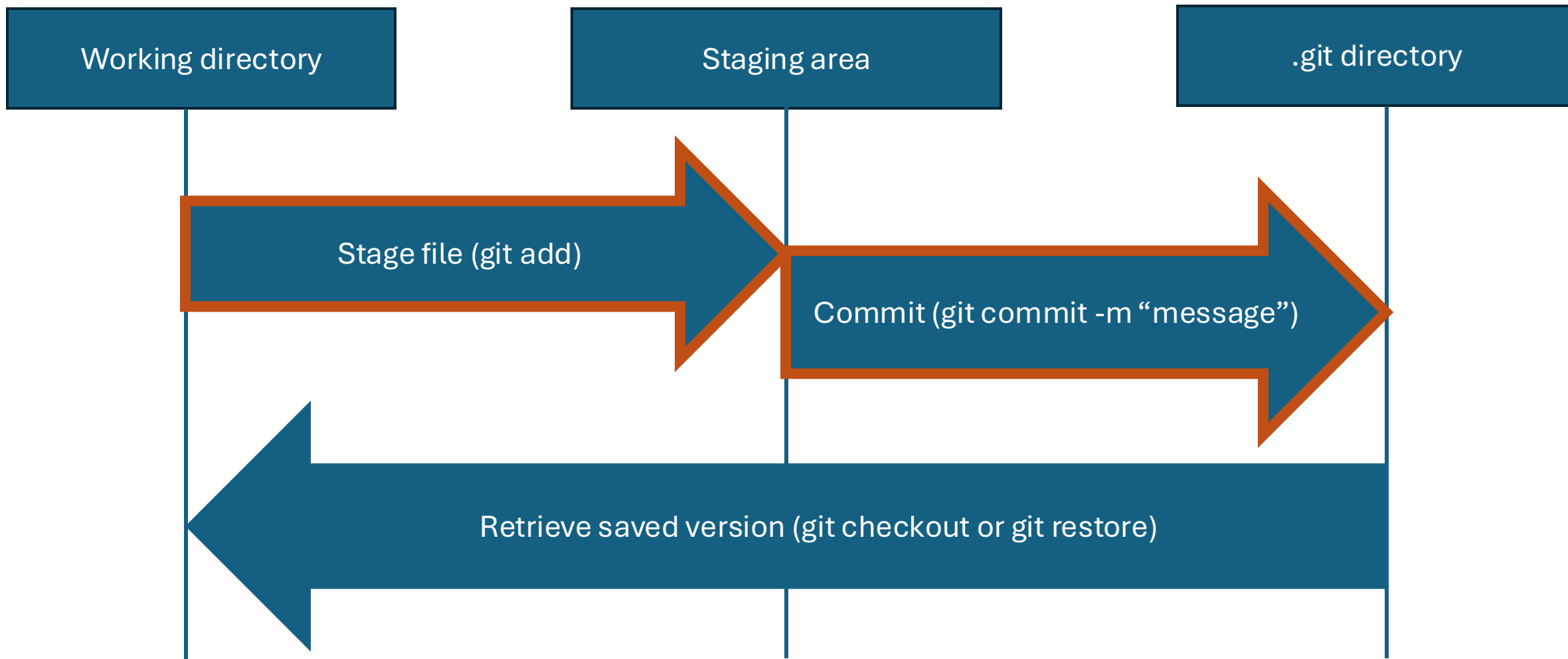
Why use git over Google Drive, Word, etc?

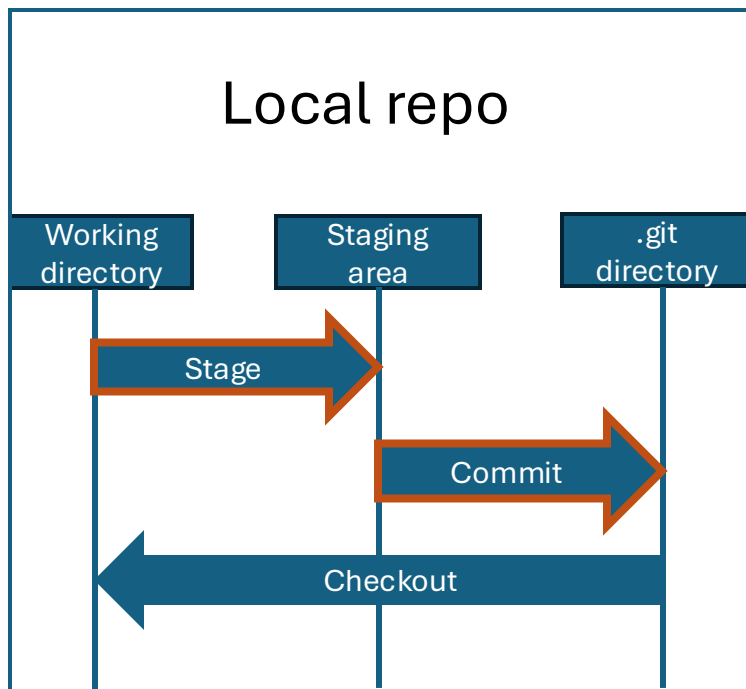
- Records **all changes and who made them**
 - compare to track changes in Word after you've "accepted changes"
- You have **control over snapshots**
- You can **experiment with different versions** while keeping a working version intact
- Will **never accidentally erase** something.
- You can **time travel** across different versions of your files

Summary of use cases

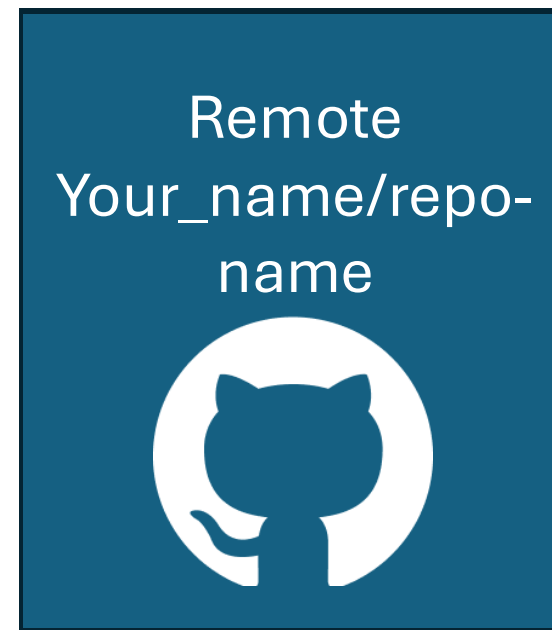
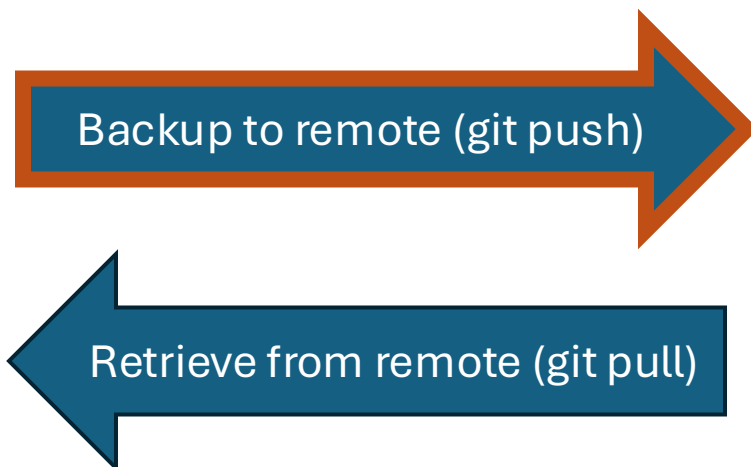
- **Version control**
- Collaboration
- **Publication**







git on your local machine



GitHub on remote servers

First steps

- Create a GitHub account
- Download and install GitHub Desktop
- Link GitHub Desktop to your account
- You should see something like this ->
- Click “Create a New Repository”

Let's get started!

Add a repository to GitHub Desktop to start collaborating



Create a Tutorial Repository...



Clone a Repository from the Internet...



Create a New Repository on your Hard Drive...



Add an Existing Repository from your Hard Drive...

Creating your first repo

- Makes a new folder AKA repo
- No spaces in repo name
- MIT license: Anyone can use (including commercially) as long as credit is given
- GNU GPLv3: Anyone can use (including commercially) as long as credit is given and changes are record. No proprietary uses.

git-workshop

Description

This will hold the code for my paper titled "Awesome Paper"

Local Path

/Users/leima/Github

Choose...

☒ Initialize this repository with a README

Git Ignore

None



License

None



Current Repository
git-workshop

Current Branch
main

Publish repository

Publish this repository to GitHub

Changes 15

History

README.md

15 changed files

☒

data_raw/README.md

+

☒

environment.yml

+

☒

figures/figure_1_flipper_bill.png

+

☒

Palmer_penguins.Rproj

+

☐

penguins_notebook.md

+

☒

public_notebook.nb.html

+

☒

public_notebook.Rmd

+

☒

README.md

•

☒

scripts/figure_1_flipper_bill.R

+

☒

scripts/process_data.R

+

@@ -1,2 +1,3 @@

git-workshop

-

+

+ This is a repository for my paper titled "Awesome paper"

Summary (required)

Description

Commit to main

Committed 42 minutes ago
Initial commit

Undo

staged

not staged

new files

modified

Current Repository: **git-workshop**

Current Branch: **main**

Publish repository
Publish this repository to GitHub

Changes: 15 | History

15 changed files

- ☒ data_raw/README.md
- ☒ environment.yml
- ☒ figures/figure_1_flipper_bill.png
- ☒ Palmer_penguins.Rproj
- ☐ penguins_notebook.md
- ☒ public_notebook.nb.html
- ☒ public_notebook.Rmd
- ☒ README.md
- ☒ scripts/figure_1_flipper_bill.R
- ☒ scripts/process_data.R

Summary (required)

Description

Commit to main

Committed 42 minutes ago
Initial commit

README.md

@@ -1,2 +1,3 @@
git-workshop
-
+ This is a repository for my paper titled "Awesome paper"

☒ = file is staged and ready to be committed

Push/upload all commits to GitHub

commit button

undo commits that haven't been pushed/uploaded

commit
message

What to upload

What to include	What NOT to include
<ul style="list-style-type: none">• Scripts• Notebooks• READMEs• Configuration files (yaml, config files)• Metadata files (maybe)	<ul style="list-style-type: none">• Data• Software (conda envs, containers)• Files >50 MB• Passwords and sensitive information (!!!)• Log files (slurm logs, error logs, etc)• Junk files (temp, OS-generated junk)

git terminology

Term	Definition
repository (repo)	a folder tracked by git
remote	a repository hosted on a server (e.g. GitHub)
add/stage	add file to staging area (start tracking it), a checkbox in GH Desktop
staging area	intermediate area where files are tracked before they are added to git
commit	make a snapshot of staging area to repository with commit message
push	upload changes to remote repo (GitHub)
gitignore	a file called “.gitignore” that lists files to be ignored by git and GH desktop
README	a top level file called “README.md” that appears on your repo landing page
license	needed upon publication to tell others how to use your code

How to organize for reproduction vs reuse

Reproduction (analysis):

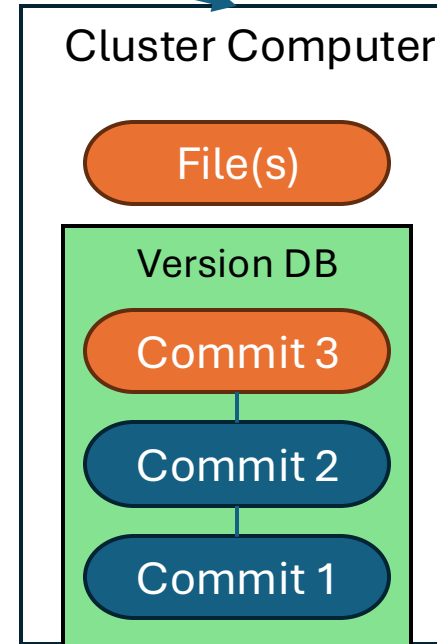
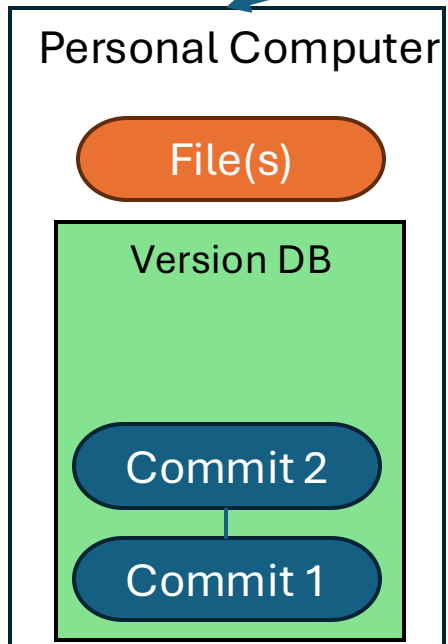
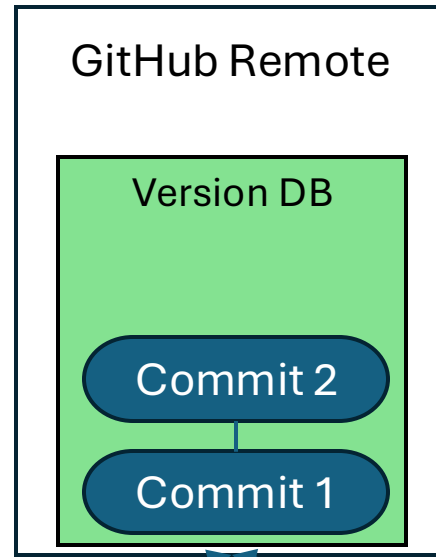
Examples [here](#) and [here](#)

- Provide installation instructions for the software dependencies
- GitHub README should provide links or instructions for downloading raw or intermediate data, link to paper, describe folder structure
- Folder organization based on Data, Metadata, Analysis, etc.
- Organize your code in the order that it was run for your paper
- Label each script with a number or put them into folders marking different aspects of your analysis
- Add readme files or notebooks to each level

Reuse (software):

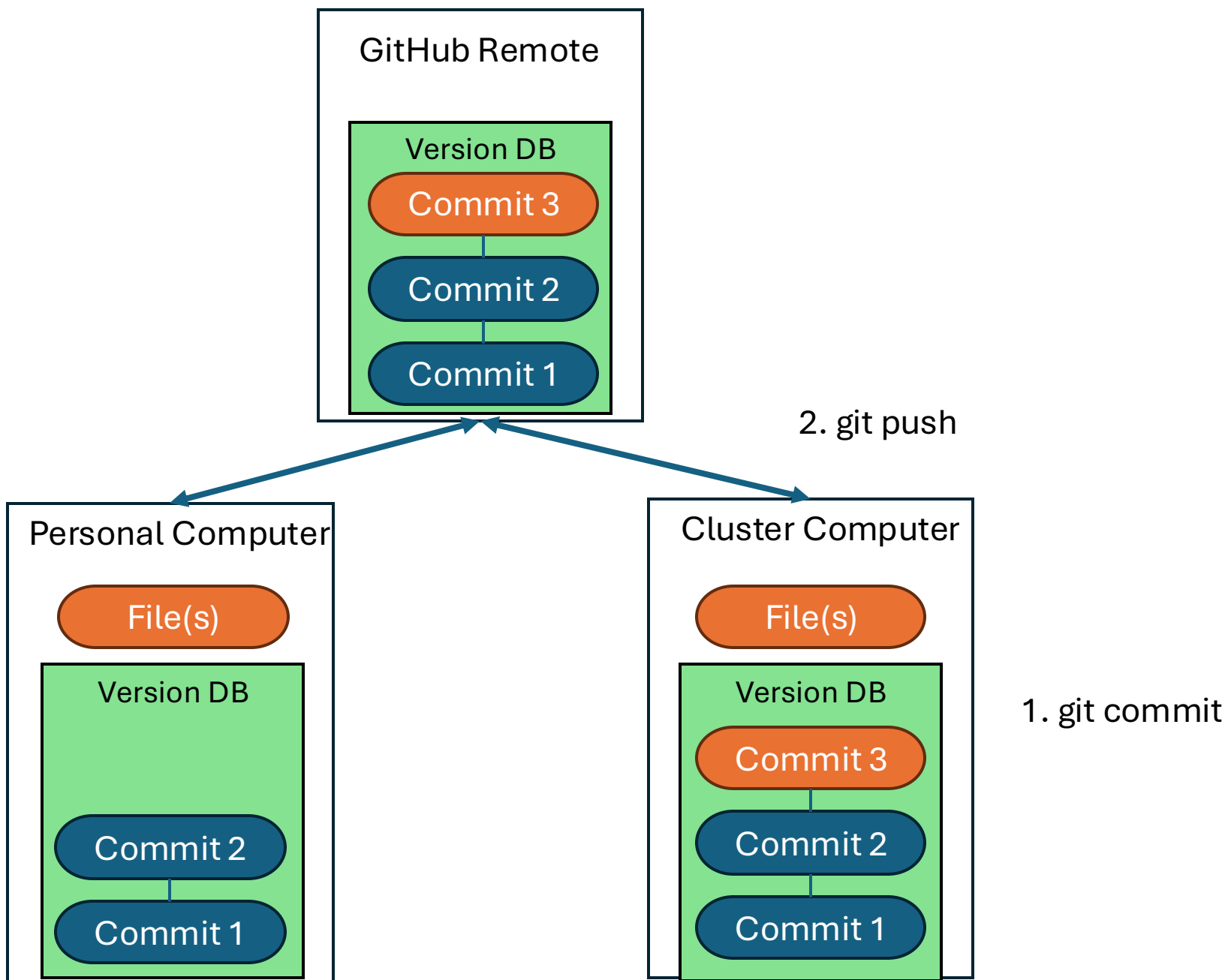
Example [here](#)

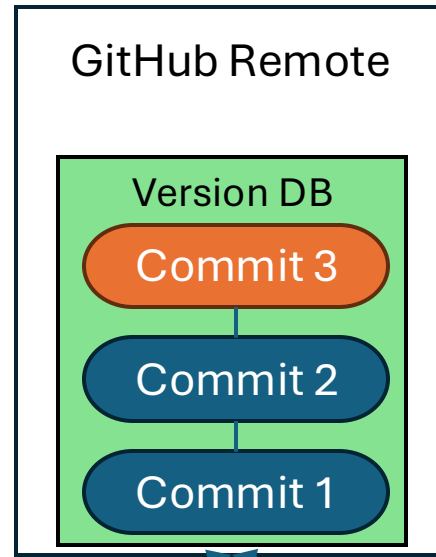
- Provide installation instructions for the software dependencies
- GitHub README should be comprehensive and tell you what the inputs and outputs are, and what each parameter means
- Folder structure is different and split based on configuration, dependencies, modules, scripts, etc.
- Provide a small test dataset to demonstrate program and check that it is running properly.
- Write a documentation page if it's a larger project
- Remove hard-coded or personalized parameters as much as possible



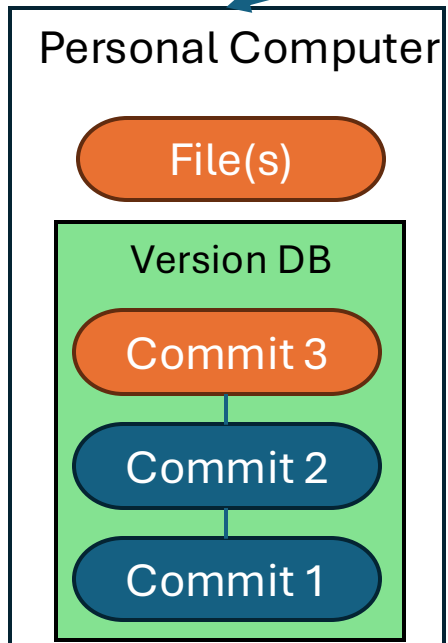
1. git commit



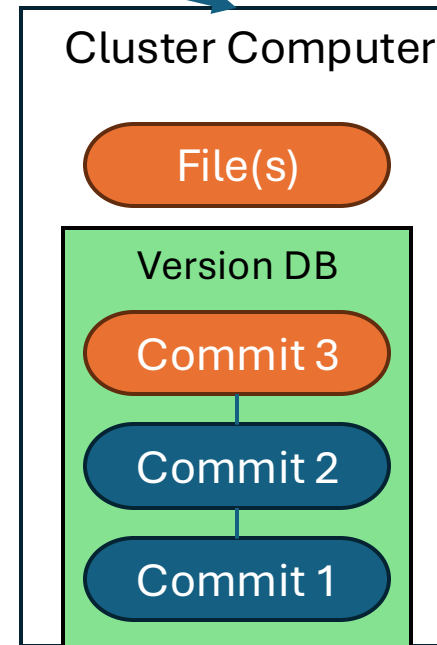




2. git push



3. git pull



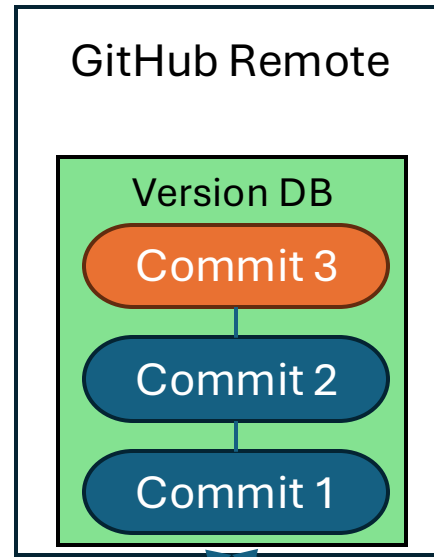
1. git commit



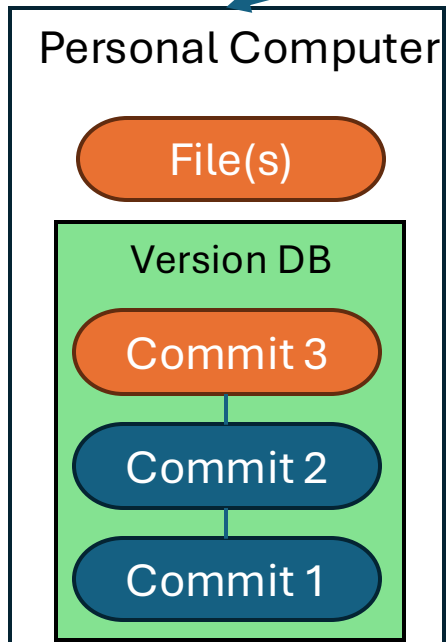


Start day

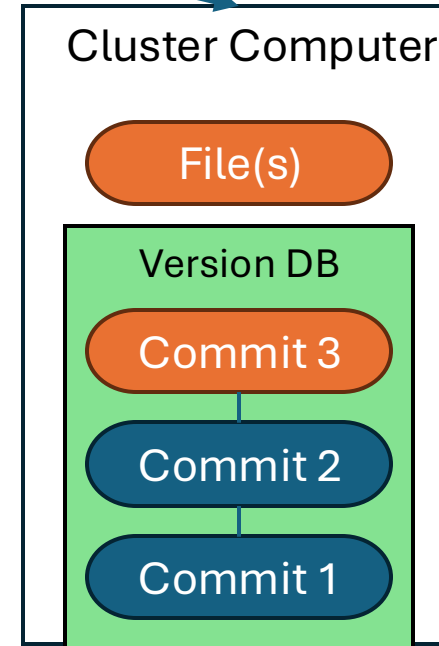
1. Pull changes
2. Make commits
3. Push changes



2. git push



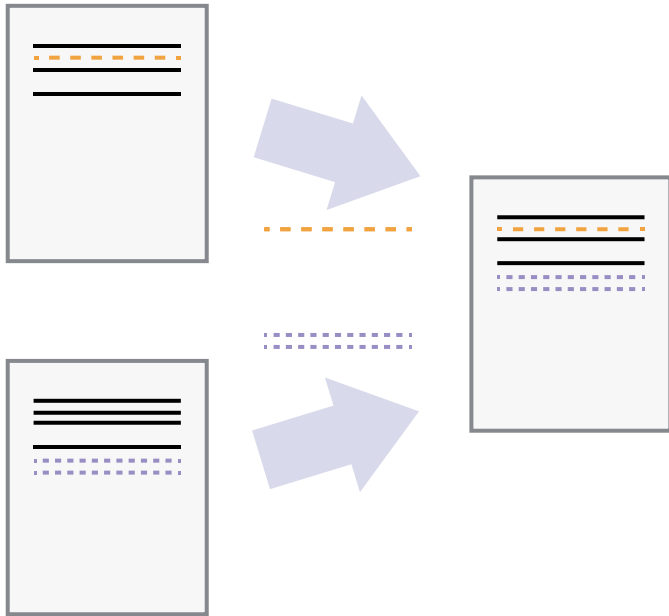
3. git pull



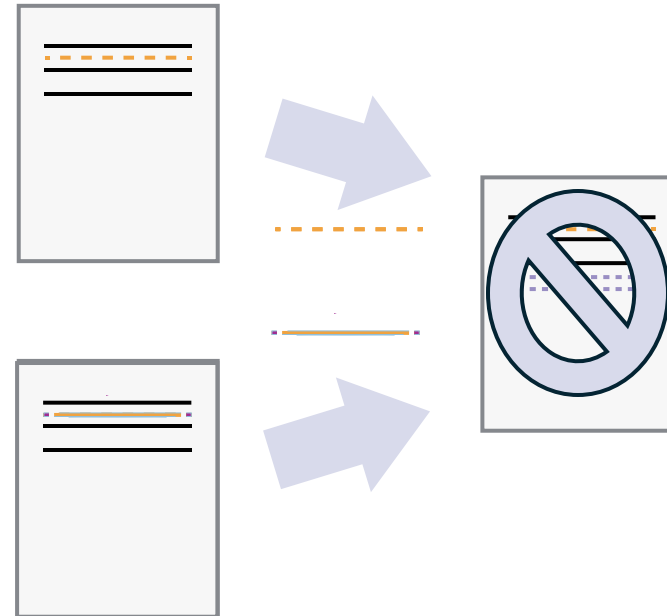
1. git commit



merging conflicts



Edits to different lines:
no conflict, automatically merge



Edits to the same line:
conflict, cannot merge

Git takeaways

1. **Commits are NOT automatic.** You have to tell git when and what to commit every time. So do it frequently, or you won't be able to go back to a version that you didn't commit.
2. **Always pull** to get the latest changes from the remote before you start working on your local version. Again, git does NOT operate automatically at all. Everything is manual.
3. **Always be aware of what you're committing** as permanently deleting things is annoying
4. Everything is recorded and almost everything can be undone, so **don't be afraid**

Getting a DOI and archiving your code

The screenshot displays the Zenodo web interface. At the top is a blue header with the Zenodo logo, a search bar labeled "Search records...", and navigation links for "Communities" and "My dashboard". A user profile dropdown shows the email "lei_ma@g...". Below the header is a grey bar labeled "My account".

The main content area is divided into two sections. On the left is a "Settings" sidebar with a list of options: Profile, Change password, Notifications, Security, Linked accounts, Applications, and GitHub (which is highlighted in blue). On the right is the "GitHub Repositories" section, which includes a "Sync now" button and a "Get started" guide.

The "Get started" guide consists of three numbered steps:

- 1 Flip the switch**: Select the repository you want to preserve, and toggle the switch below to turn on automatic preservation of your software. Below the text is a toggle switch labeled "ON" (example).
- 2 Create a release**: Go to GitHub and [create a release](#). Zenodo will automatically download a .zip-ball of each new release and register a DOI.
- 3 Get the badge**: After your first release, a DOI badge that you can include in GitHub README will appear next to your repository below. Below the text is a badge example: `DOI 10.5281/zenodo.8475` (example).

Summary

4 steps to sharing your code

1. Decide if you are sharing for reuse or reproduction
2. Write your READMEs and organize your folders
3. Upload to GitHub
4. Link to Zenodo

Bonus topics

Pick one!

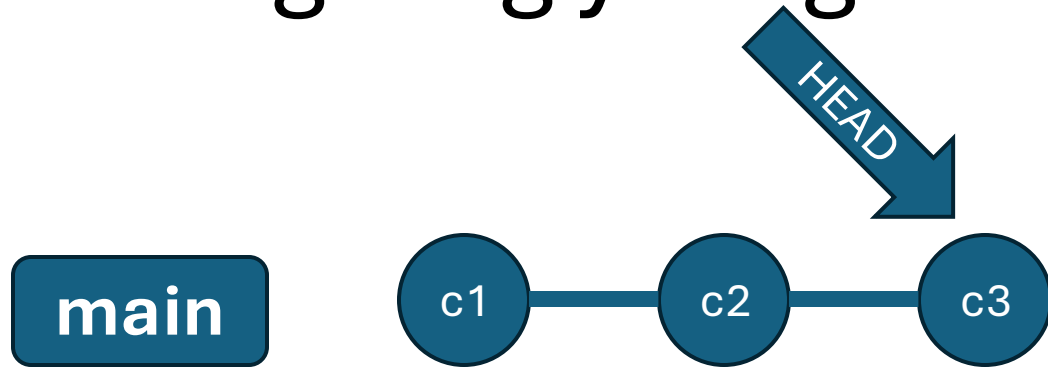
1. git branching & merging
2. collaboration (forking and pull requests)
3. GitHub pages
4. GitHub lab organization

git branching explanation

visual demo of branching

- Demo on [learngitbranching](https://learngitbranching.js.org/) : <https://learngitbranching.js.org/>
- git branch bugfix
- git commit
- git switch main
- git commit
- git merge bugFix
- git branch -d bugFix

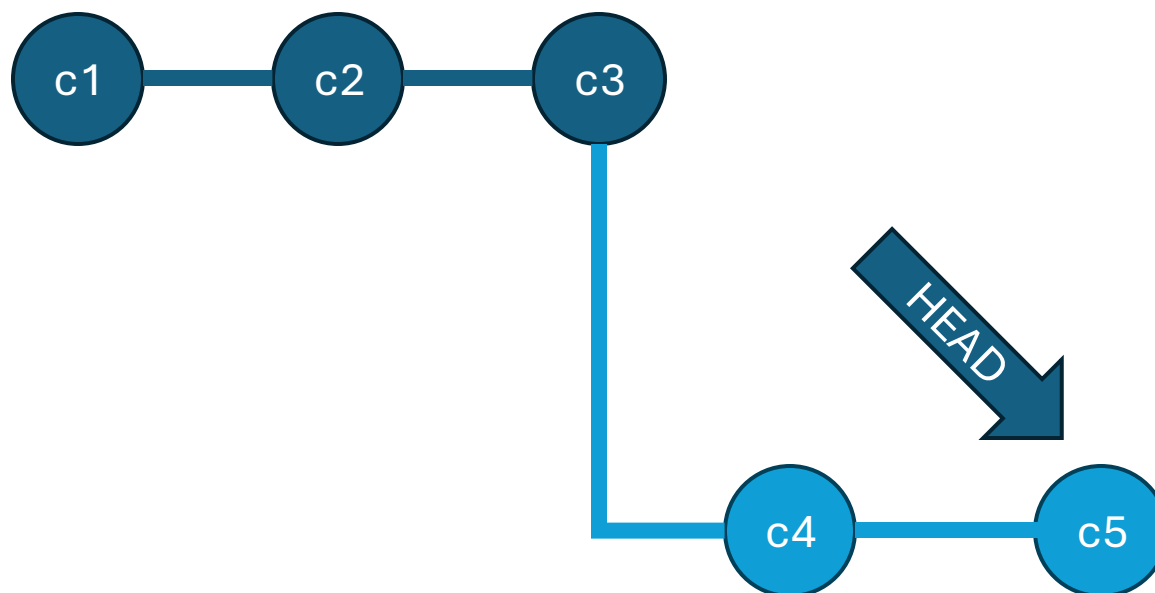
Navigating your git history: the HEAD pointer



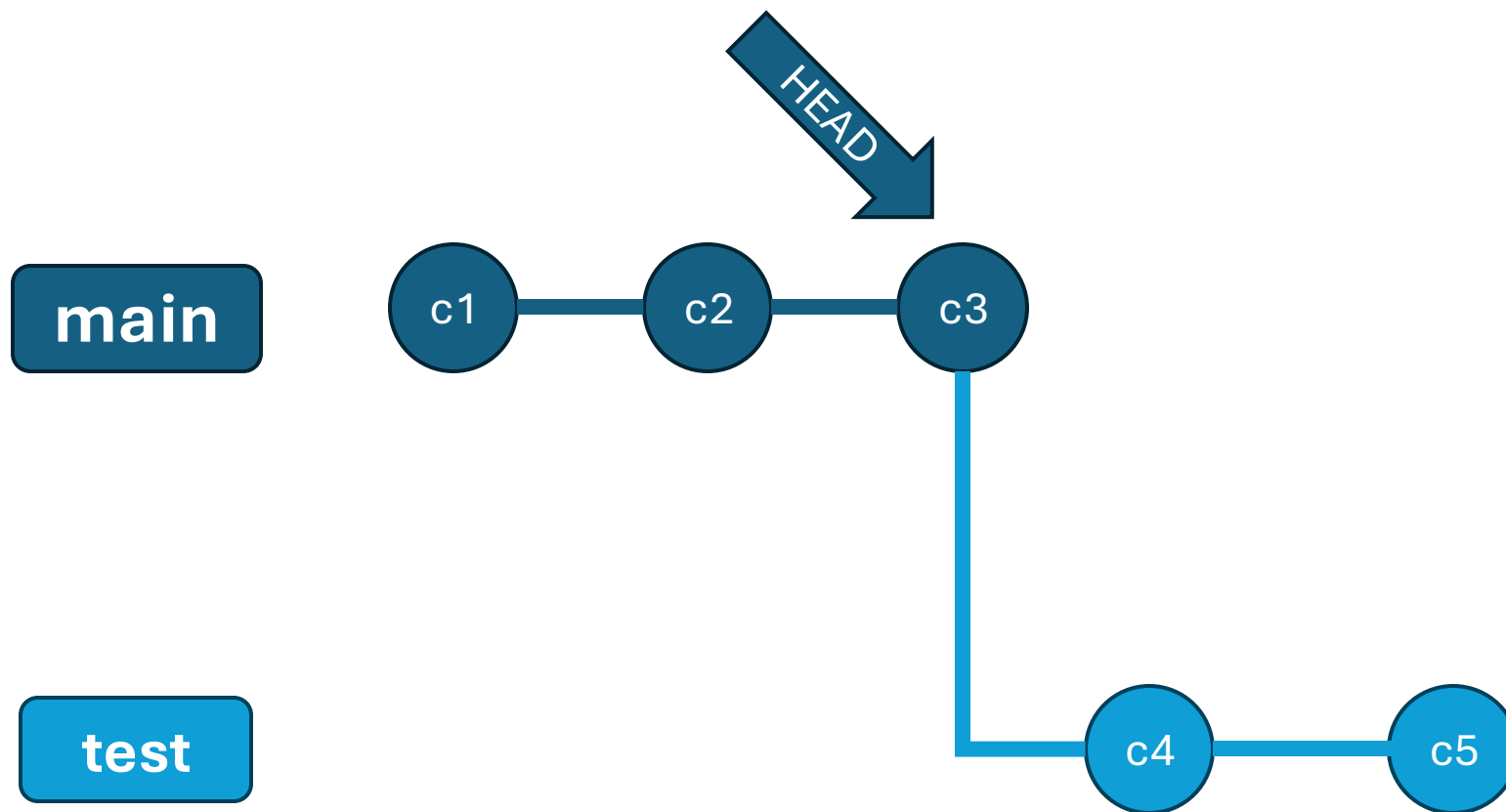
git commit
git commit
git commit

main

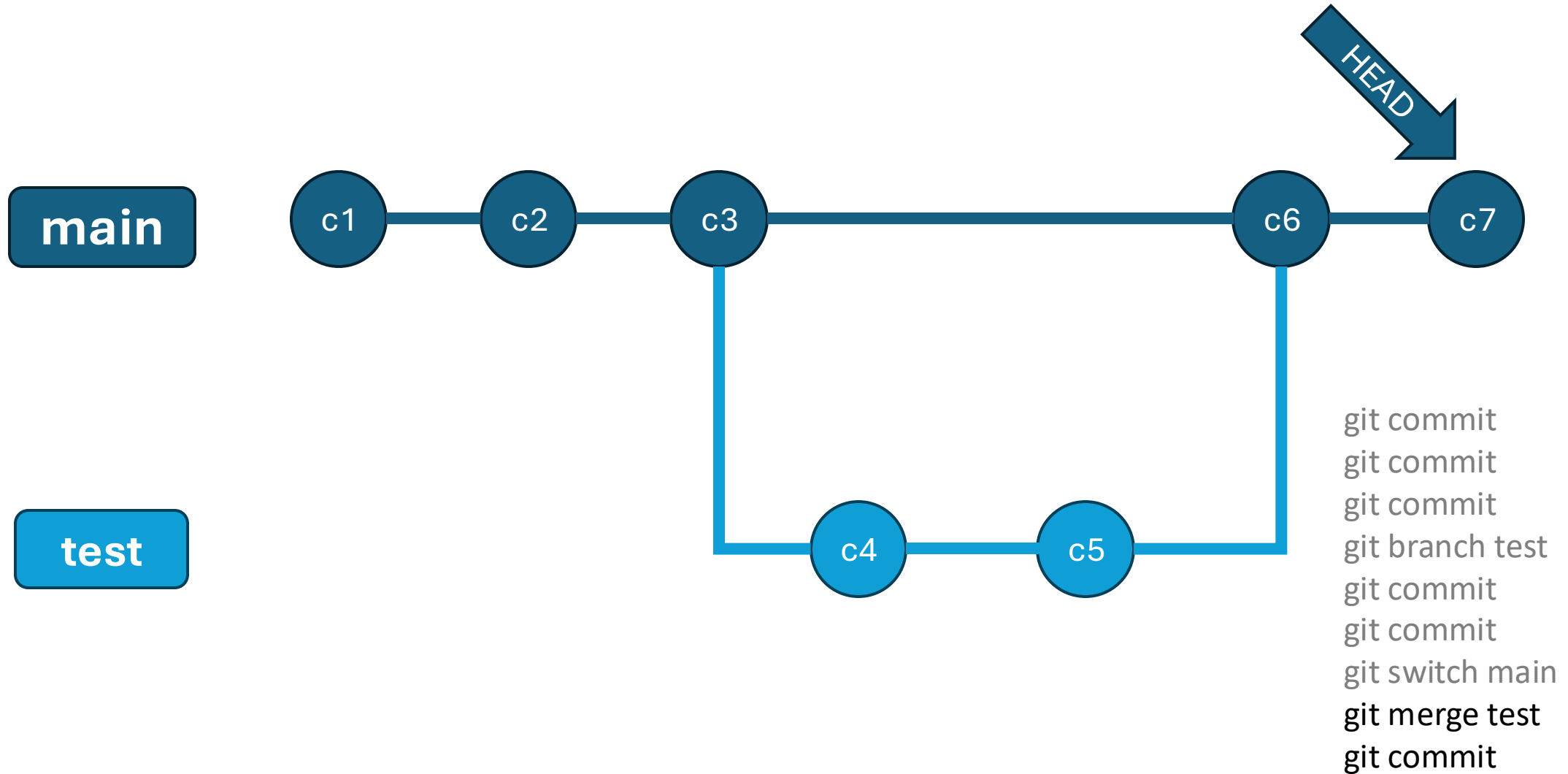
test



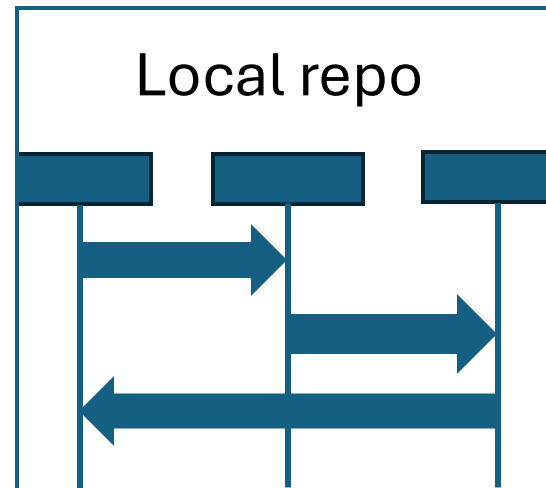
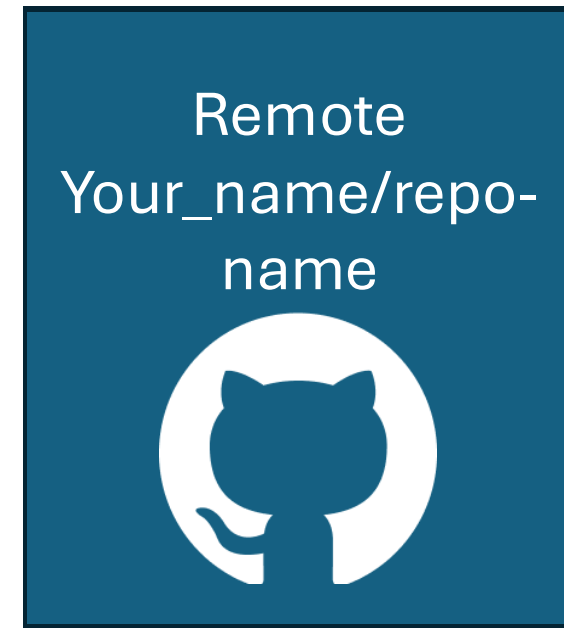
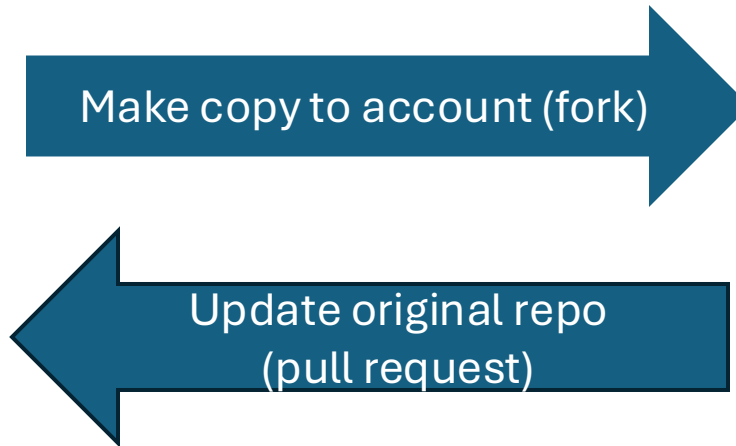
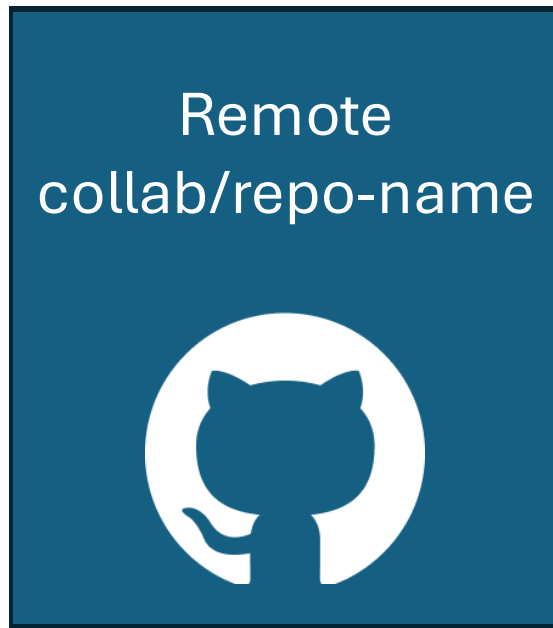
git commit
git commit
git commit
git branch test
git commit
git commit



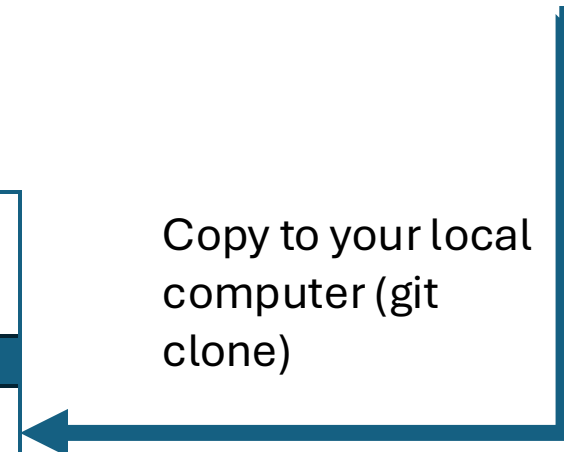
git commit
git commit
git commit
git branch test
git commit
git commit
git switch main

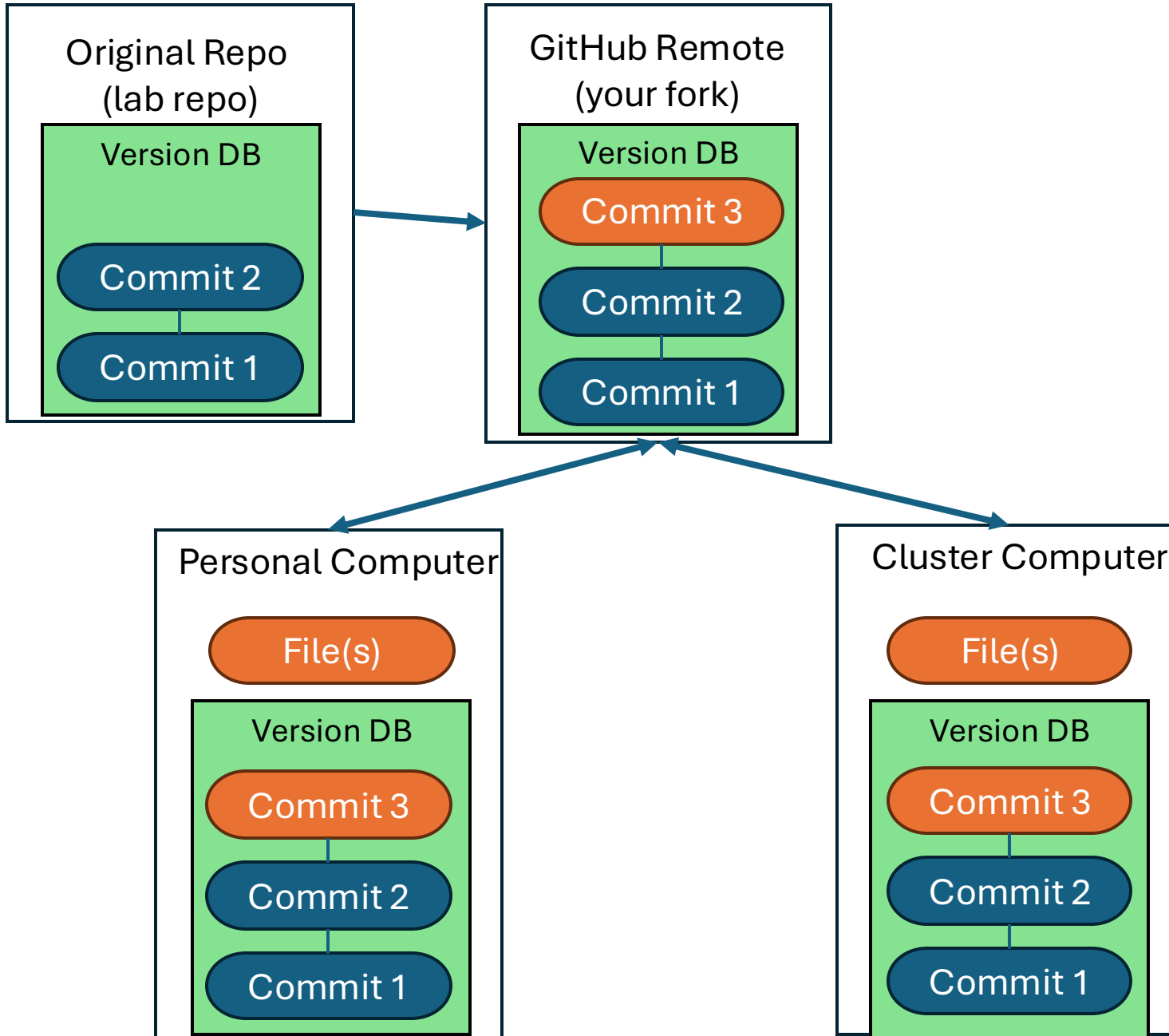


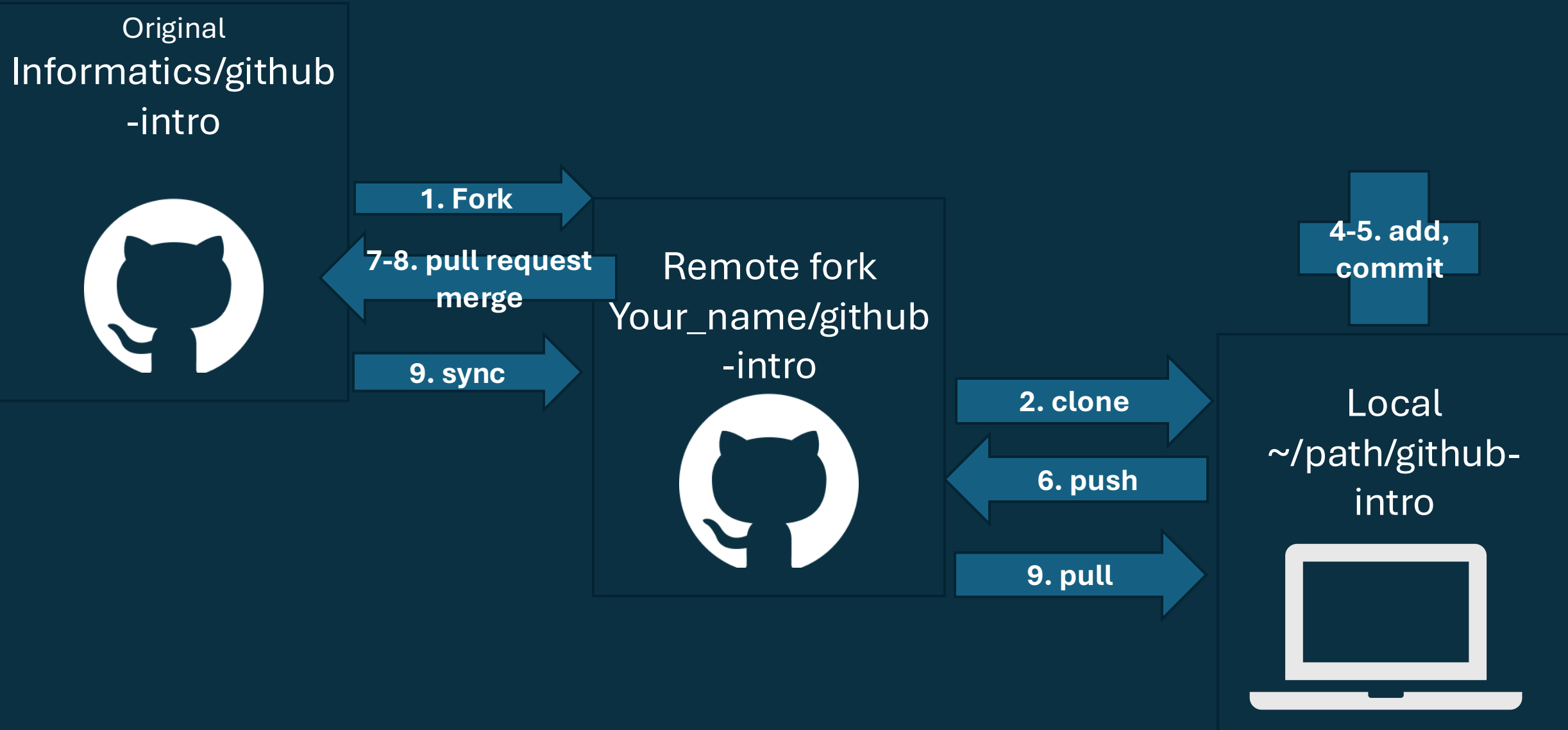
collaboration (forking and pull requests)



Copy to your local
computer (git
clone)







<https://github.com/harvardinformatics/github-intro>

GitHub Pages: a website for your repo

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

General

Access

Collaborators

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

Build and deployment

Source

Deploy from a branch ▾

Branch

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

main ▾ / (root) ▾ Save

Visibility [GitHub Enterprise](#)

With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise. You can try GitHub Enterprise risk-free for 30 days. [Learn more about the visibility of your GitHub Pages](#)

Website walkthrough

- Your repo will be found at `www.github.com/Your_name/repo-name`
- <> Code is your landing page
- ⚙ Settings for repo-specific preferences
 - go to “Danger Zone” to change visibility
- README.md will render on your landing page

GitHub Organizations

Create a lab organization

- Your data belongs to Harvard, which needs to archive for 7 years
- Transfer ownership of your repository to your lab
- Easier to pass on project for others to manage
- Example: <https://github.com/schlosslab>