Intro: Files, BASH and Git

Ádám T. Kocsis

Friedrich-Alexander-Universität Erlangen-Nürnberg

2026-08-17

GeoZentrum Nordbayern

Why?

and GitHub

Paleontological data in the 21st century

We have gone a long way…



Collectors only

-1960s



Pioneers

1960-1990/2000

The Paleobiology Database

revealing the history of life

NEOTOMADB.ORG

Neptune SANDBOX Berlin

Community of database-based research

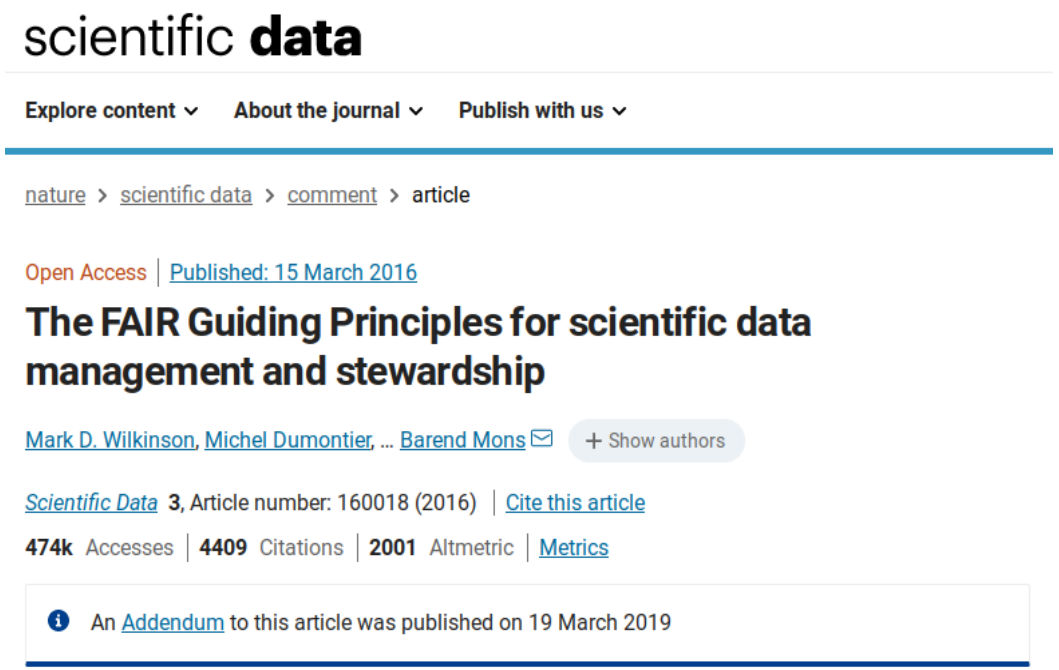
Being FAIR

A standard way to publish data and dat-based research.

- **F**indable
- **A**ccessible
- **I**nteroperable
- **R**eproducible

www.go-fair.org




scientific **data**

Explore content | About the journal | Publish with us

nature › scientific data › comment › article

Open Access | Published: 15 March 2016

The FAIR Guiding Principles for scientific data management and stewardship

Mark D. Wilkinson, Michel Dumontier, ... Barend Mons + Show authors

Scientific Data 3, Article number: 160018 (2016) | Cite this article

474k Accesses | 4409 Citations | 2001 Altmetric | Metrics

An Addendum to this article was published on 19 March 2019

Reproducibility

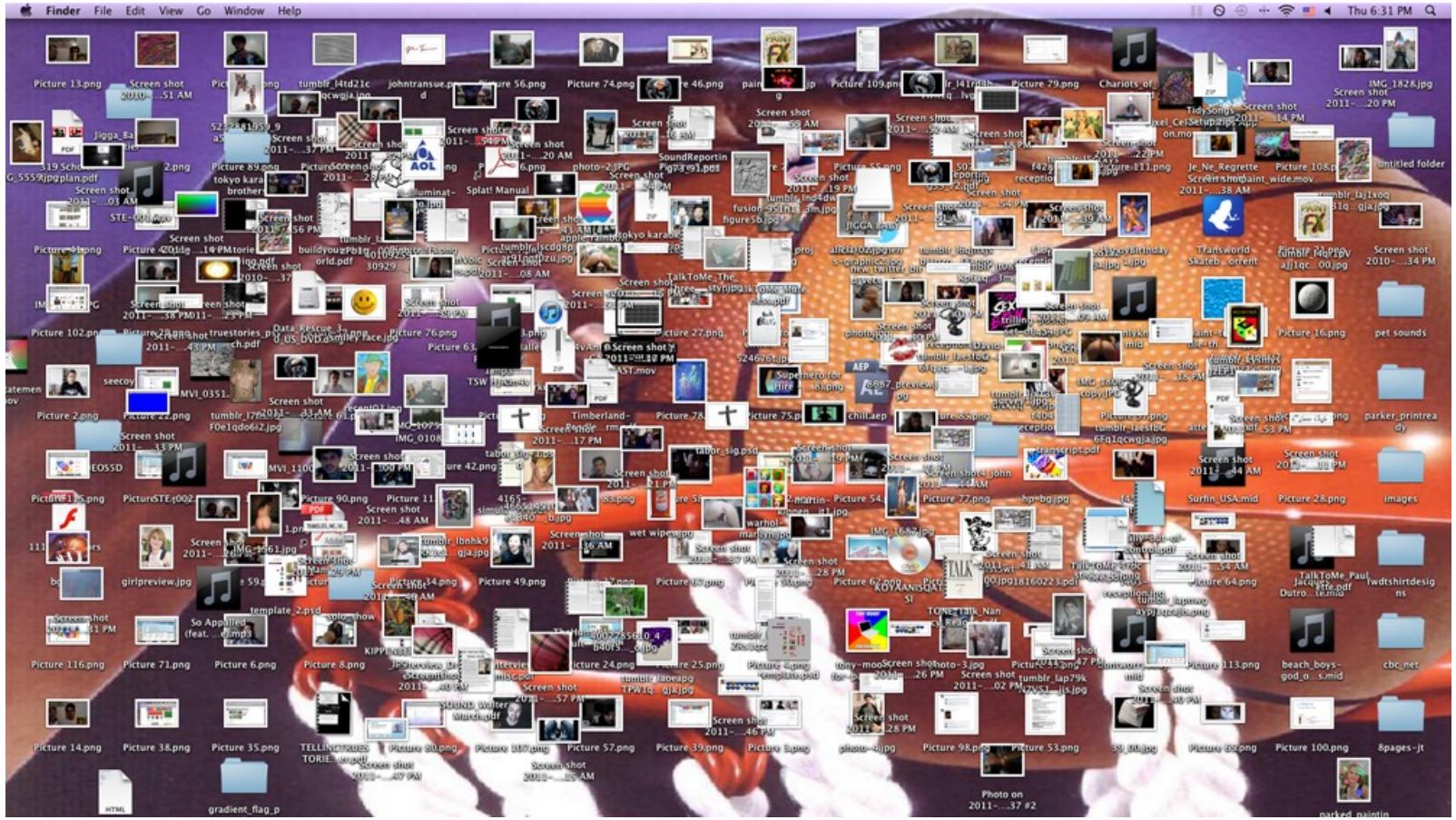
The foundation of the scientific experiment

- Can you reproduce the exact results that you acquired 5 years ago?
- If you cannot reproduce what you have done, how can other people?

Source: The Turing Way: https://the-turing-way.netlify.app/

		Data	
		Same	Different
Analysis	Same	Reproducible	Replicable
	Different	Robust	Generalisable

Avoid this at all costs…



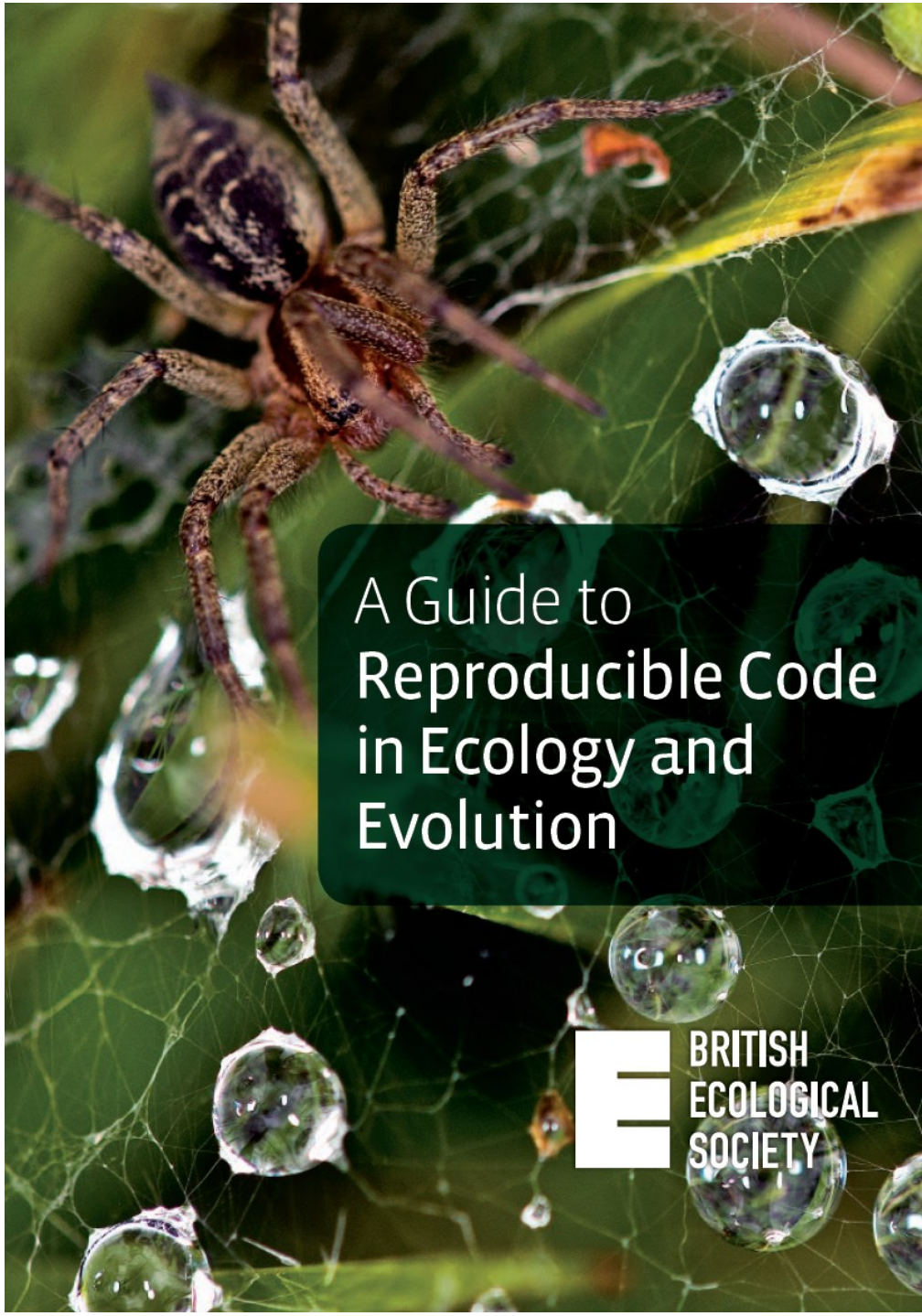
Do not keep things on your desktop!

Overall file management

Suggestions

- Keep all your stuff together (separate partition!)
- Logical hierarchy
- Make it portable (Windows!)
- Regularly spend time on organizing and cleaning files
- Naming and grouping: self-explanatory – make it for somebody else (you!)
- Try to avoid spaces in paths
- Cloud backups!

Reproducibility is your main goal!

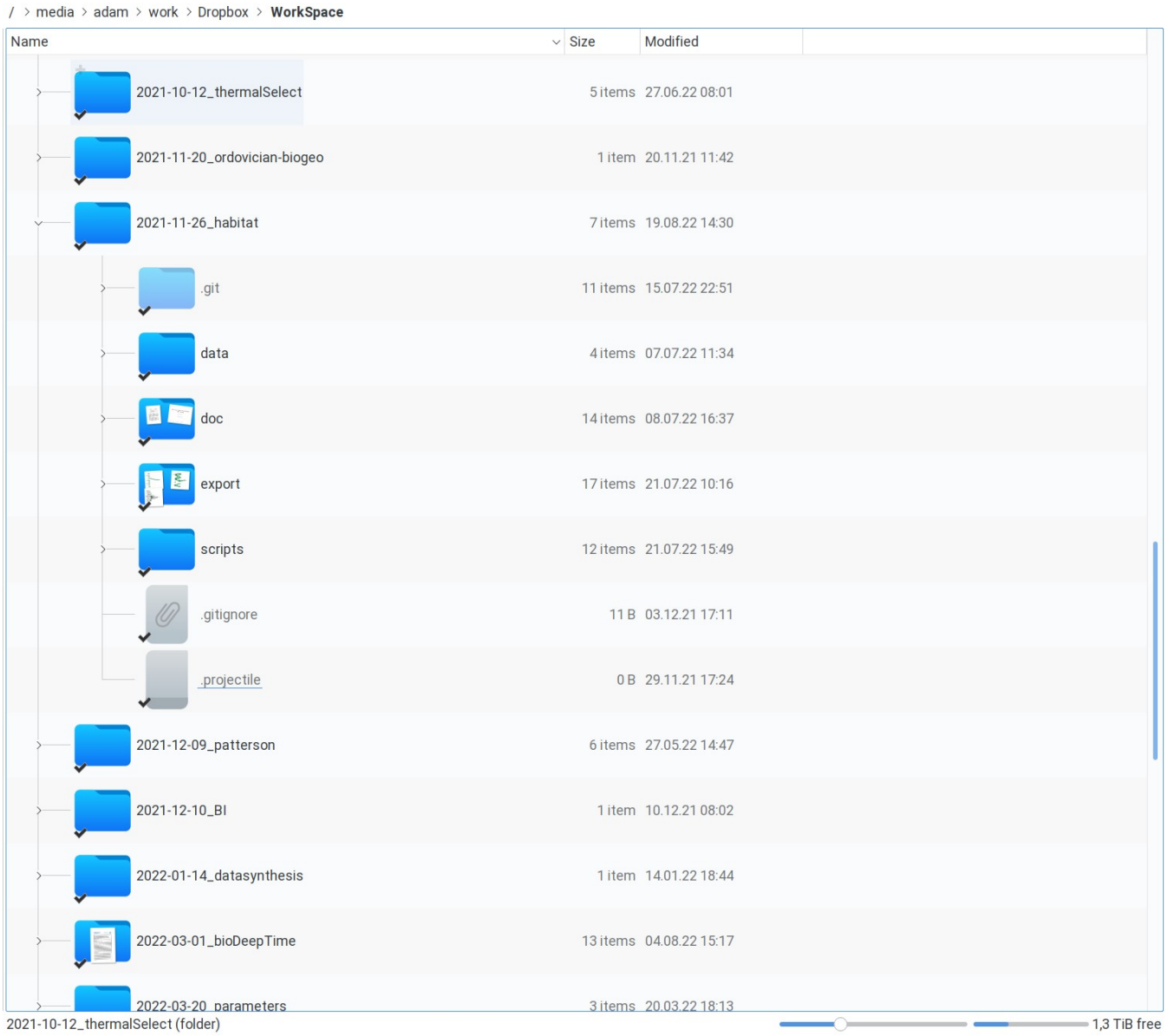


Suggestions

Keep all your projects separate!

Use the same project structure:

- Input Data (data)
- Computer code (code/scripts)
- Written documents (doc)
- Calculation output (export/output)



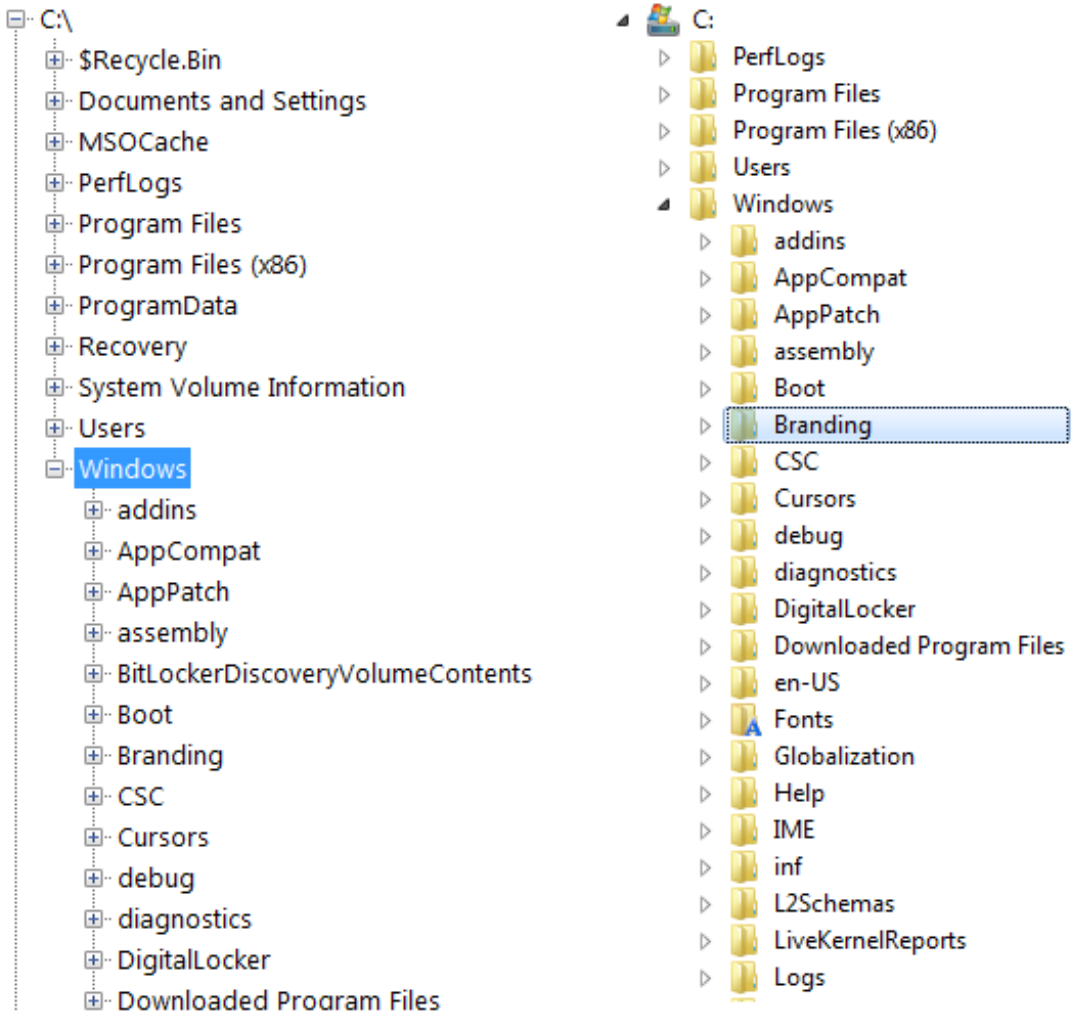
About files…

The Windows file system

- Files are data items on storage devices
- Paths use the characteristic backslash \ character to depict nestedness
- Directories are called “Folders”
- File format: filename.ext
- Total path to “Branding”:

```
C:\Windows\Branding
```

- Case insensitive!

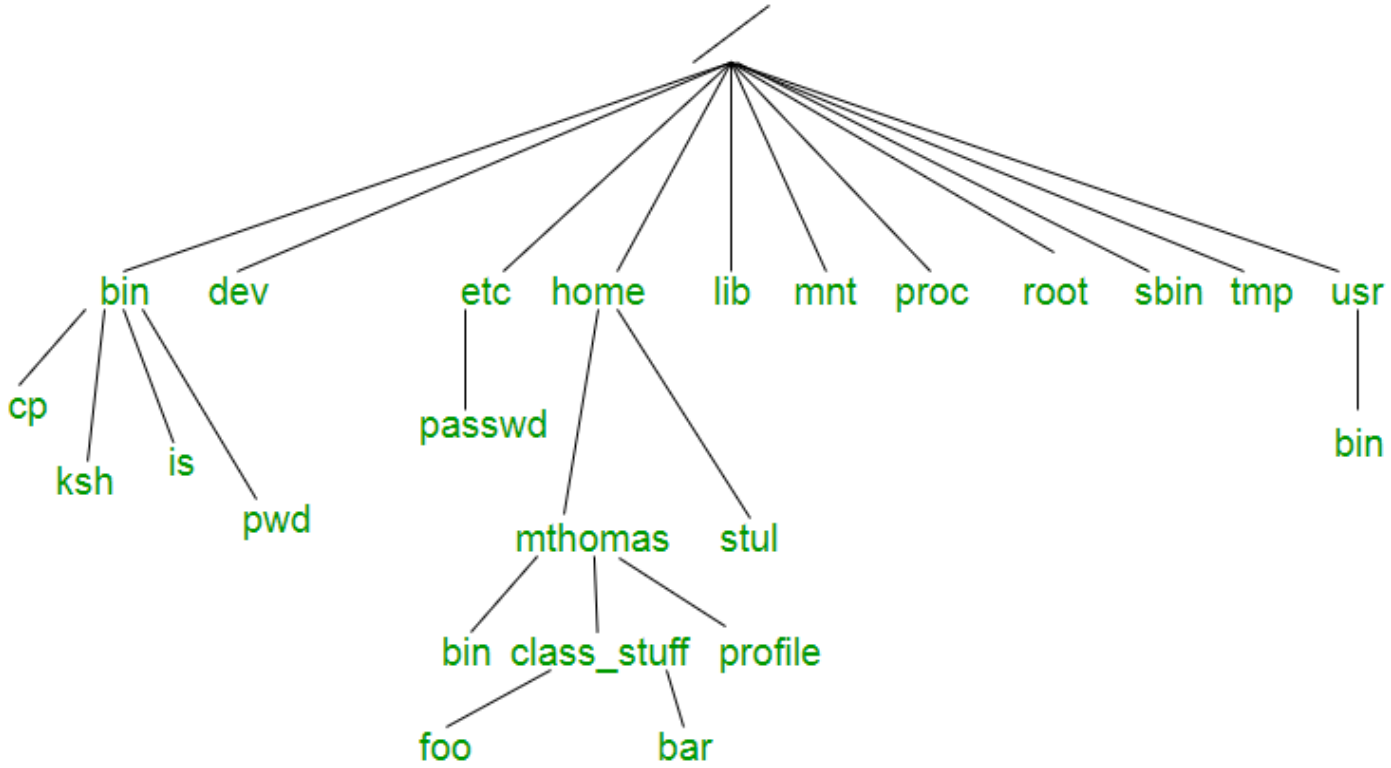


The UNIX file system

- Shared for UNIX and UNIX-like systems (GNU/Linux, macOS, Android)
- Concept: everything in the computer is represented by a file
- Nestedness coded with forward slash : /
- File format can be anything
- Complete path to “bar”

```
/home/mthomas/class_stuff/bar
```

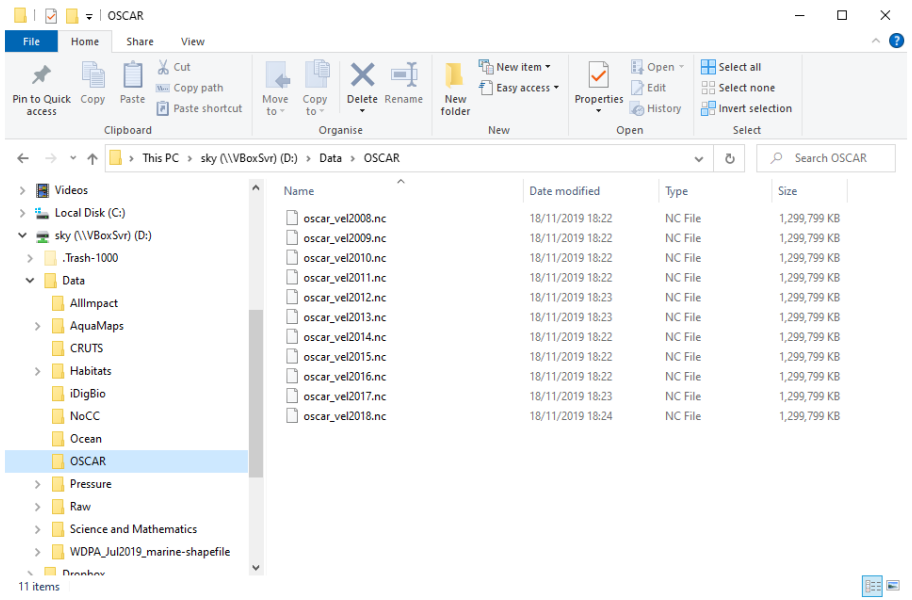
- Case sensitive!



Two main options:

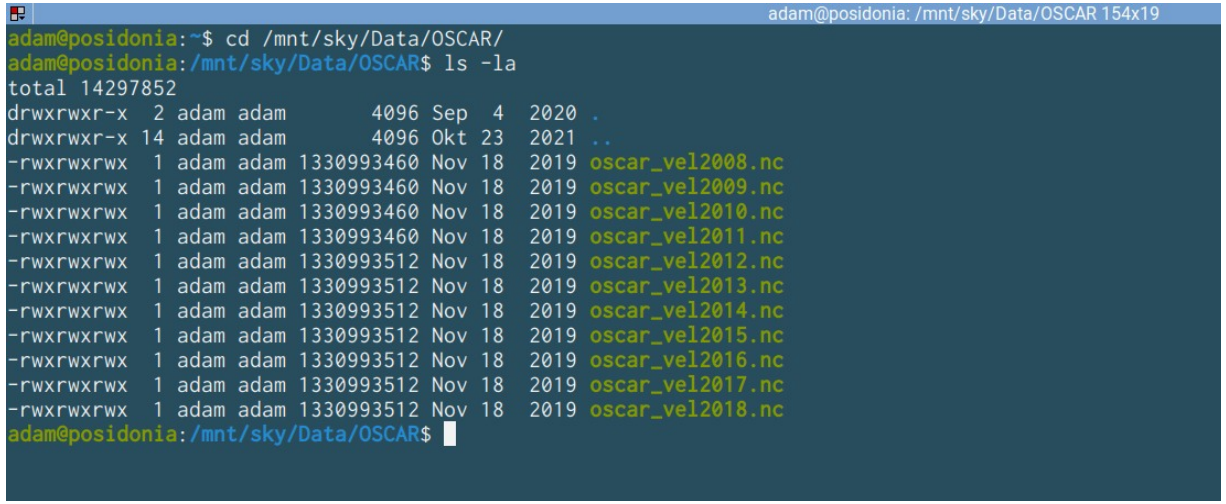
Graphical User Interface (GUI)

- Super simple + mouse
- Visually appealing
- “Novice-friendly”



Command Line Interpreter (CLI)

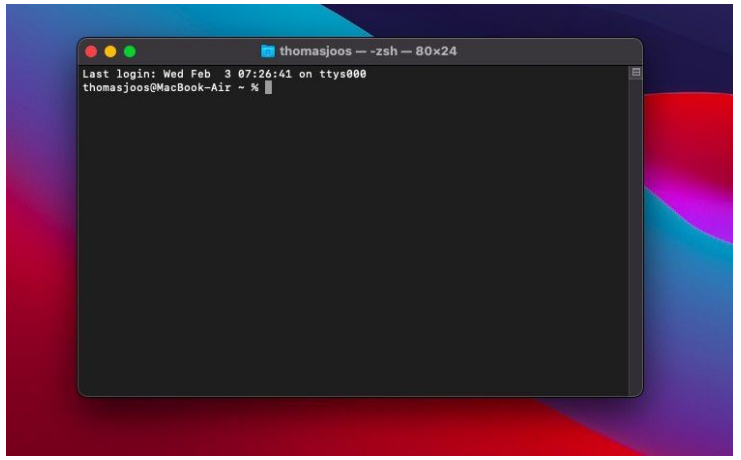
- Steeper learning curve
- Automation
- Keyboard-only “Expert-friendly”



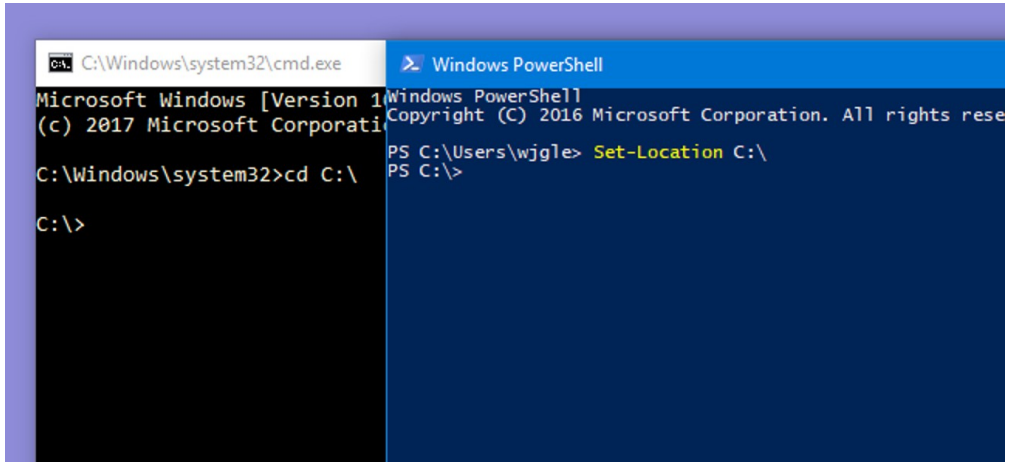
Terminal emulators

- Every OS has one
- Graphical applications that run a program called the “shell”: an interpreter program that translates instructions
- Console applications can be run with the shell
  - Automation
  - Program building
  - Scientific calculations
- Shells are programmable

Mac (zsh or bash)



Windows (cmd and powershell)



The BASH shell

- Ubiquitous
- Most frequently used on servers and clusters
- UNIX-native: most programming systems use UNIX-like paths – even on Windows!
- Mac: have it, z shell (zsh) is almost the same
- Windows: a simplified version is available with git (git bash)



BASH

THE BOURNE-AGAIN SHELL



https://git-scm.com

Installing git for Windows

and GitHub

BASH essentials

Most important functions and browsing directories

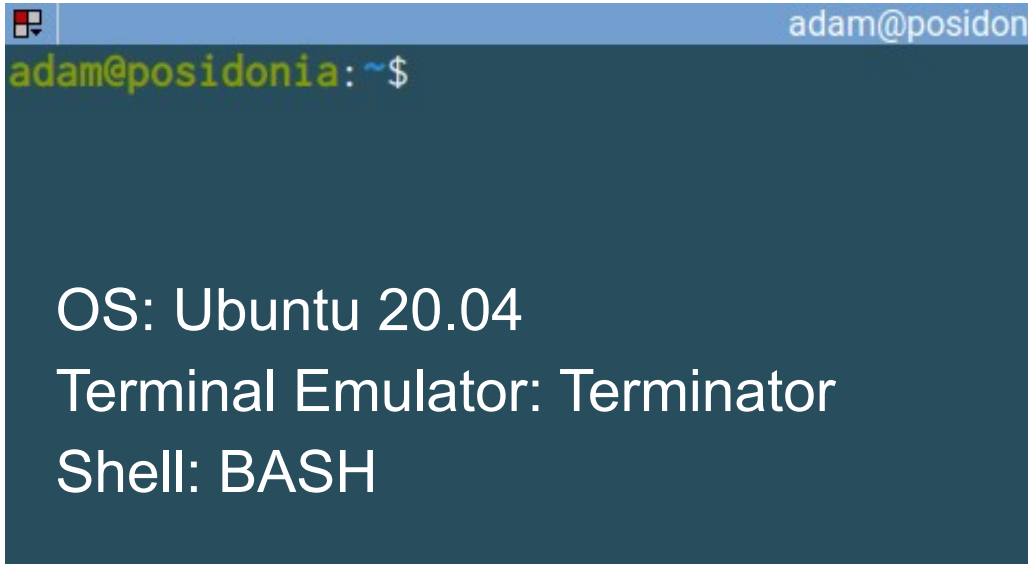
The prompt

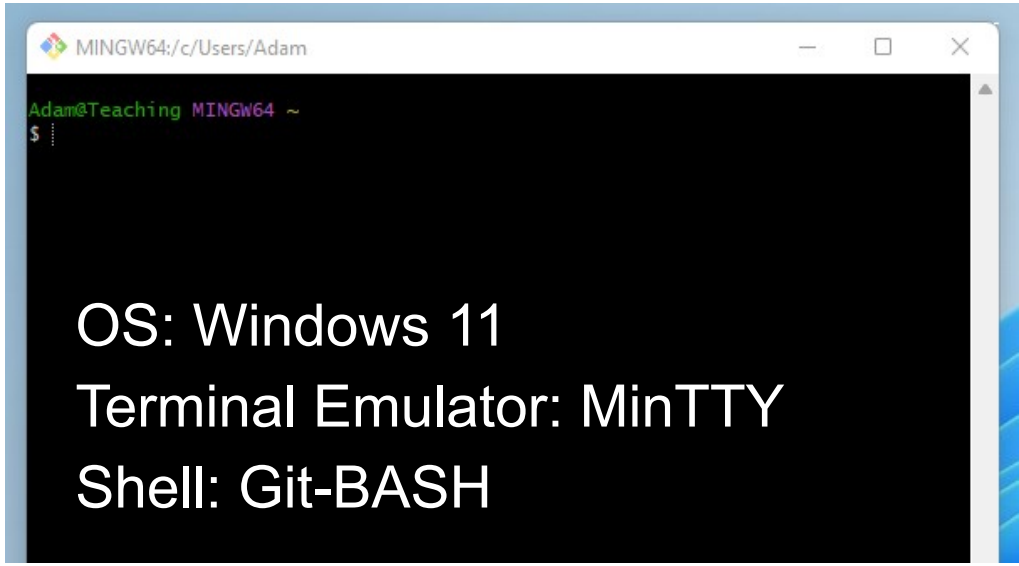
- User input expected (typing)
- Looks different on all, but there are conventions:

`user@host`

`~`: is shorthand for user home

`$`: means normal user mode





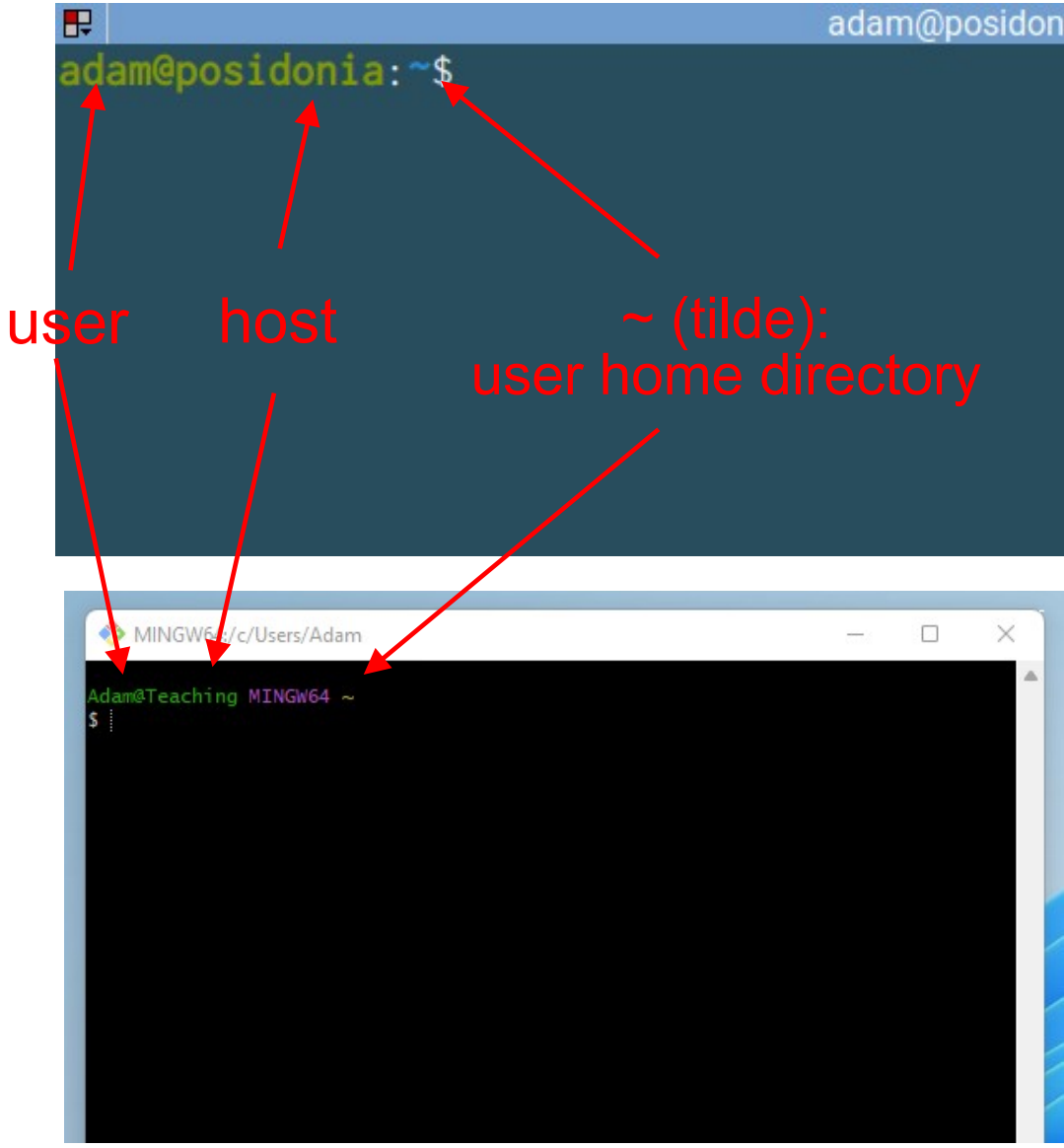
The prompt

- User input expected (typing)
- Looks different on all, but there are conventions:

`user@host`

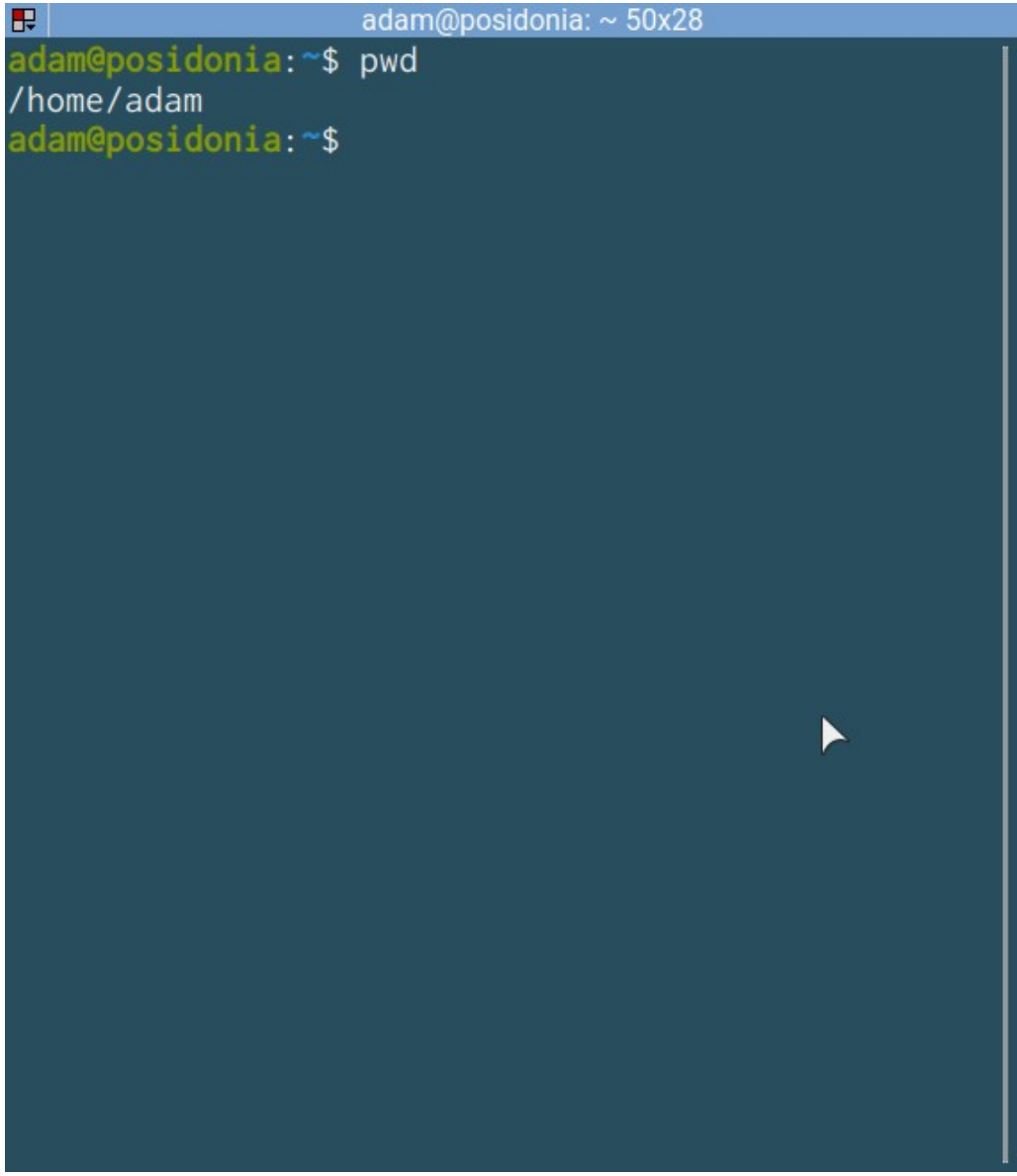
`~`: is shorthand for user home

`$`: means normal user mode



pwd

Return path to current directory

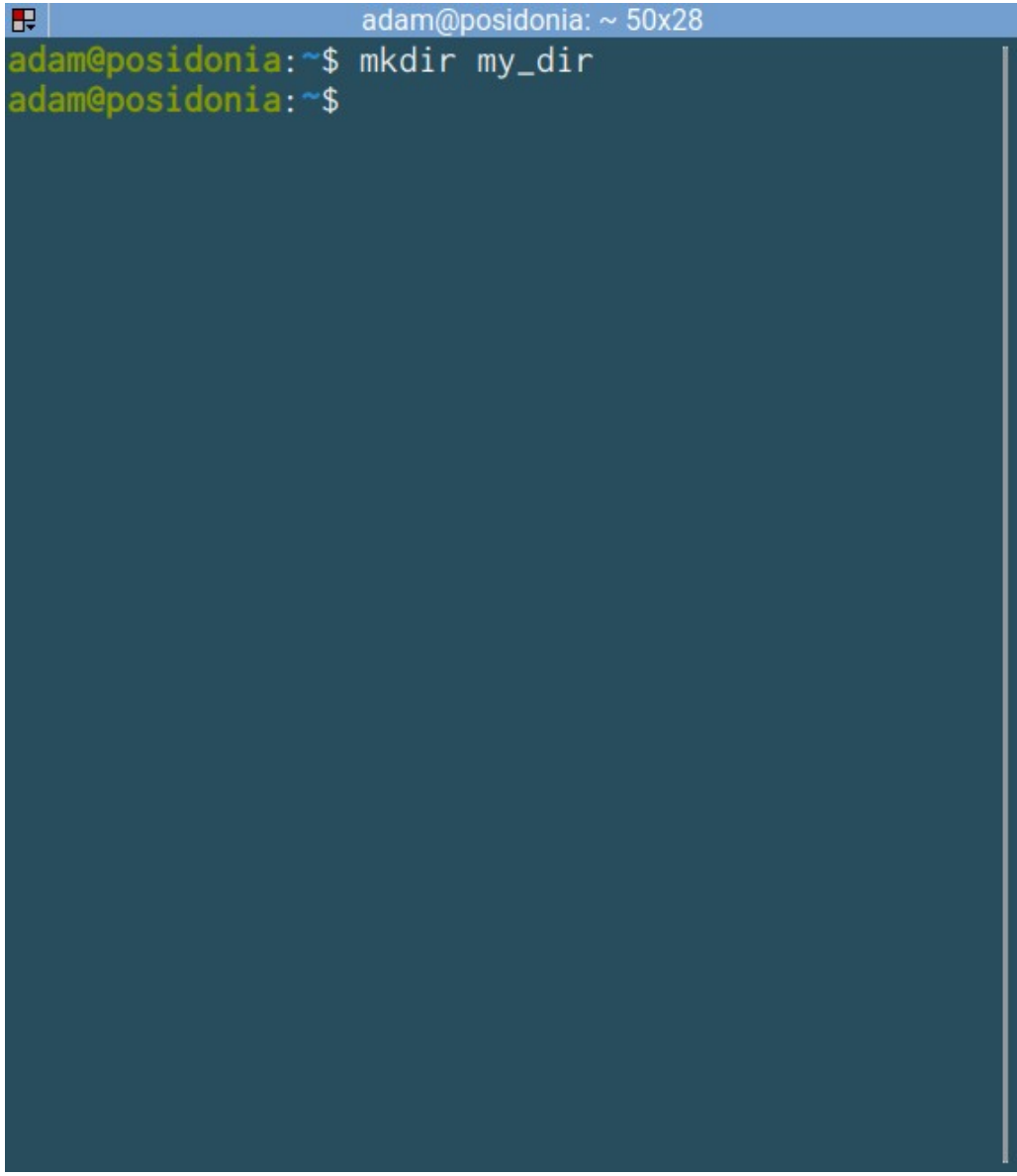


```
mkdir␣<name>
```

space

Create a directory

- No output to the console: no error occurred (directory was created)

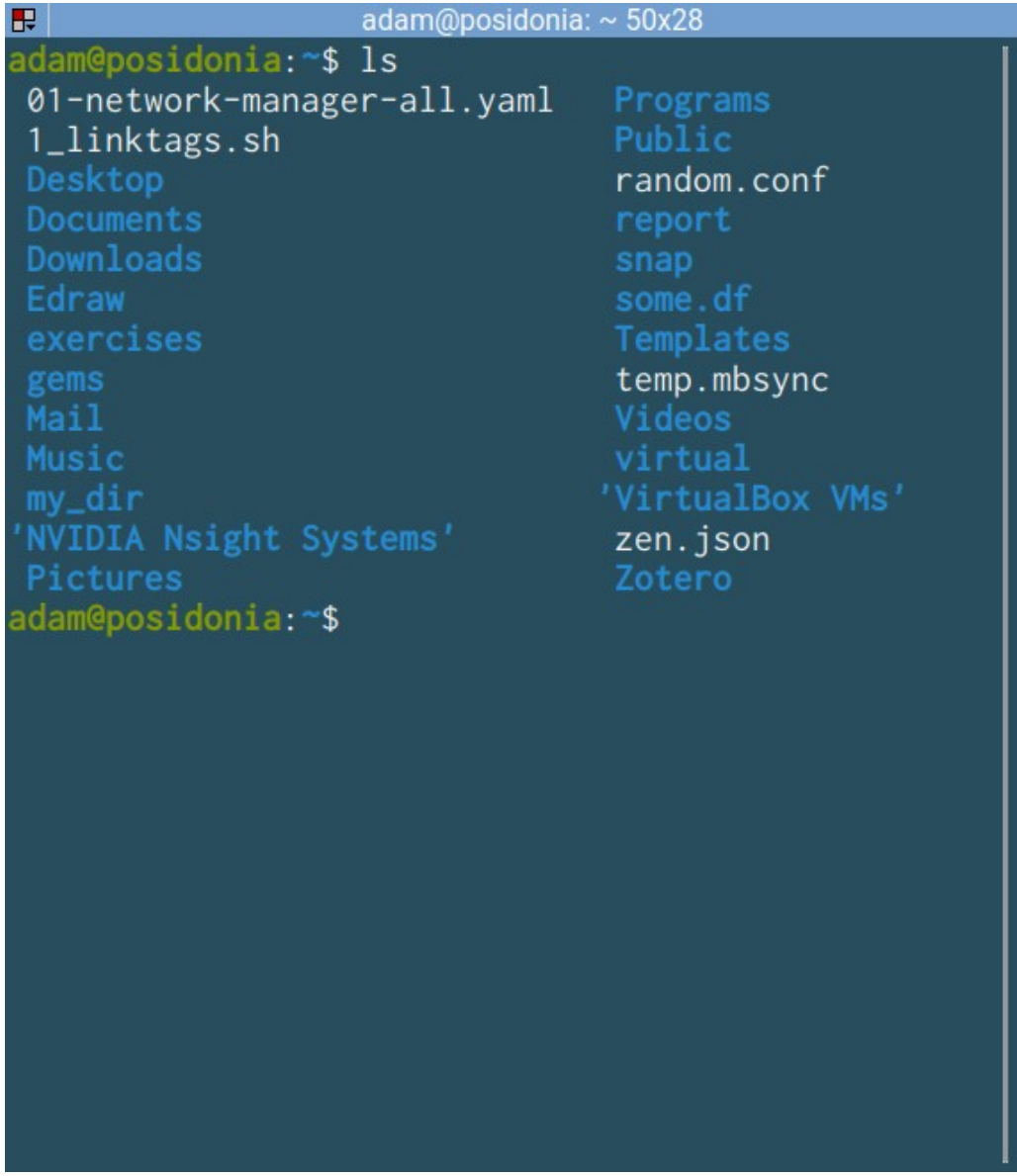


`ls`

List directory contents

- Returns a list of entries (both normal files and directories) – can be colored
- Note the quotes around entries with spaces in them!

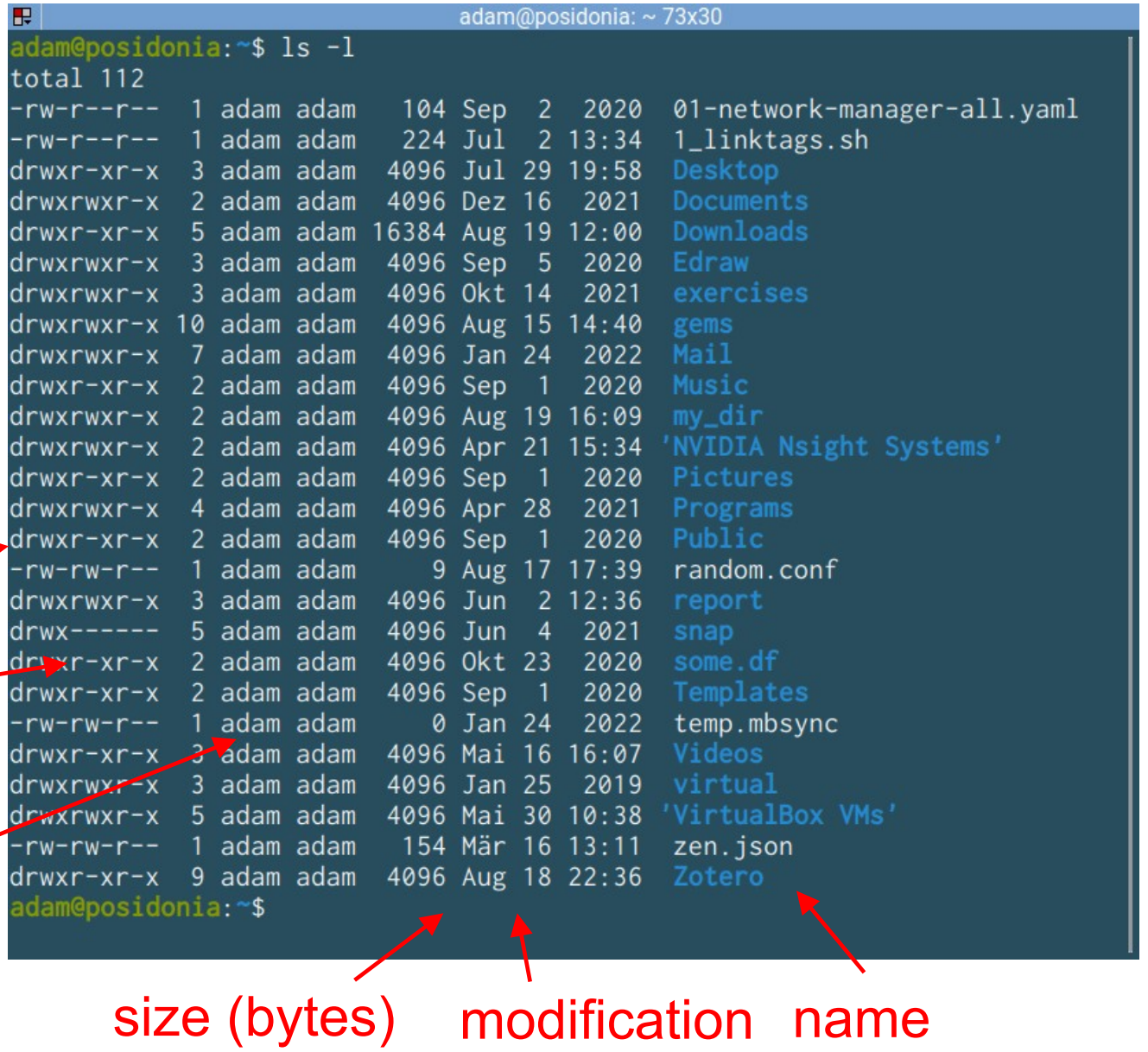




`ls -l`

List directory contents (with option l)

- Long output, includes attributes



`cd␣<path_to_directory>`

Go to a directory

- Can be relative or absolute!

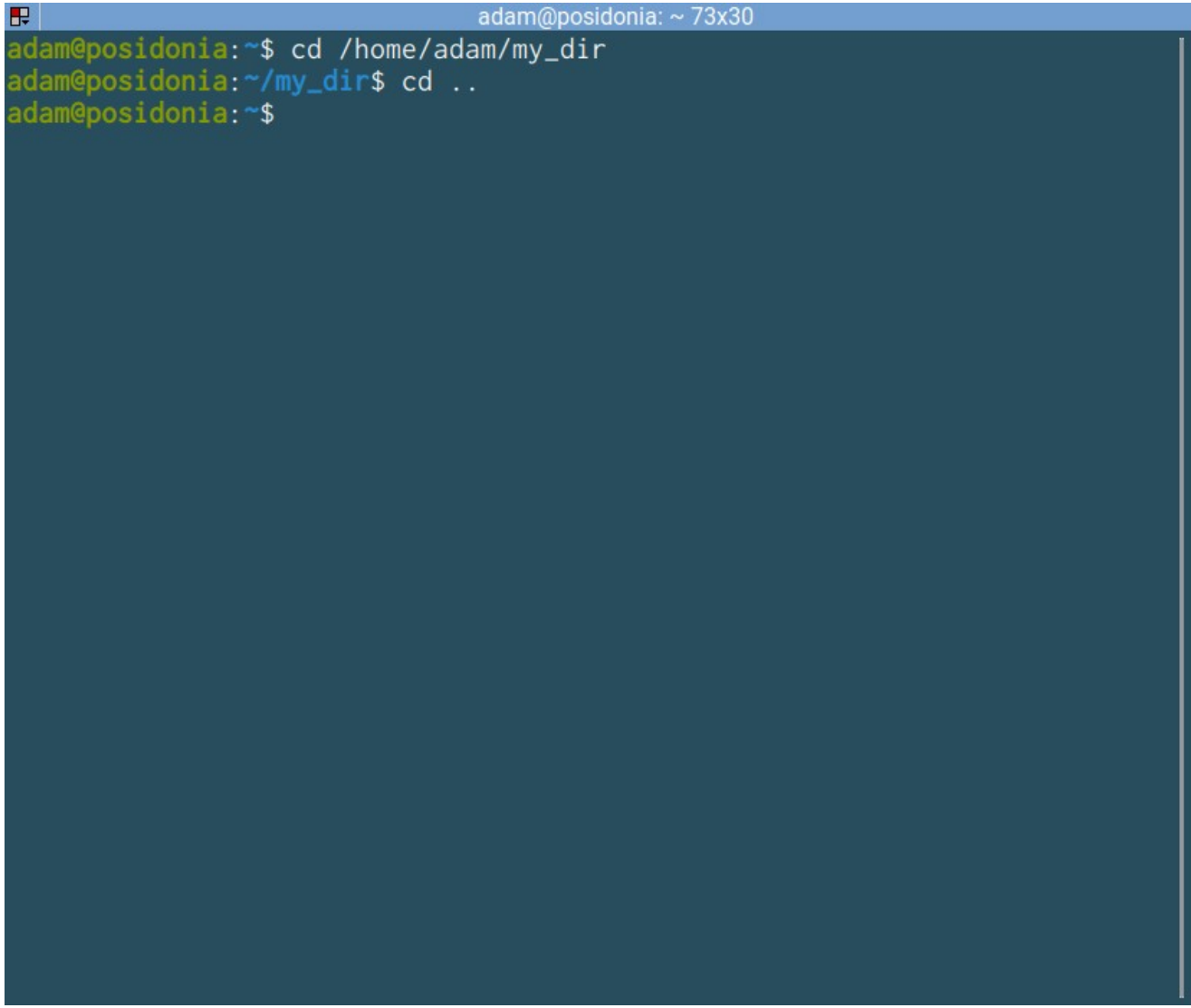




cd␣..

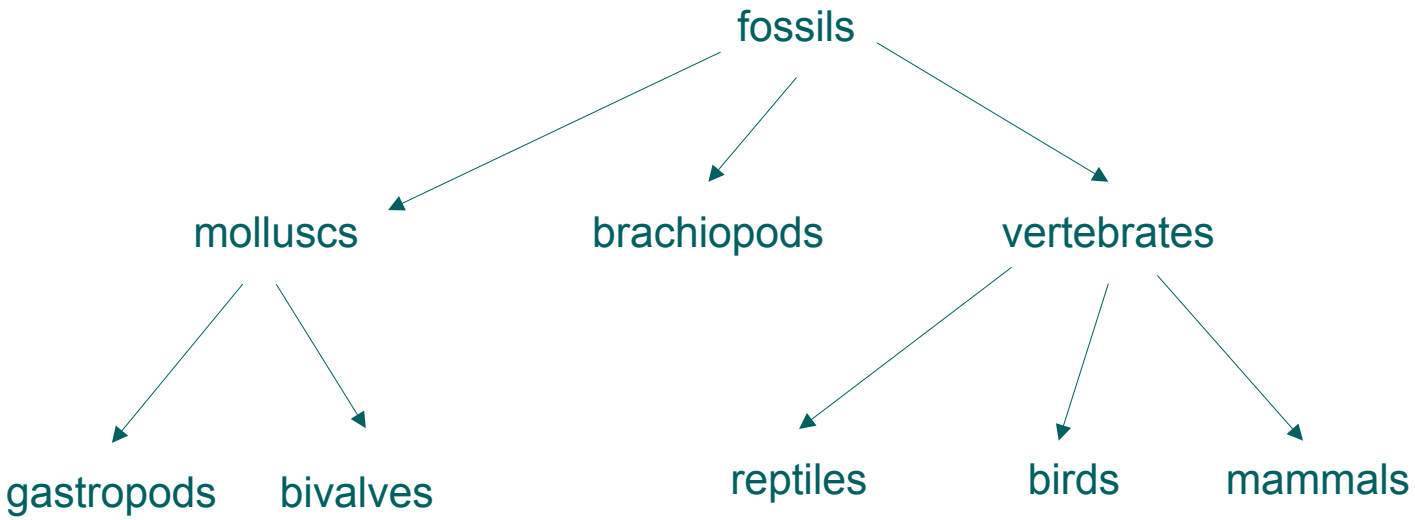
Go to parent directory

- .. (dot dot) is a placeholder for the parent of the current directory (one up in the hierarchy)



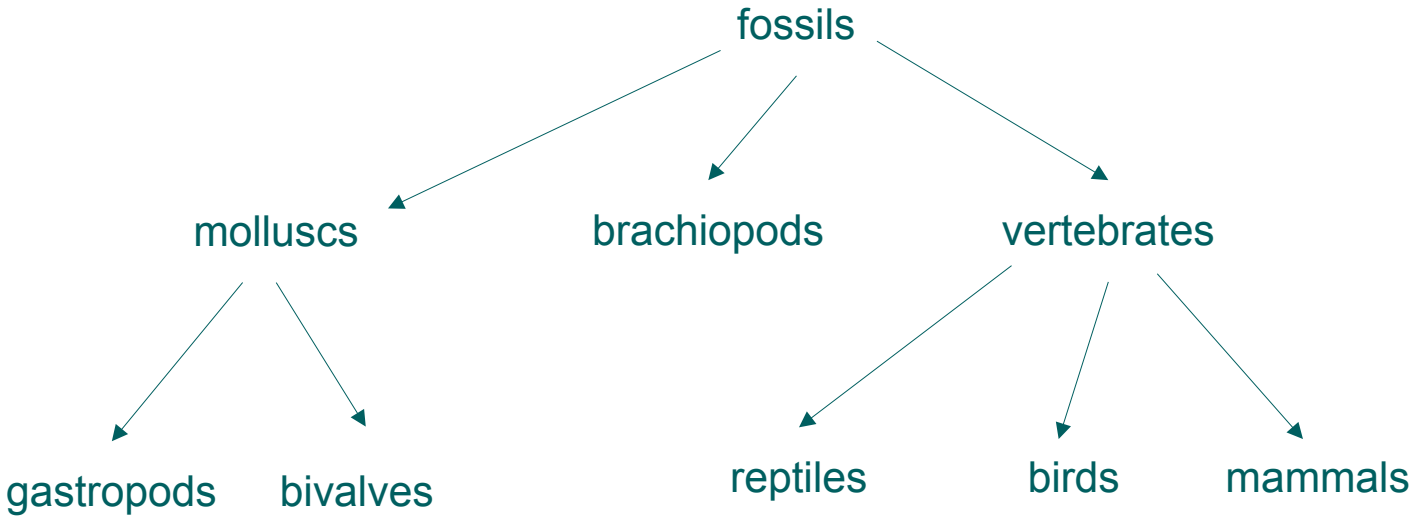
Exercise (5 minutes)

- Create this directory structure using the combinations of the previous commands!



Exercise (5 minutes)

- Create this directory structure using the combinations of the previous commands!

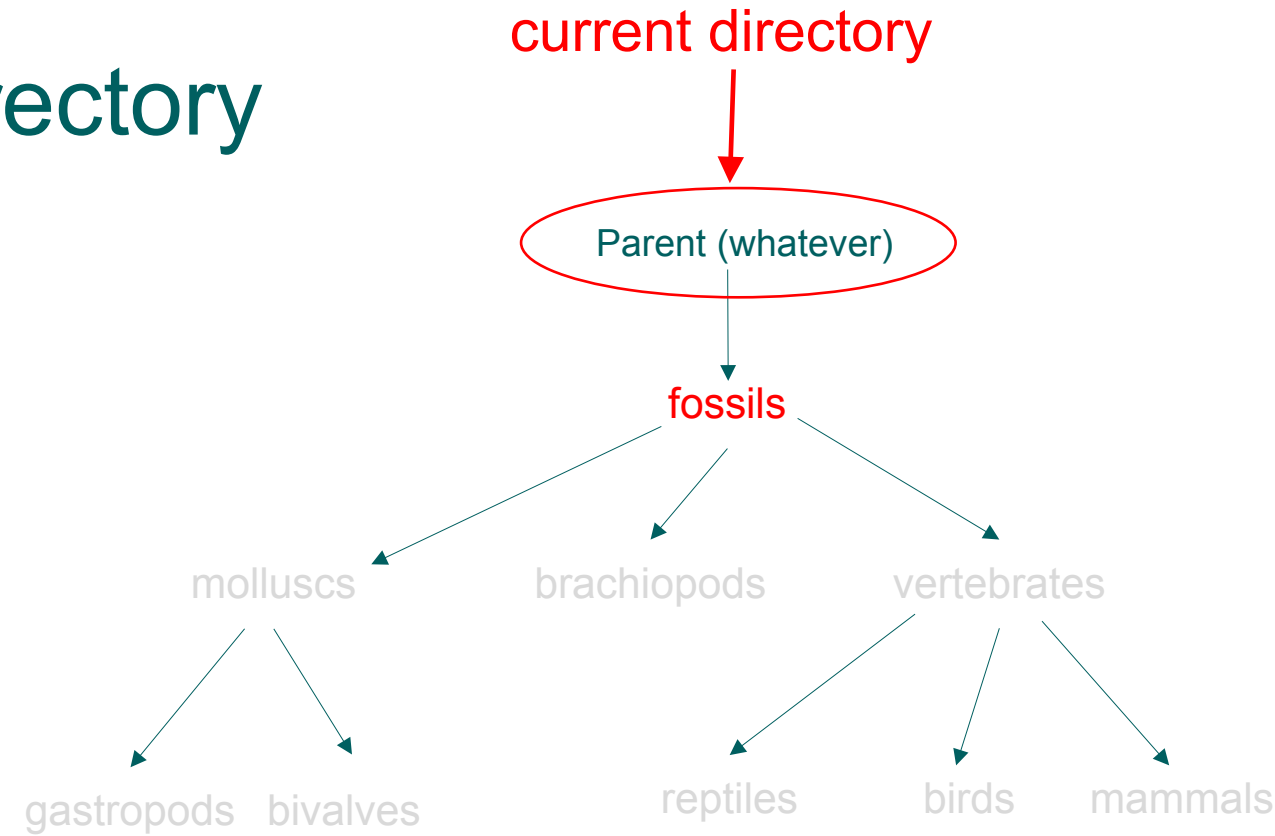


Hints

- use <TAB> completion (try <TAB> <TAB> to see multiple solutions)
- use <UP> and <DOWN> to search command history for already given commands

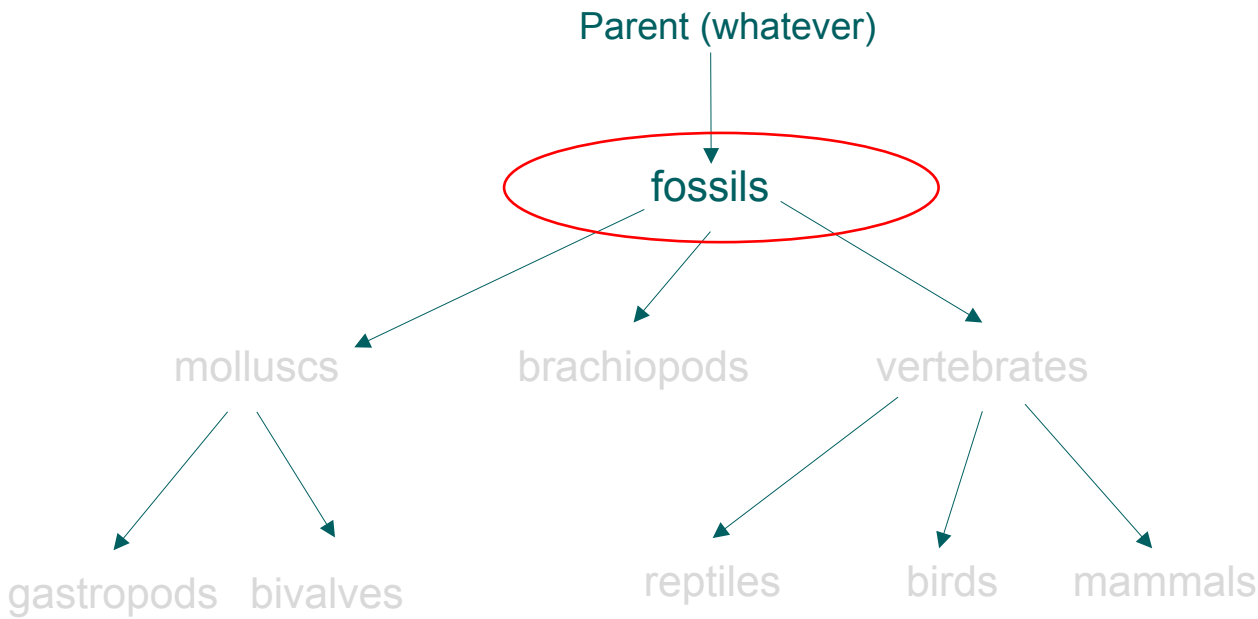
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



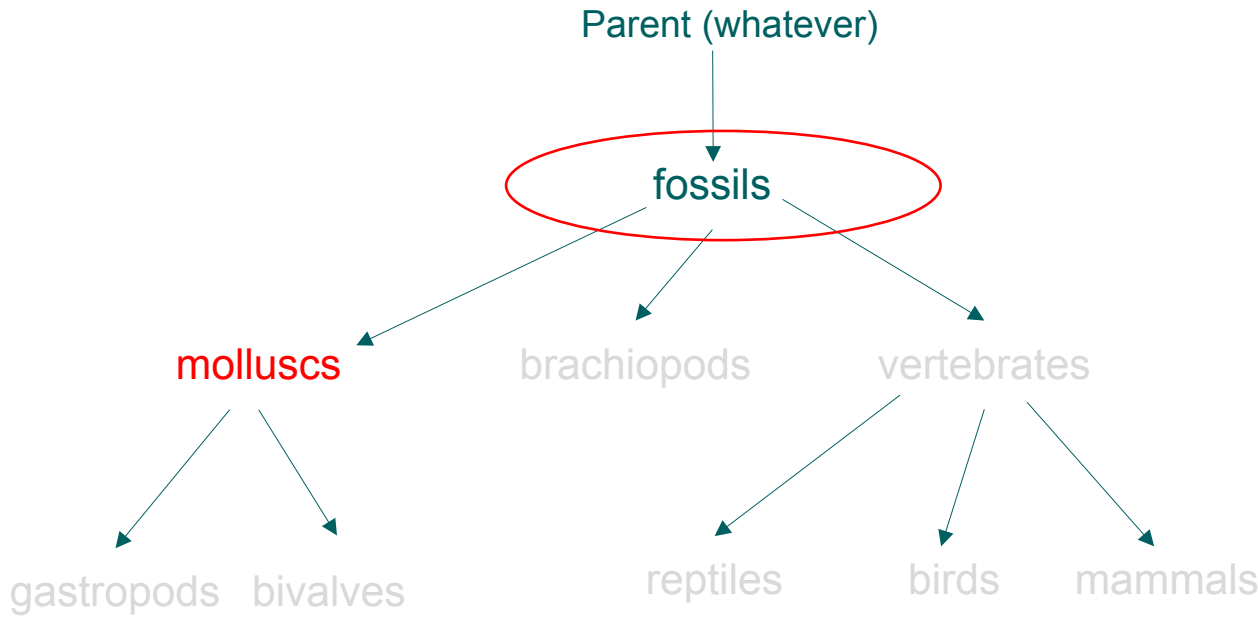
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



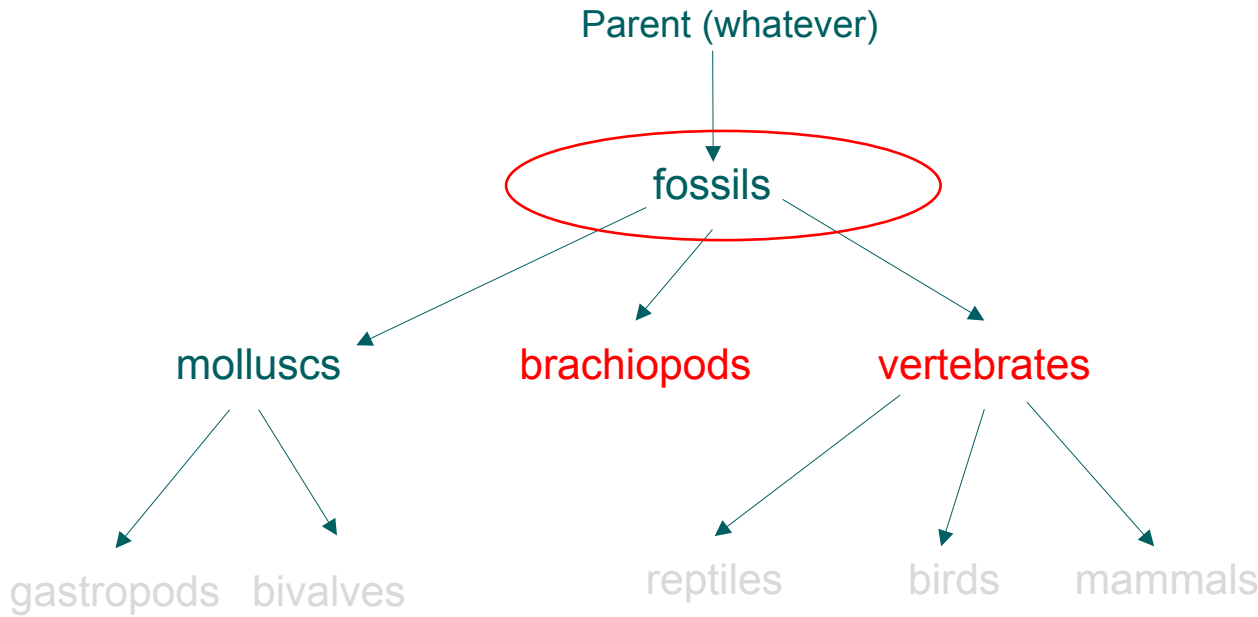
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



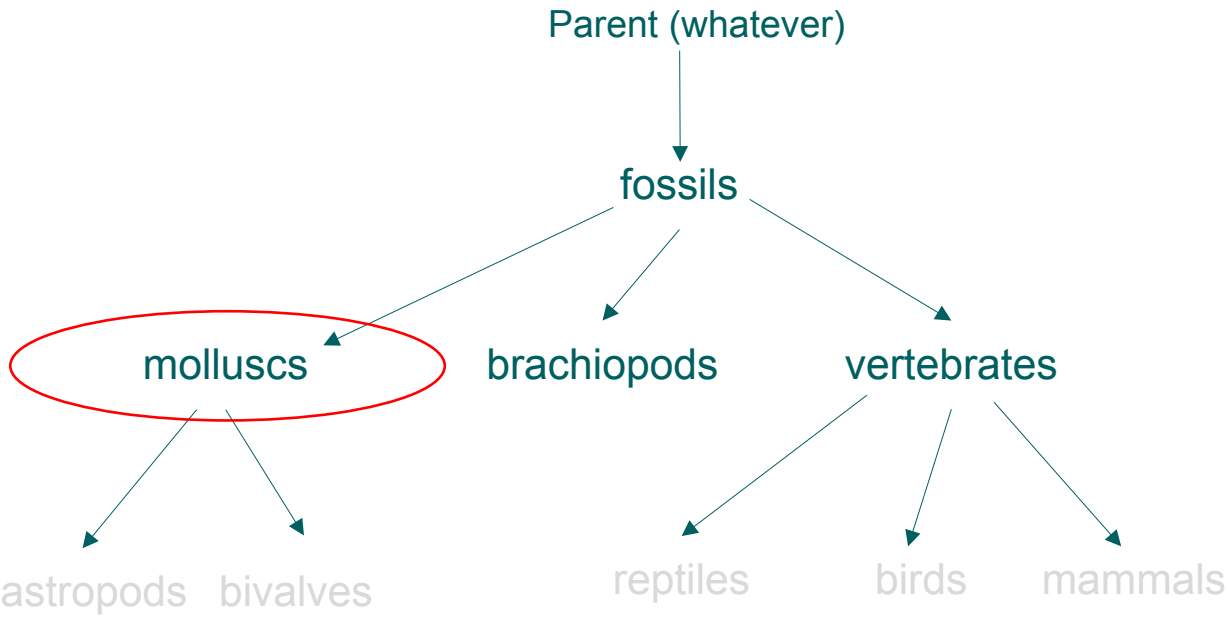
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



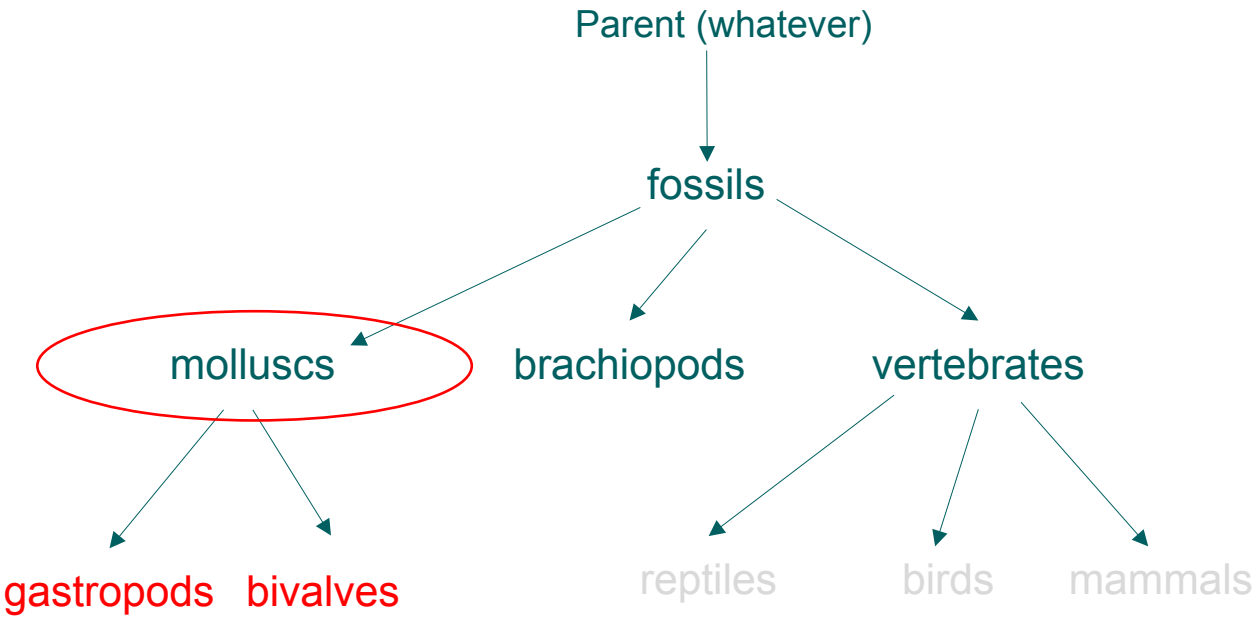
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



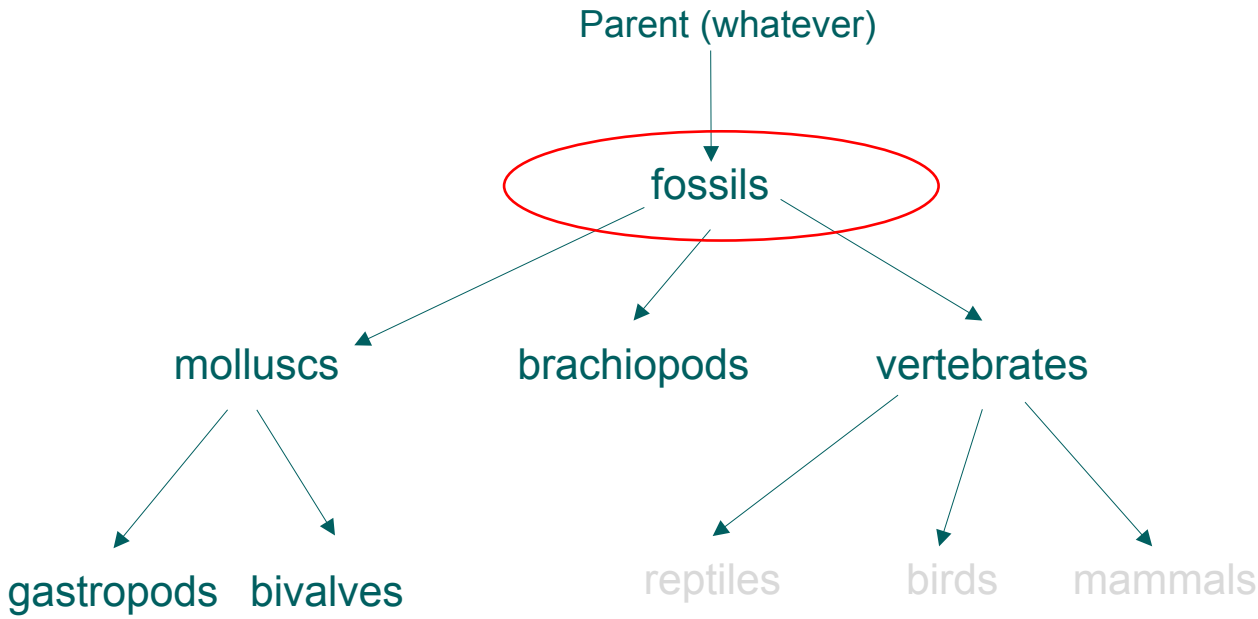
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



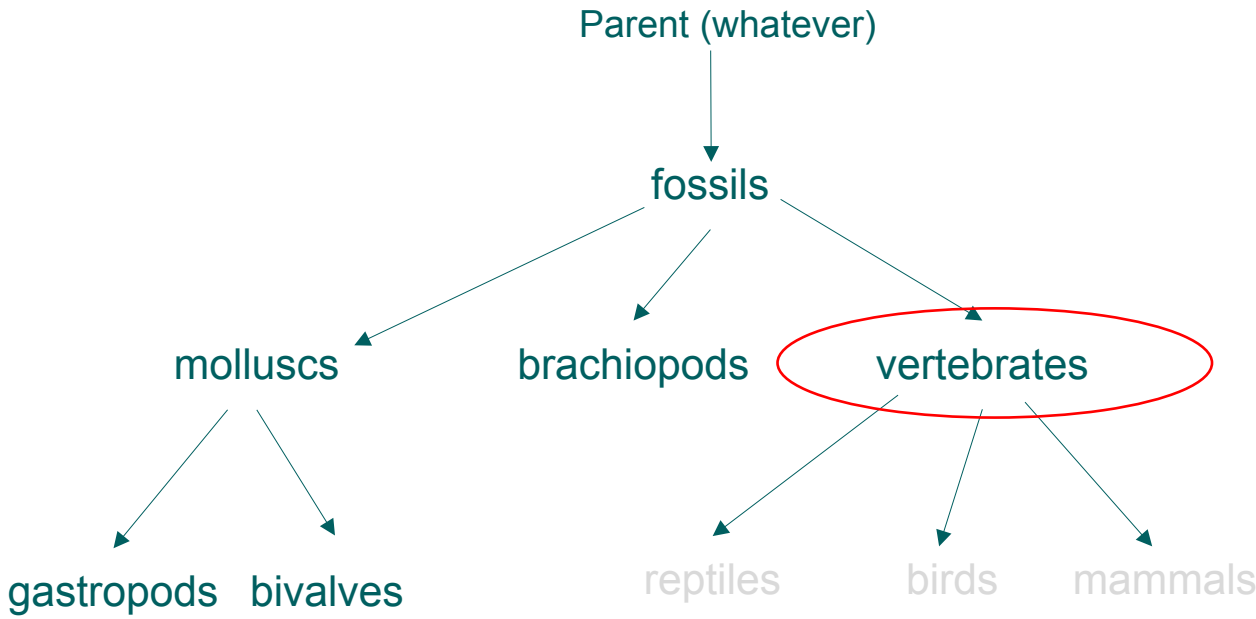
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



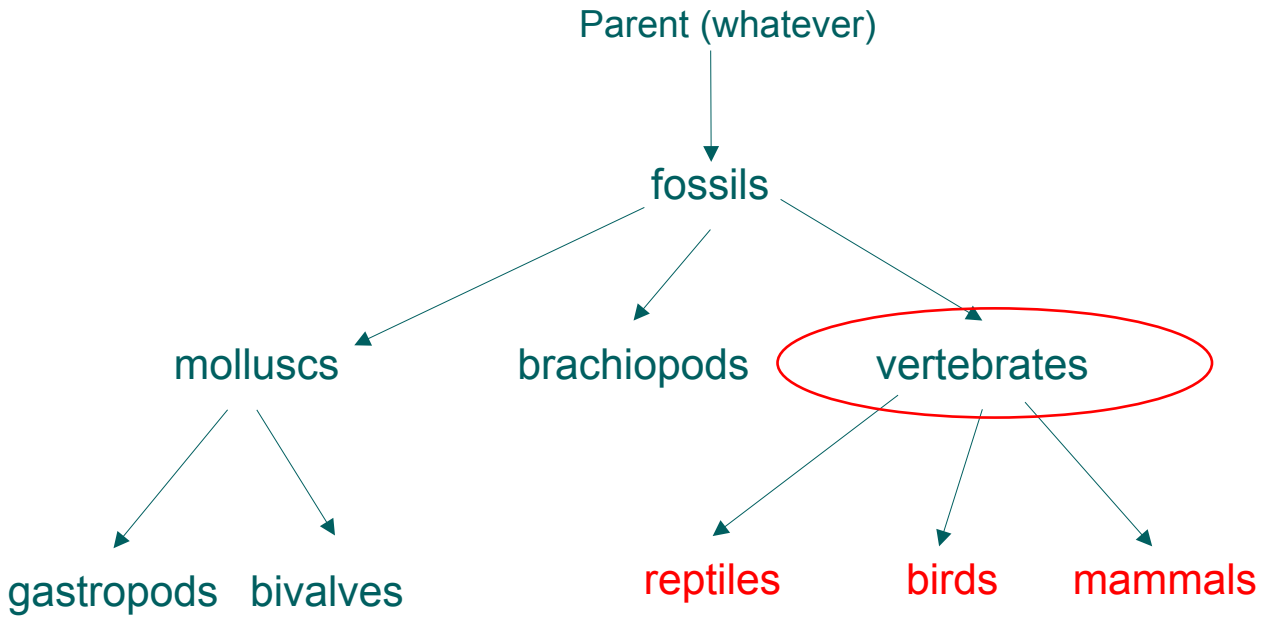
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



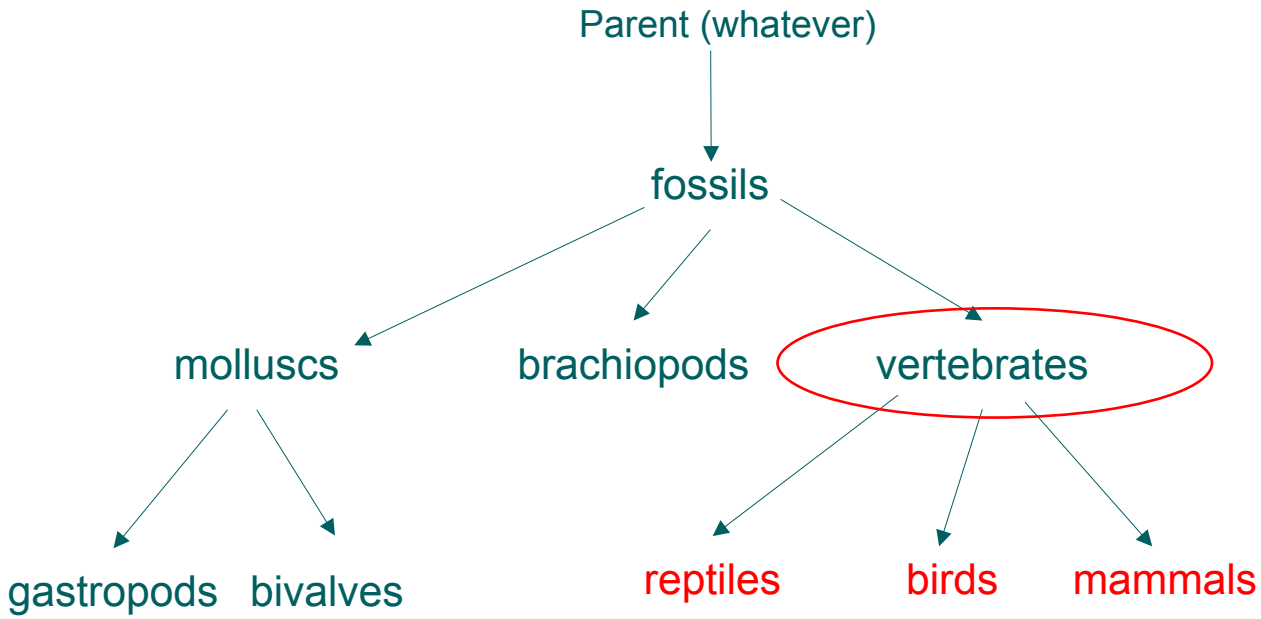
Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```



Solution 1 – changing directory

```
adam@posidonia: ~/fossils/vertebrates 73x30
adam@posidonia:~$ mkdir fossils
adam@posidonia:~$ cd fossils
adam@posidonia:~/fossils$ mkdir molluscs
adam@posidonia:~/fossils$ mkdir brachiopods vertebrates
adam@posidonia:~/fossils$ cd molluscs
adam@posidonia:~/fossils/molluscs$ mkdir gastropos bivalves
adam@posidonia:~/fossils/molluscs$ cd ..
adam@posidonia:~/fossils$ cd vertebrates
adam@posidonia:~/fossils/vertebrates$ mkdir reptiles birds mammals
adam@posidonia:~/fossils/vertebrates$
```

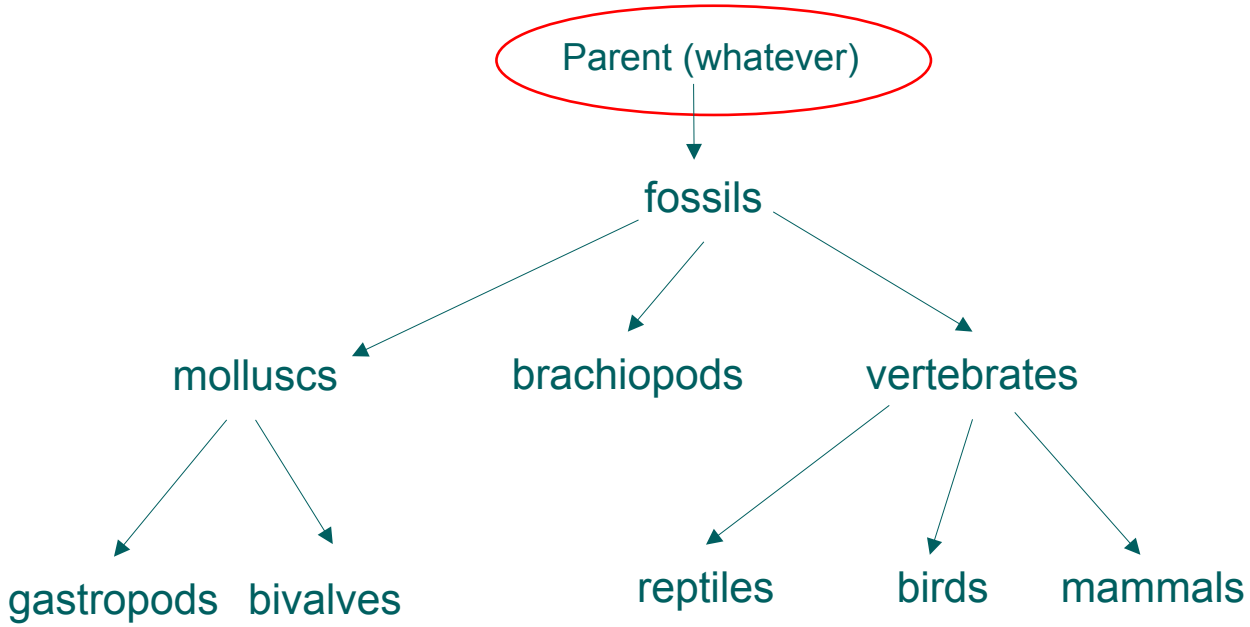


How to check?

find␣<path to directory>

Recursive listing

- 1. Go back to the parent
- 2. Use find there!




```
adam@posidonia: ~ 73x30
adam@posidonia:~/fossils/vertebrates$ cd ../..
adam@posidonia:~$ find fossils
fossils
fossils/molluscs
fossils/molluscs/gastropos
fossils/molluscs/bivalves
fossils/brachiopods
fossils/vertebrates
fossils/vertebrates/reptiles
fossils/vertebrates/birds
fossils/vertebrates/mammals
adam@posidonia:~$
```

parent of parent

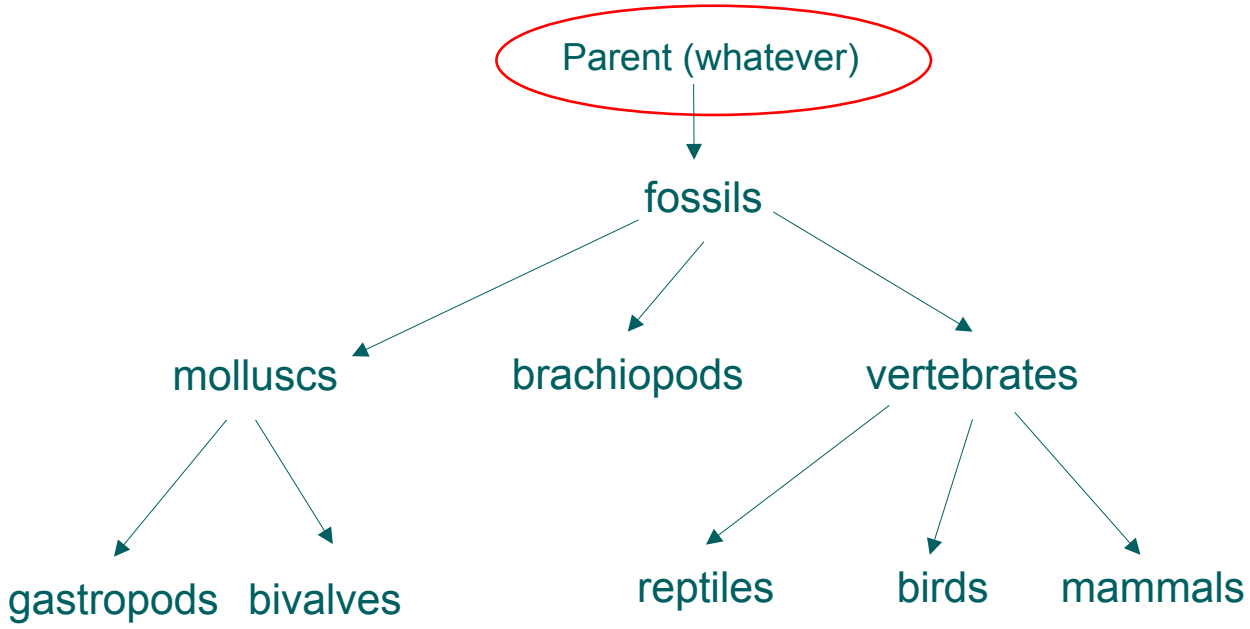
relative paths

Angled bracket or ***chevron***

find␣<path>␣>␣<path_to file>

Output redirection

- Whatever was output to the console is now in a new file!




```
adam@posidonia: ~ 73x30
adam@posidonia:~/fossils/vertebrates$ cd ../..
adam@posidonia:~$ find fossils
fossils
fossils/molluscs
fossils/molluscs/gastropos
fossils/molluscs/bivalves
fossils/brachiopods
fossils/vertebrates
fossils/vertebrates/reptiles
fossils/vertebrates/birds
fossils/vertebrates/mammals
adam@posidonia:~$ find fossils > fossil_path.txt
adam@posidonia:~$
```

“>” Will overwrite existing files!

Suggested nomenclature

( ): **Parenthesis** (open and close)

[ ]: **Bracket** (open and close)

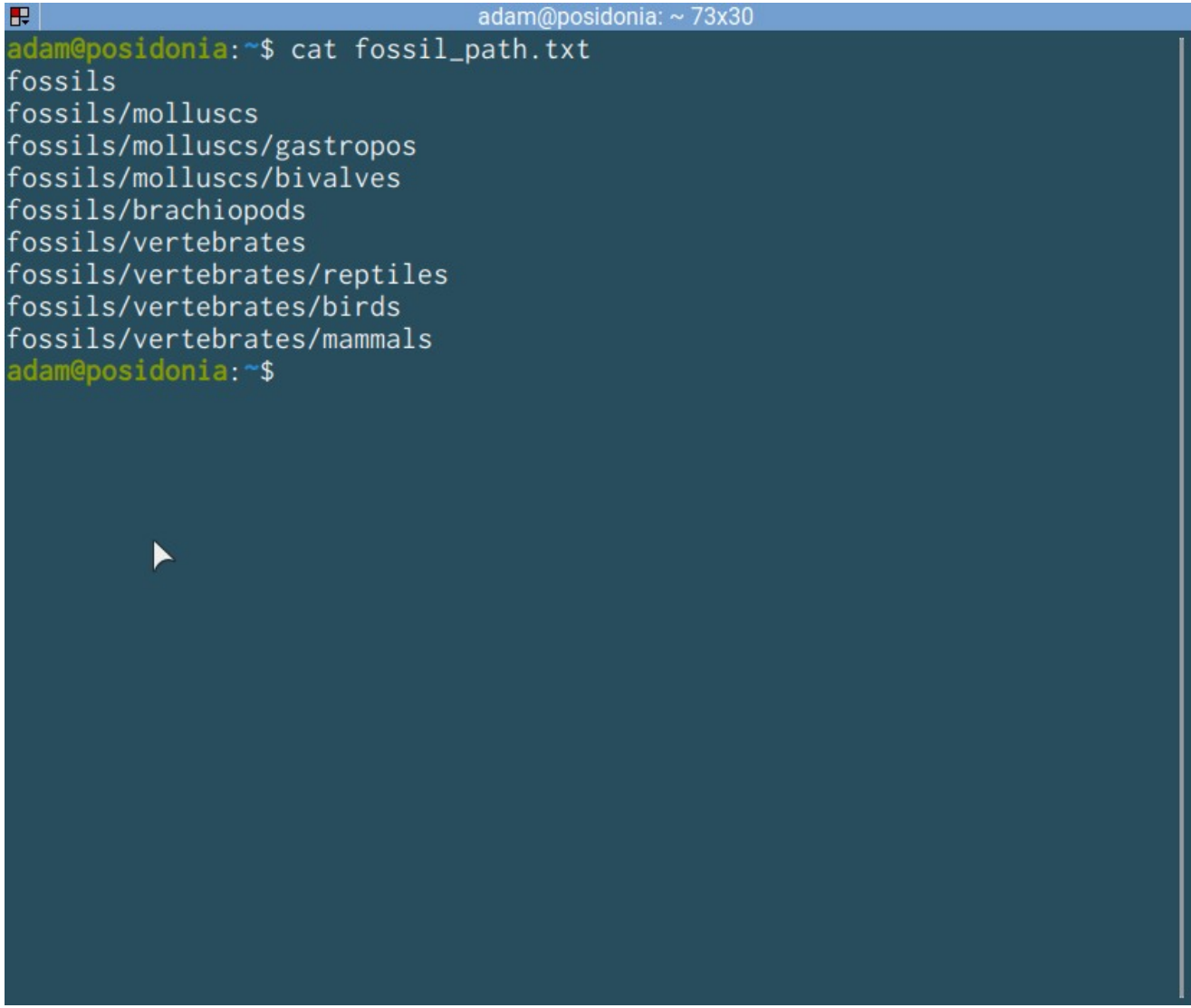
{ }: **Brace** (open and close)

< >: **Chevrons** (left and right)

`cat␣<path to file>`

Display contents of file

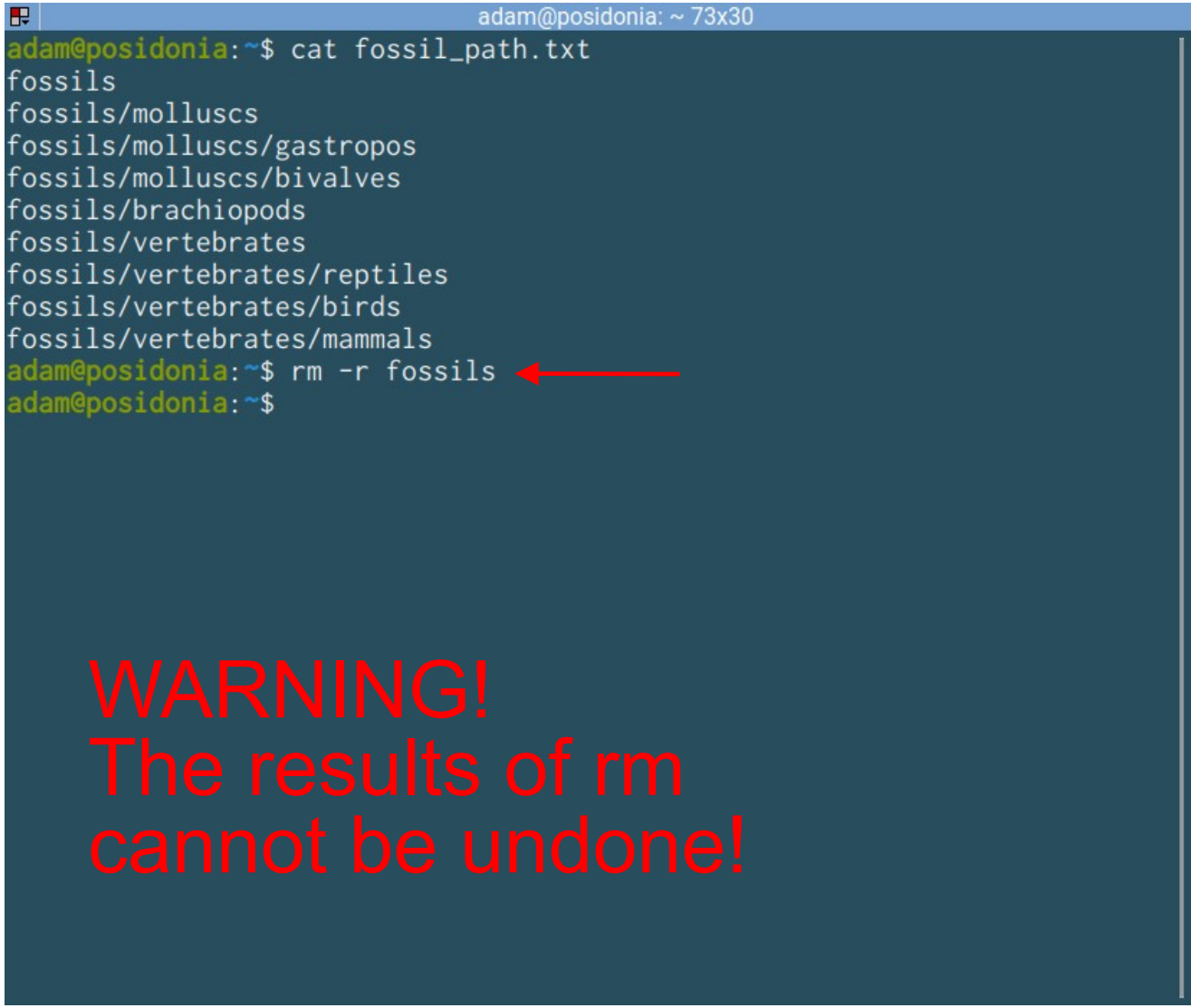
- Exactly as it was output to the console



rm␣-r␣<path to dir>

Recursive deletion (-r)

- Deletes the content of the directory and the directory itself
- rmdir doesn't work!
- No output = success?!

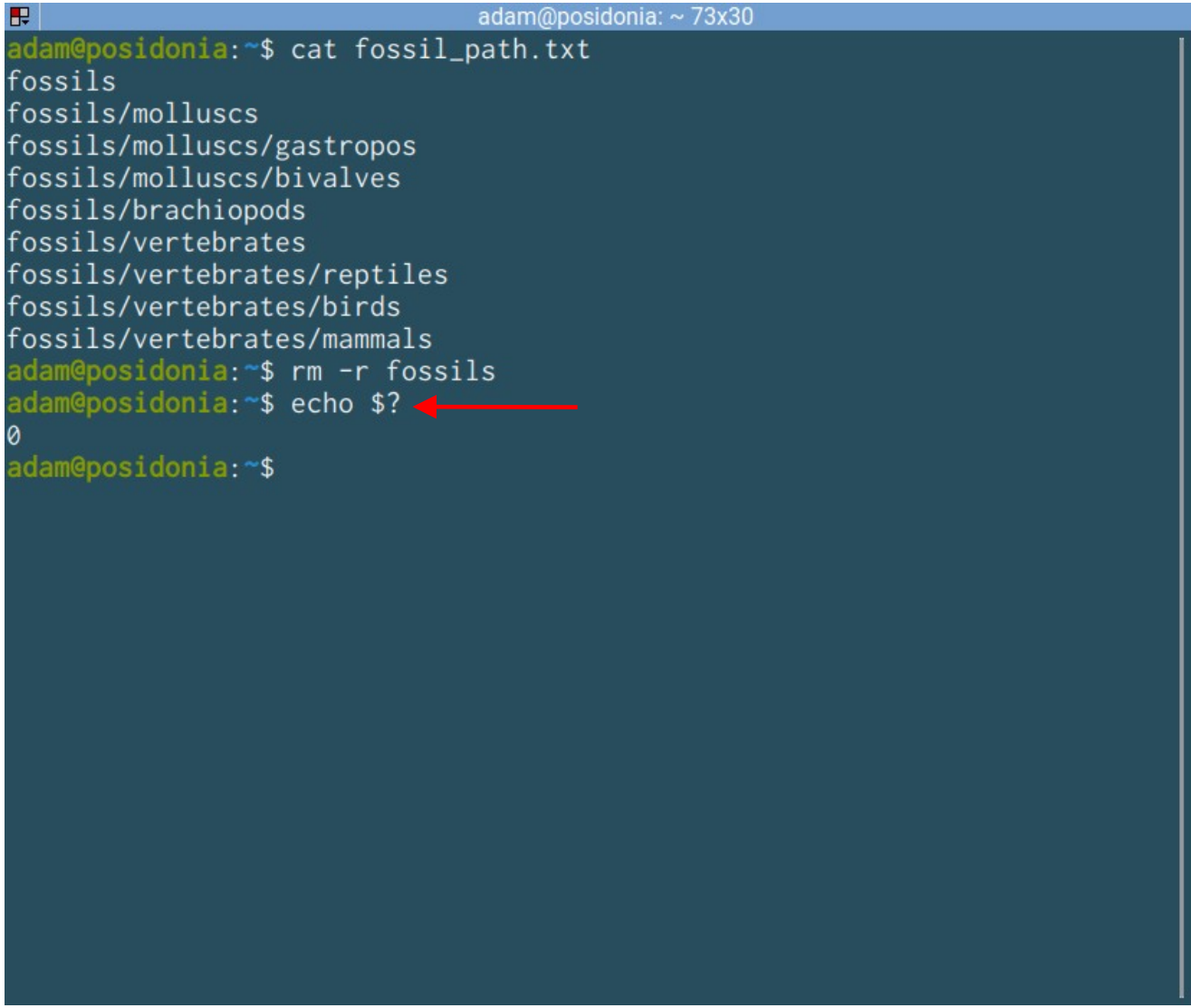


echo␣<text>

Print something

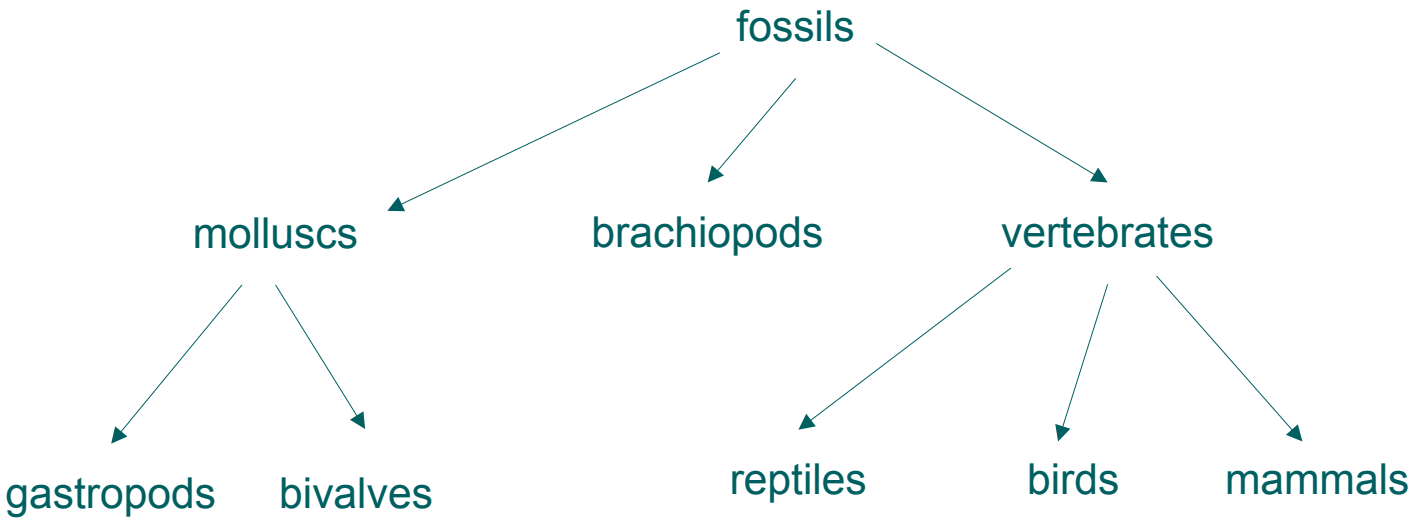
- Used to print things to the console (standard output)
- $? Is a special symbol: the exit code of the last command:
  - 0: Success
  - Other: Failure

https://www.redhat.com/sysadmin/exit-codes-demystified



Recreate the structure!

- Did you type things into the console?!



Hint 1. Use a general-purpose text editor!

Novice-friendly:

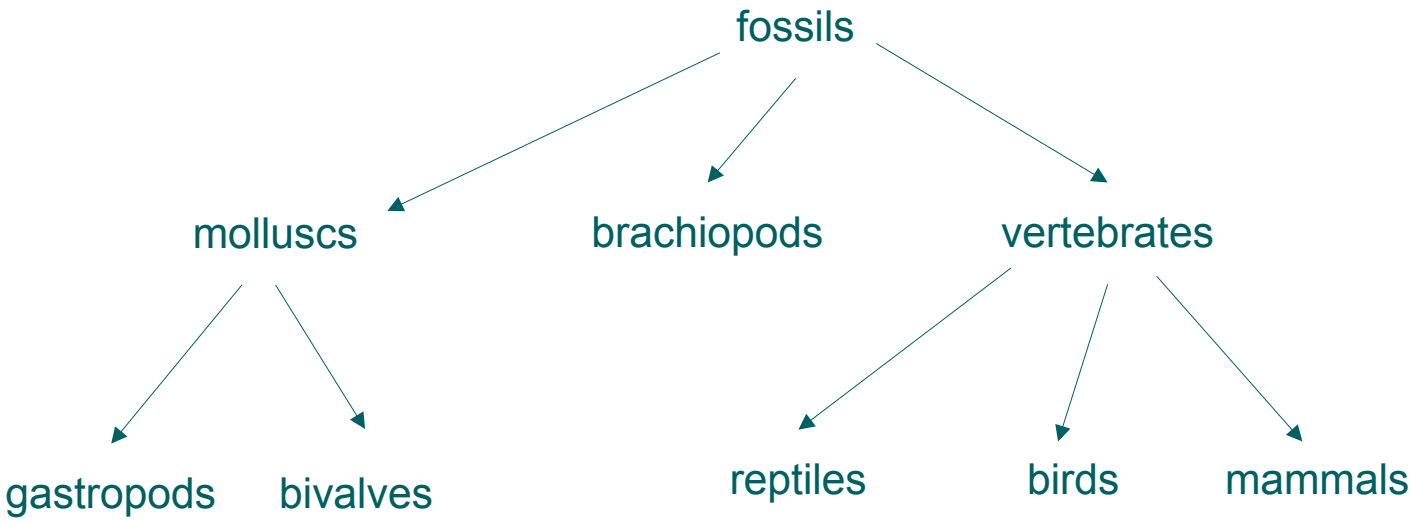
- Sublime Text
- **VS Code**
- Atom

Expert-friendly:

- Vim
- Emacs

Solution 2

- Using the same reference directory



Hint 2. we can use the contents of fossil_path.txt

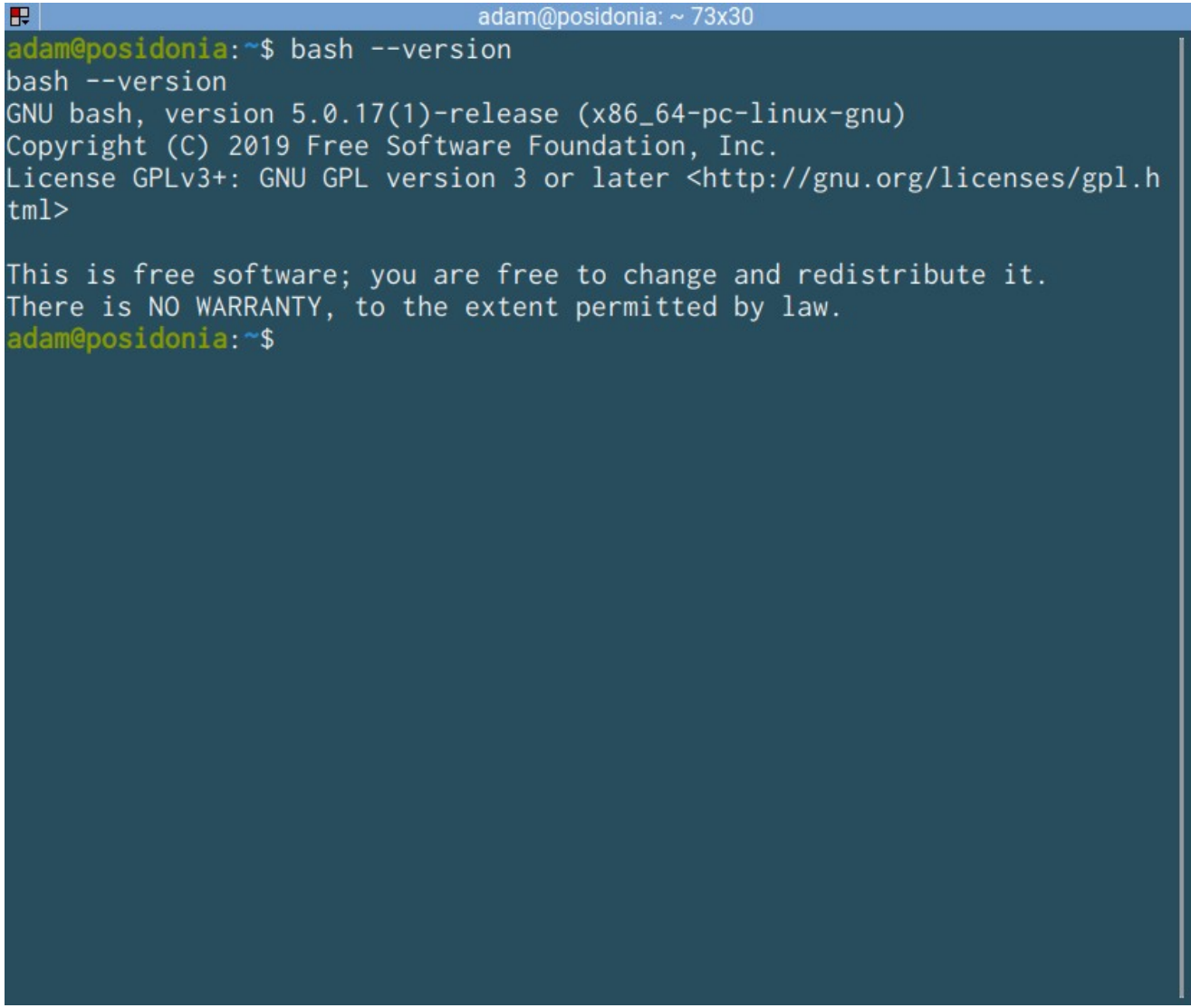
Add mkdir in front of every line, then copy and paste into the console!

```
1 mkdir fossils
2 mkdir fossils/molluscs
3 mkdir fossils/molluscs/gastropods
4 mkdir fossils/molluscs/bivalves
5 mkdir fossils/brachiopods
6 mkdir fossils/vertebrates
7 mkdir fossils/vertebrates/reptiles
8 mkdir fossils/vertebrates/birds
9 mkdir fossils/vertebrates/mammals
```

`bash␣--version`

Running console applications

- `--version`: ask for program version
- `--help`: display help for program

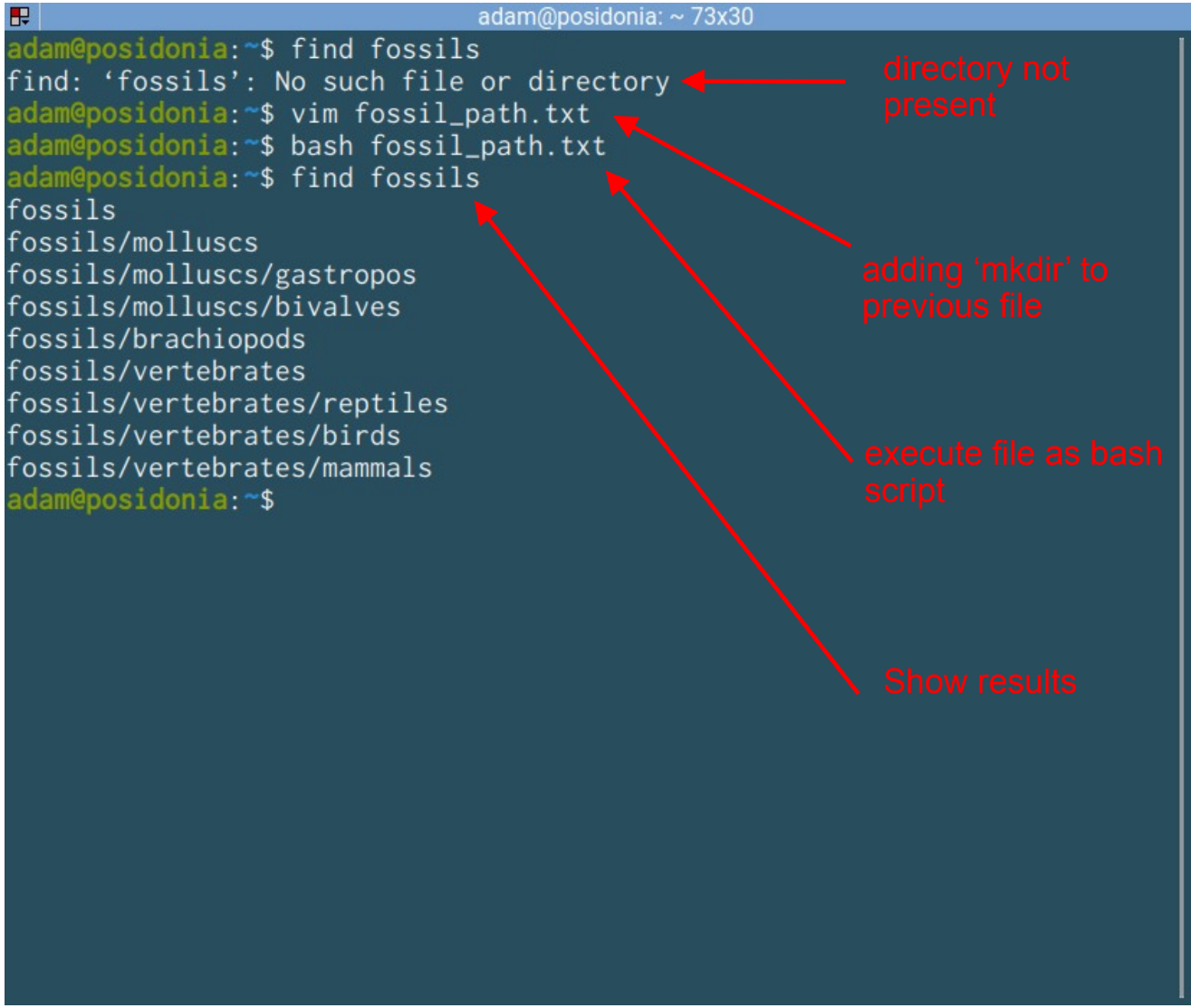


bash␣<path>

Executing shell scripts

- The text we created is actually a shell script draft
- The “bash” console application program can be used to execute it.

https://www.redhat.com/sysadmin/exit-codes-demystified



chmod␣+x␣<path>

Making executables

- Works with any file!
- Can be used to change file attributes: read, write, execute (for every user)
- “Shebang” indicates the kind of language used to interpret the file, for bash: “#!/bin/bash”
- Typical shell script files with .sh
- Execution requires a path!

Shebang

```
 1 #!/bin/bash
 2
 3 mkdir fossils
 4 mkdir fossils/molluscs
 5 mkdir fossils/brachiopods
 6 mkdir fossils/vertebrates
 7 mkdir fossils/vertebrates/reptiles
 8 mkdir fossils/vertebrates/birds
 9 mkdir fossils/vertebrates/mammals
10 mkdir fossils/molluscs/bivalves
11 mkdir fossils/molluscs/gastropods
12
13 echo "You have run the fossils.sh file!"
```

```
 adam@TheStuff  ~/work  ls -l
total 4
-rw-rw-r-- 1 adam adam 311 aug   18 05:26 fossils.sh
 adam@TheStuff  ~/work  chmod +x fossils.sh
 adam@TheStuff  ~/work  ls -l
total 4
-rwxrwxr-x 1 adam adam 311 aug   18 05:26 fossils.sh
 adam@TheStuff  ~/work  ./fossils.sh
You have run the fossils.sh file!
 adam@TheStuff  ~/work  find .
.
./fossils.sh
./fossils
./fossils/brachiopods
./fossils/molluscs
./fossils/molluscs/gastropods
./fossils/molluscs/bivalves
./fossils/vertebrates
./fossils/vertebrates/reptiles
./fossils/vertebrates/mammals
./fossils/vertebrates/birds
 adam@TheStuff  ~/work
```

Not executable

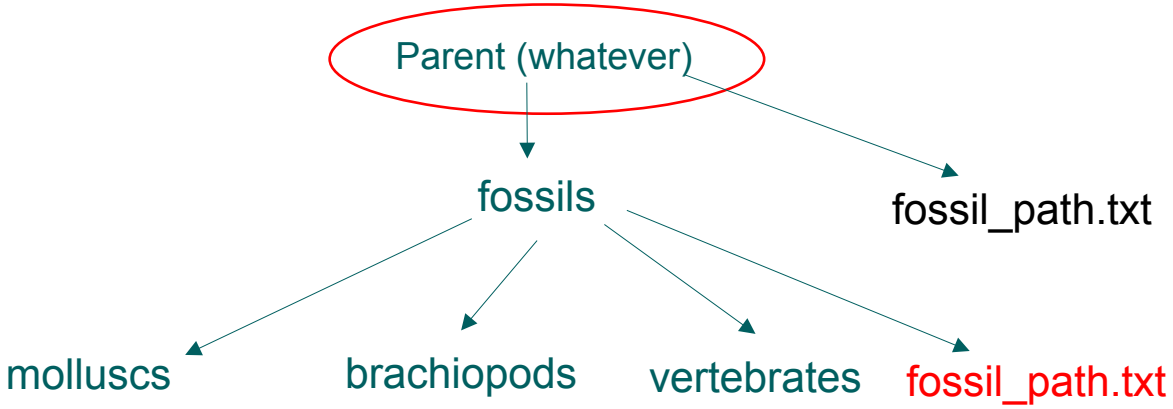
Executable

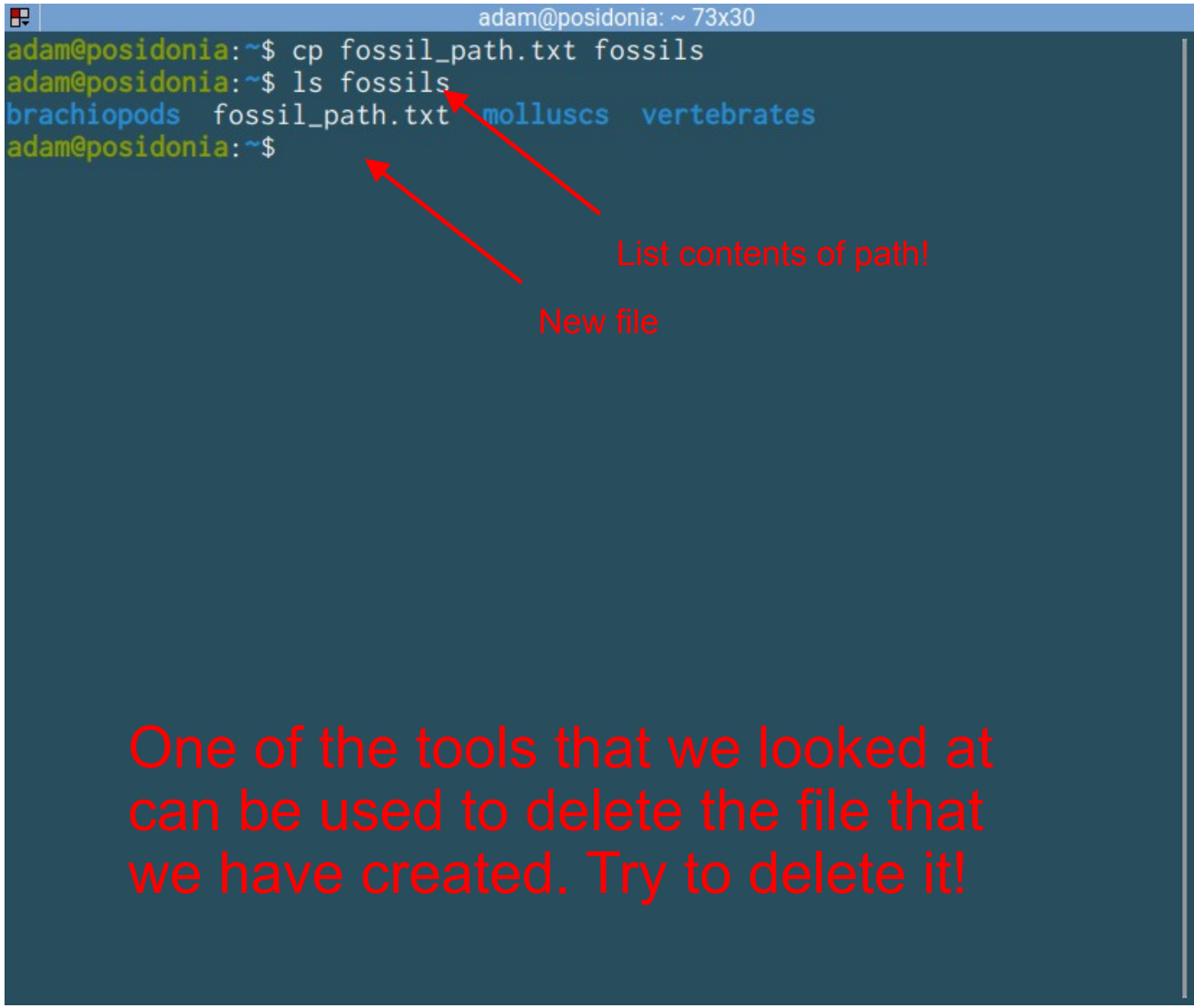
Execution

cp␣<what>␣<where>

Copying a file or directory

- Target directory or file
- If directory, the file will be put into it

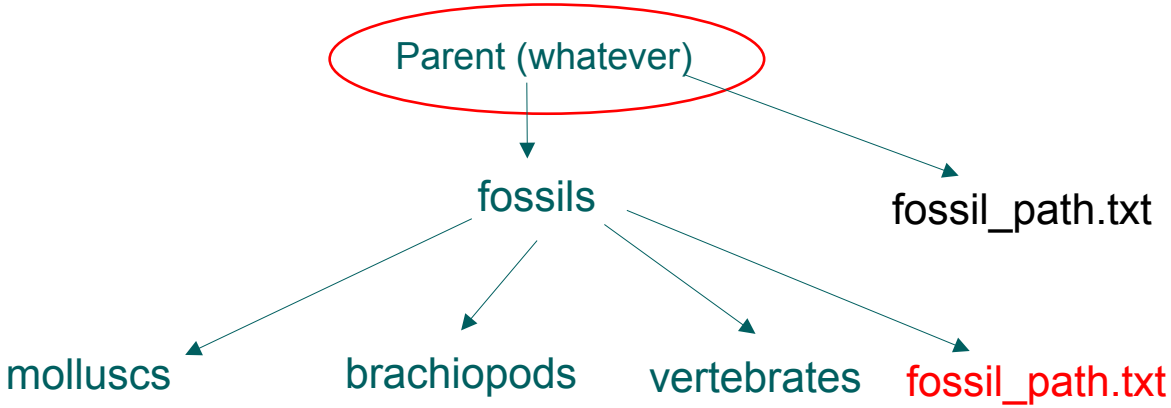


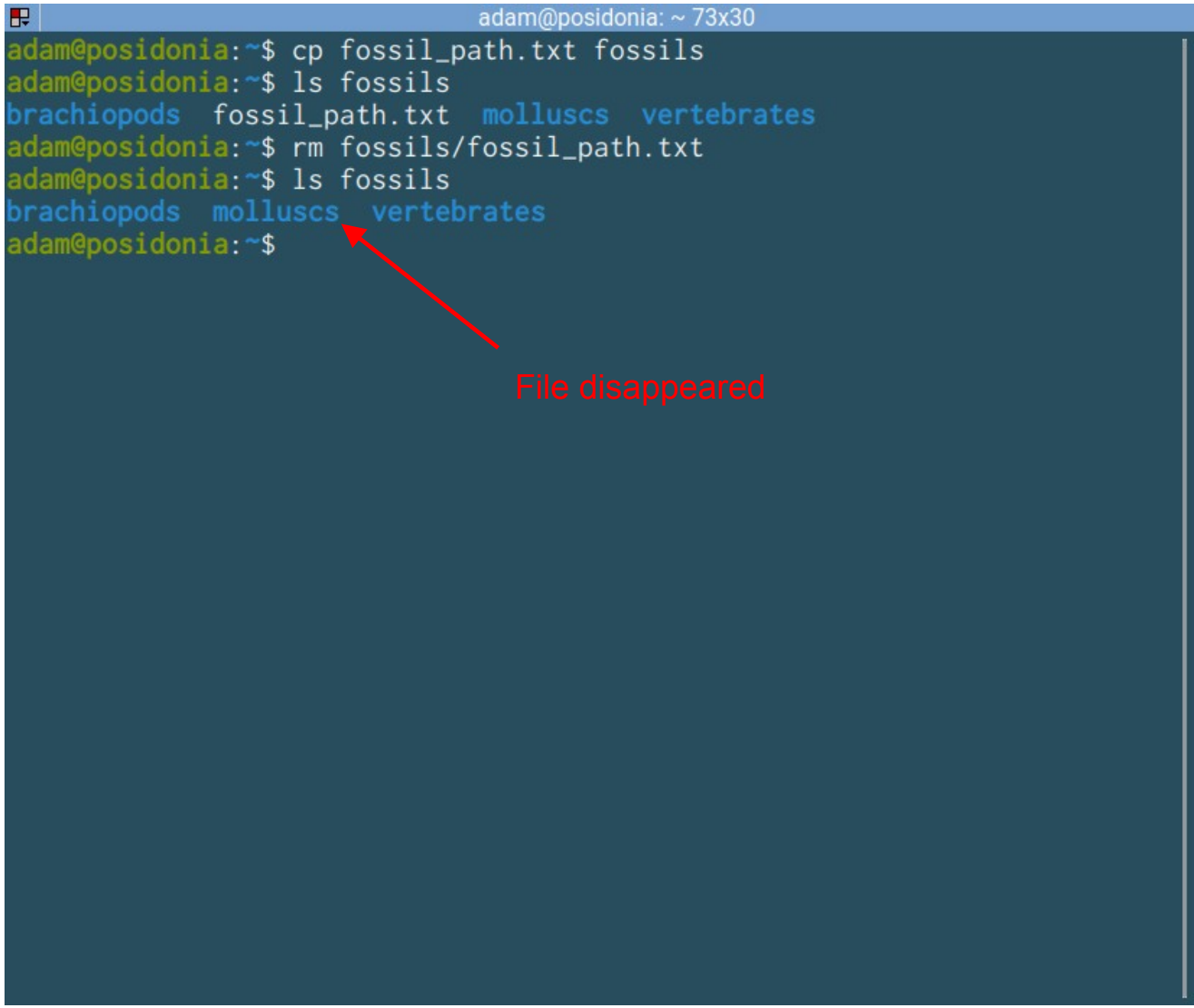


rm␣<path_file>

Without –r removes a single file

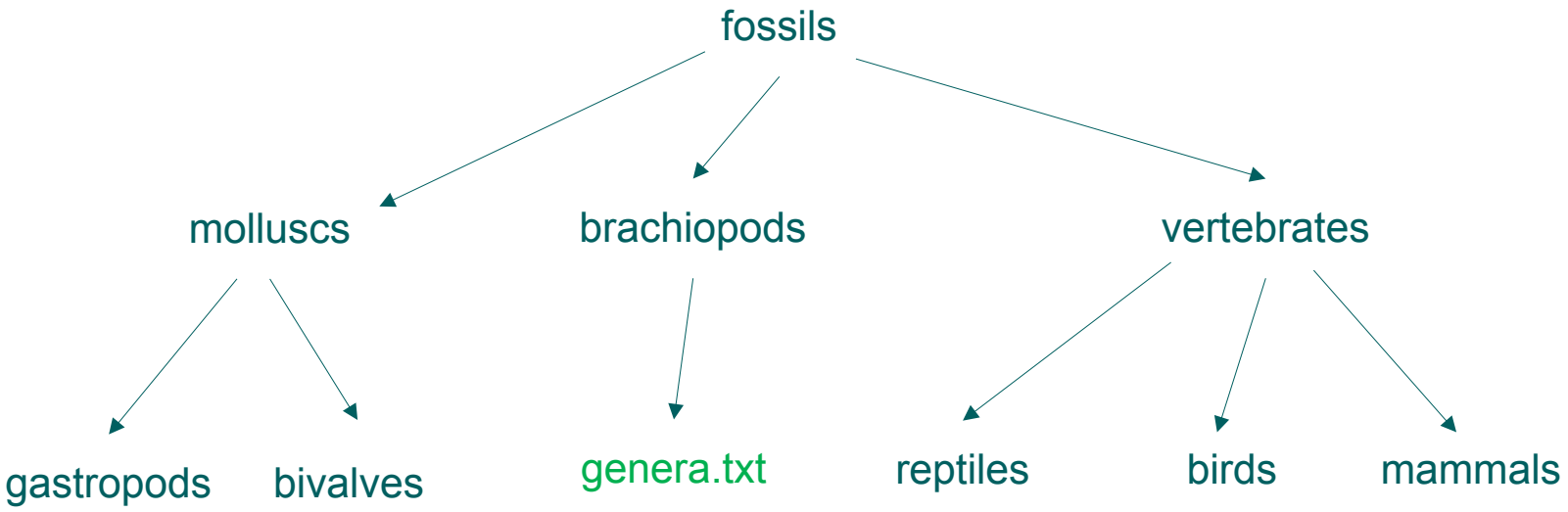
- As with cp, multiple files can be passed to this (separated by spaces)





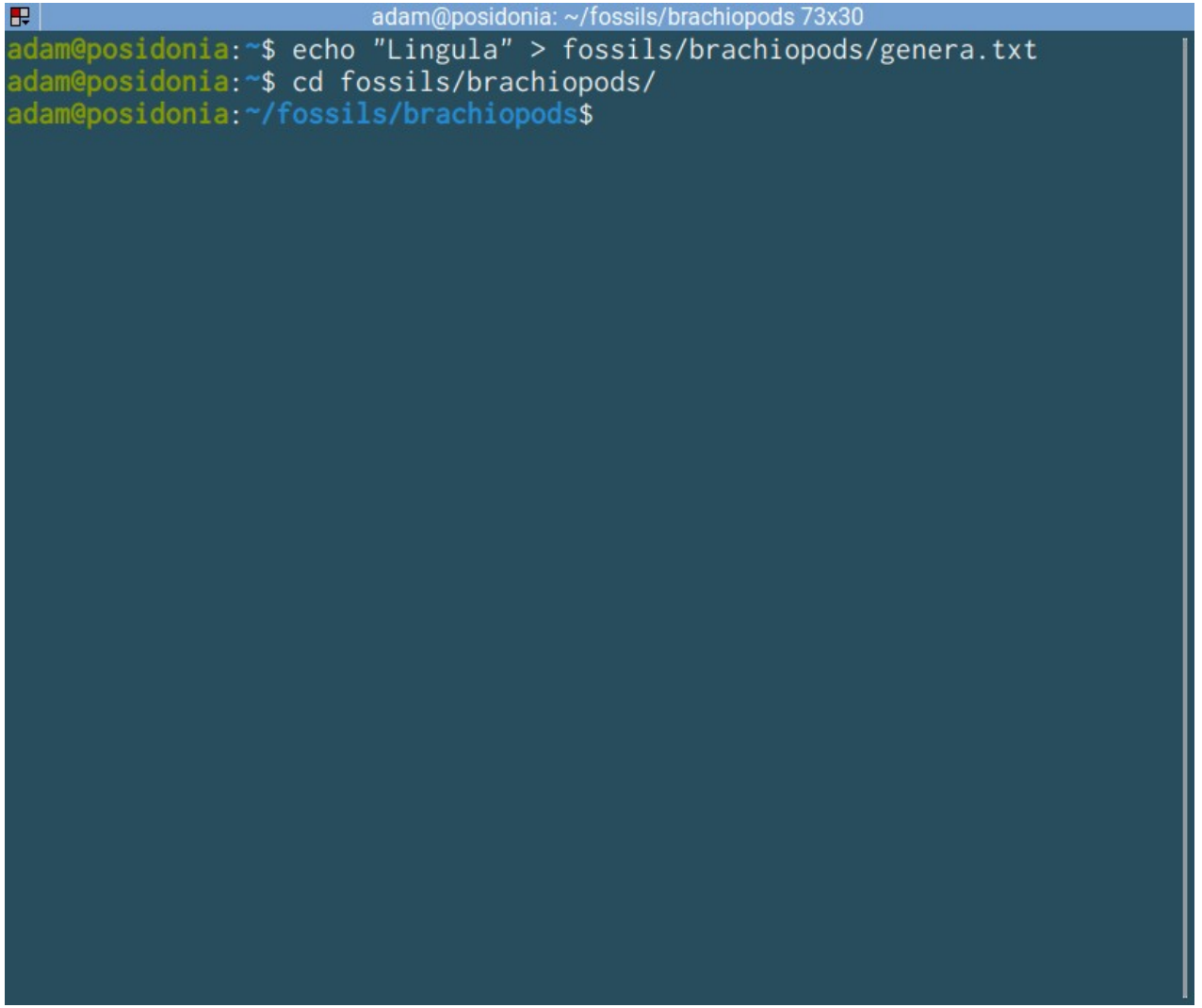
Demo!

1. Use an `echo` statement to write the genus name "*Lingula*" into `fossils/brachiopods/genera.txt`!
2. Then change directory to `brachiopods`.



echo␣“Lingula”␣>␣fossils/brachiopods/genera.txt

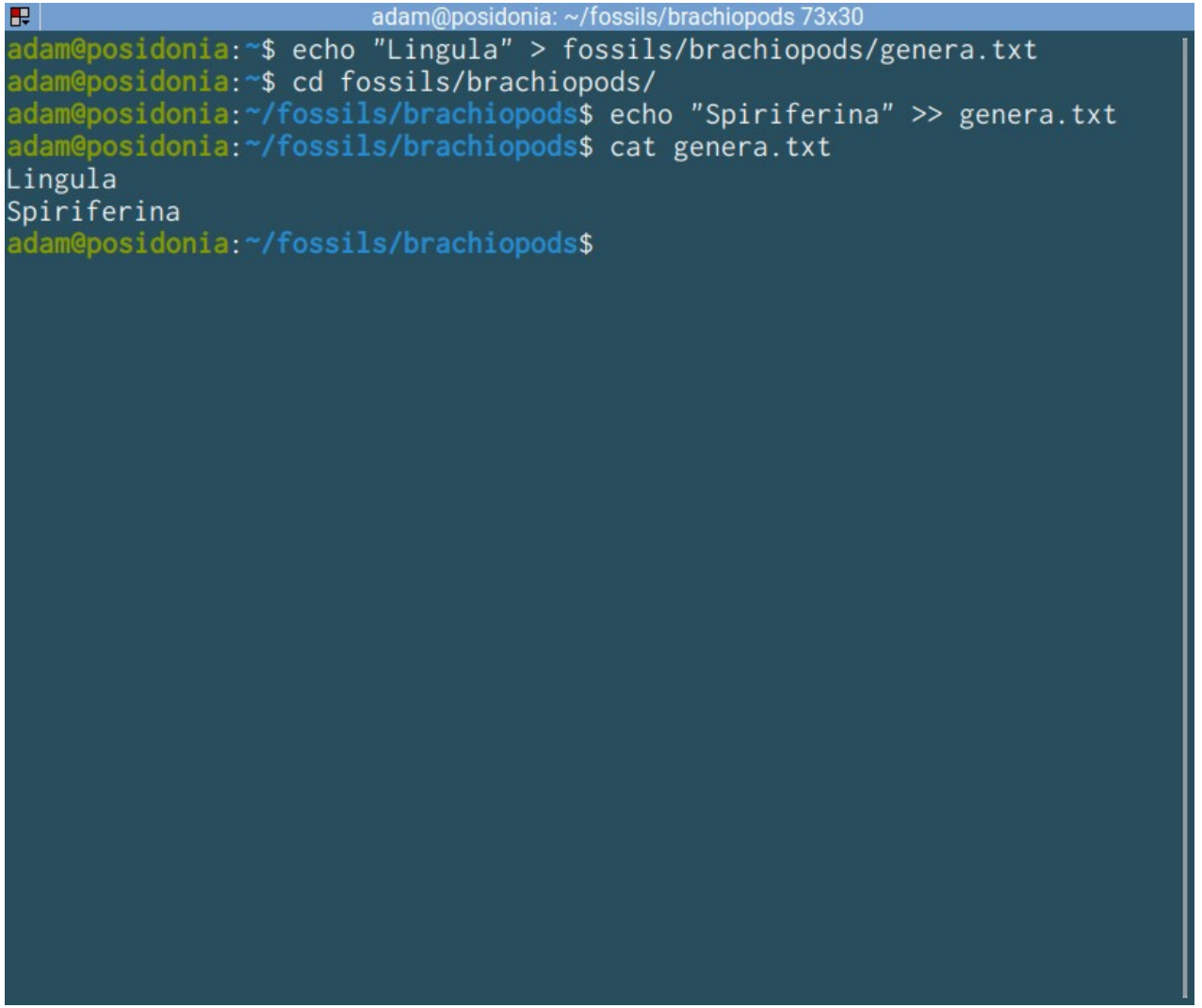
- You can use the double chevron >> to append to an existing file



echo␣“Spiriferina”␣>>␣genera.txt

Appending to files

- You can use the double chevron >> to append to an existing file
- Added to new line!



Special characters

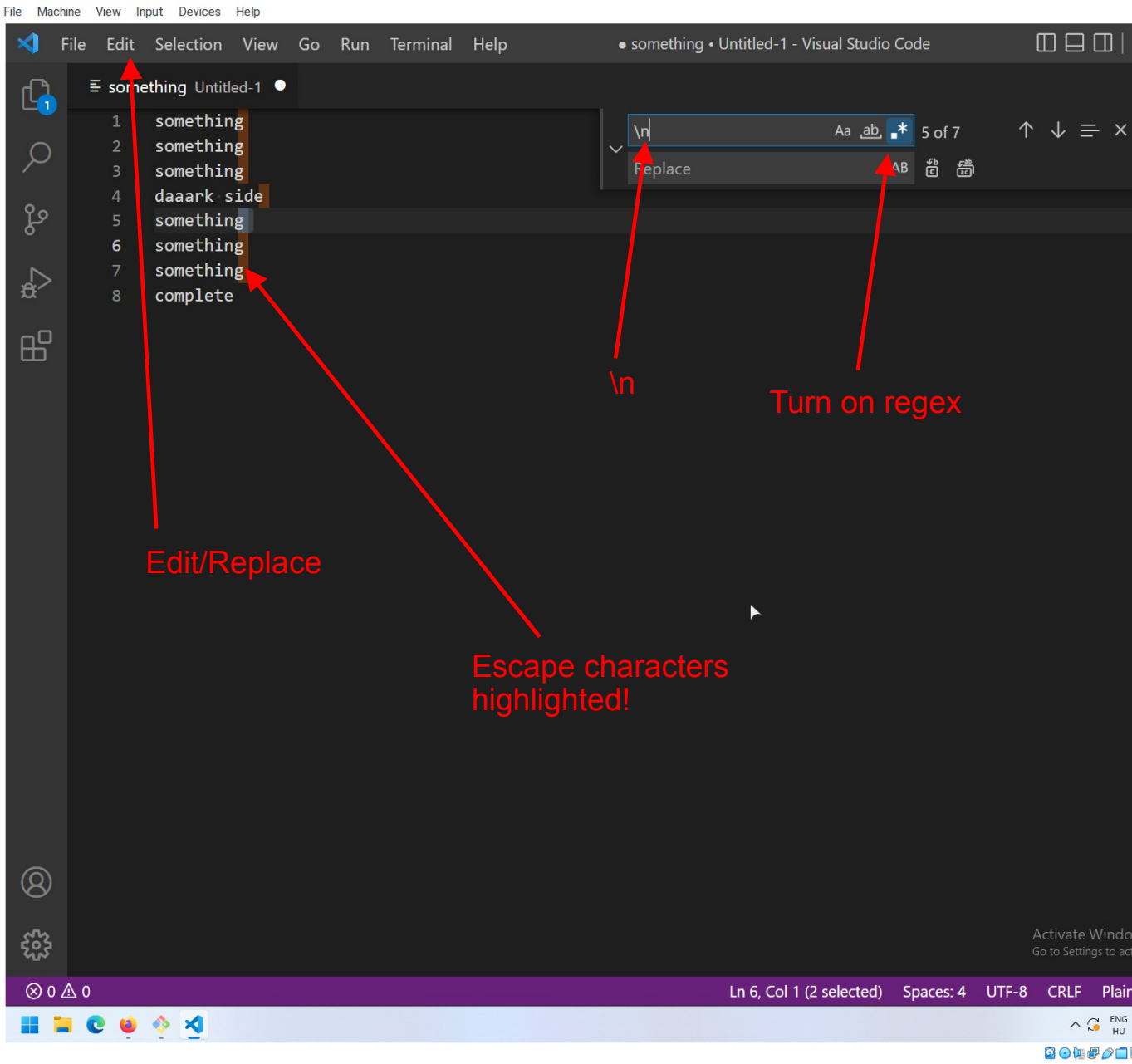
We use so called “escape characters” to denote special symbols, that sometimes have other meanings.

\n: newline escape

\t : tab escape

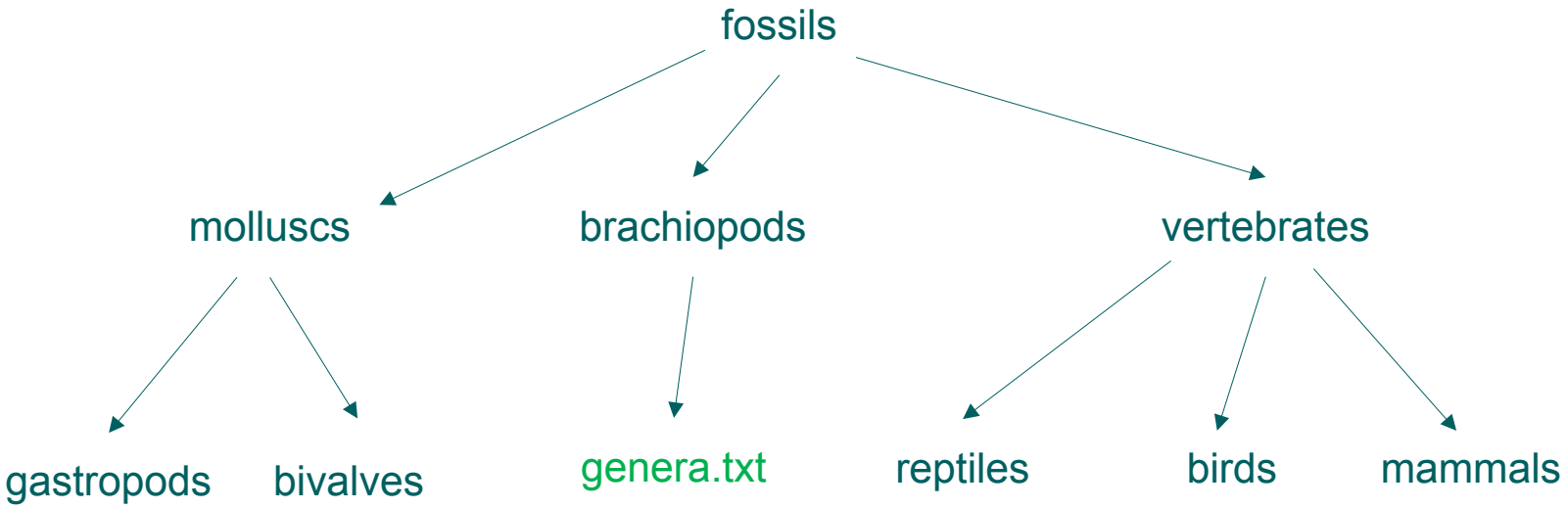
\” : double quote escape

\’ : single quote escape



Demo!

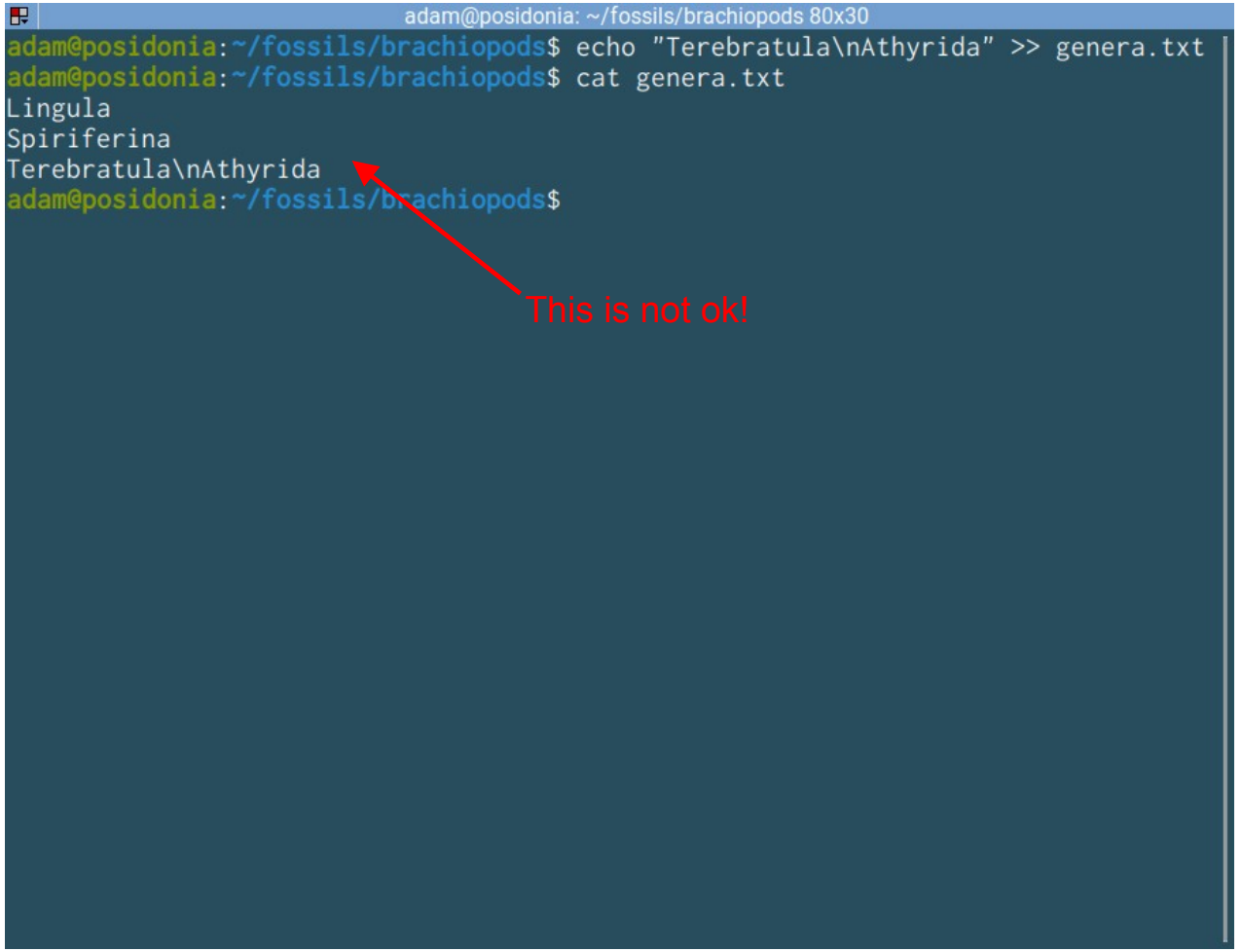
1. Use an `echo` statement to also add the genus name "*Terebratula*" and "*Athyris*" into `fossils/brachiopods/genera.txt`, **`use a newline escape between them!`**
2. Then change directory to `brachiopods`.



`echo␣"Terebratula\nAthyrida"␣>>␣genera.txt`

Appending to files

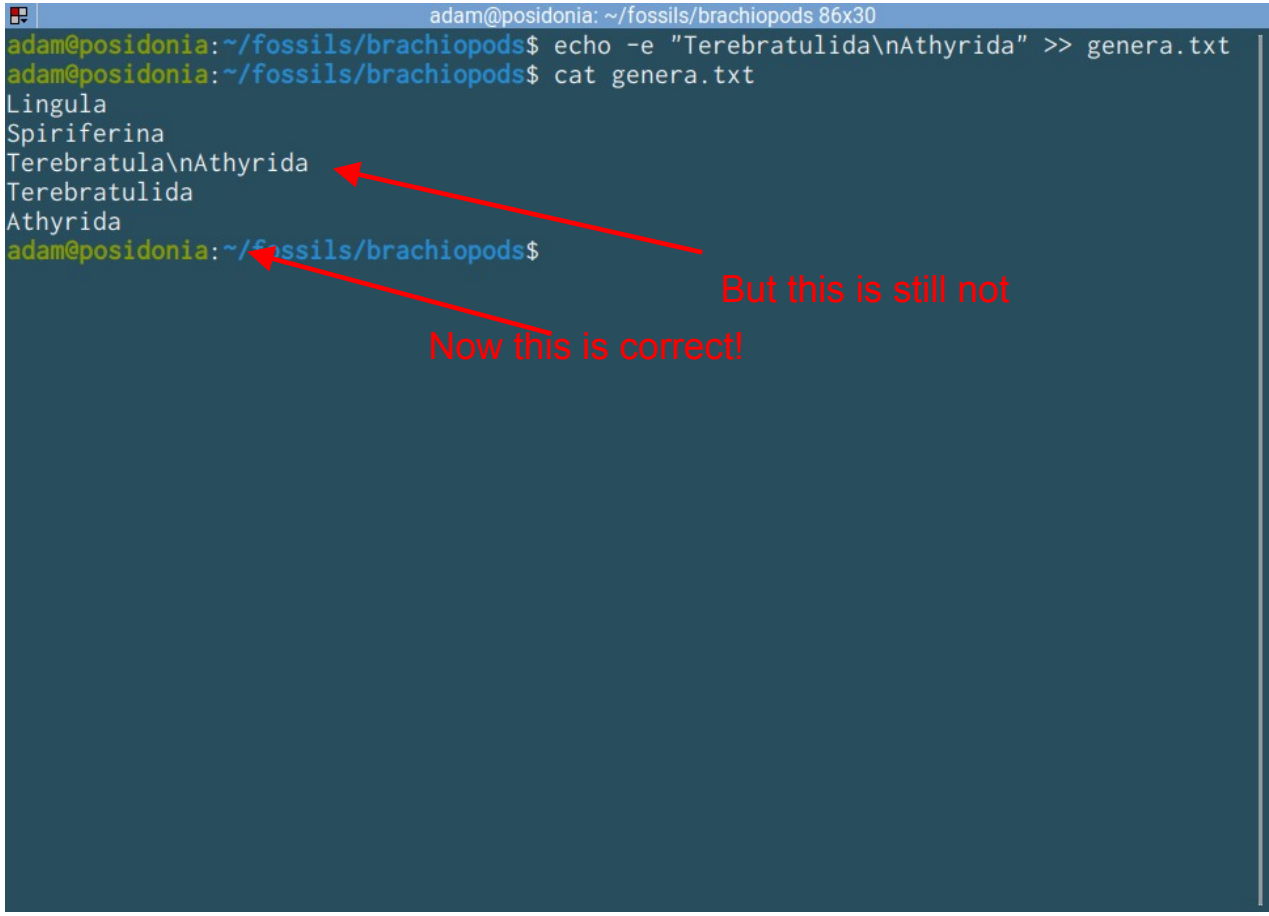
- It doesn't seem to work!
- Echo needs to know to replace the combination `\n` with the newline character!



echo␣-e␣"Terebratula\nAthyrida"␣>>␣genera.txt

Appending to files

- Use the –e option!
- Our file is messed up. Options:
  - 1. Redo our file
  - 2. Use an editor to correct
- Delete the bad line!
- Better, next time: go back in time.



Basic version control with Git

and GitHub

Why version control?

Projects evolve in a non-linear way, especially programming projects.

- Multiple people work on them, sometimes at the same time
- Recording the history of project development
- Working with many files
- Sharing code is necessary, we also need to know who changes what





Difference between Git and GitHub?

git

- Locally running application
- Operates with files in a local directory (repository)
- Works without a remotes!



GitHub and GitLab

- Remote servers with copies of the repository

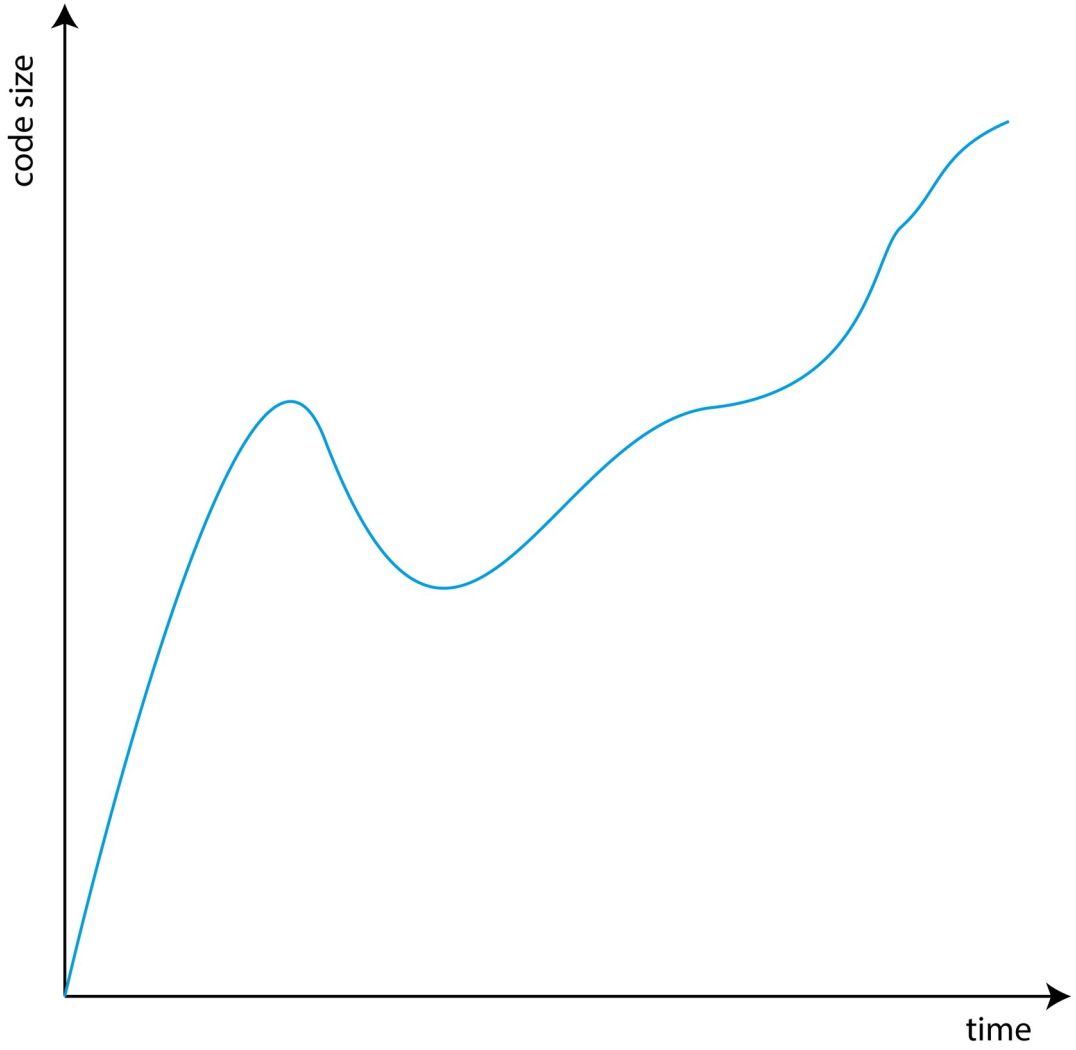




The basic use of git

Record snapshots of how a project develops.

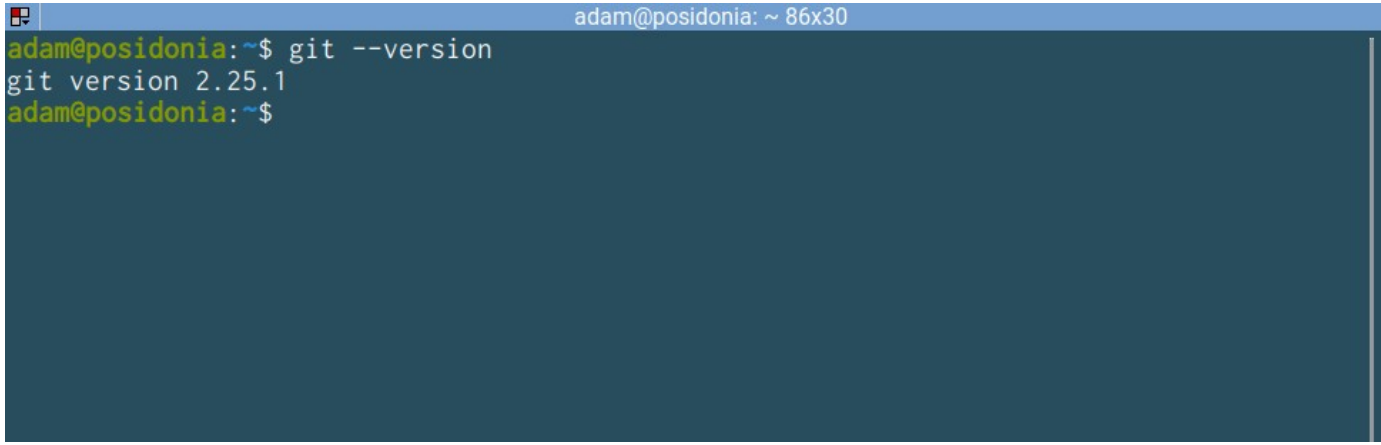
- Code develops in a non-linear, but continuous way, with lots of small changes:
  - Contents of files change
  - New files are added to the repository
  - Old files are delete from the respository

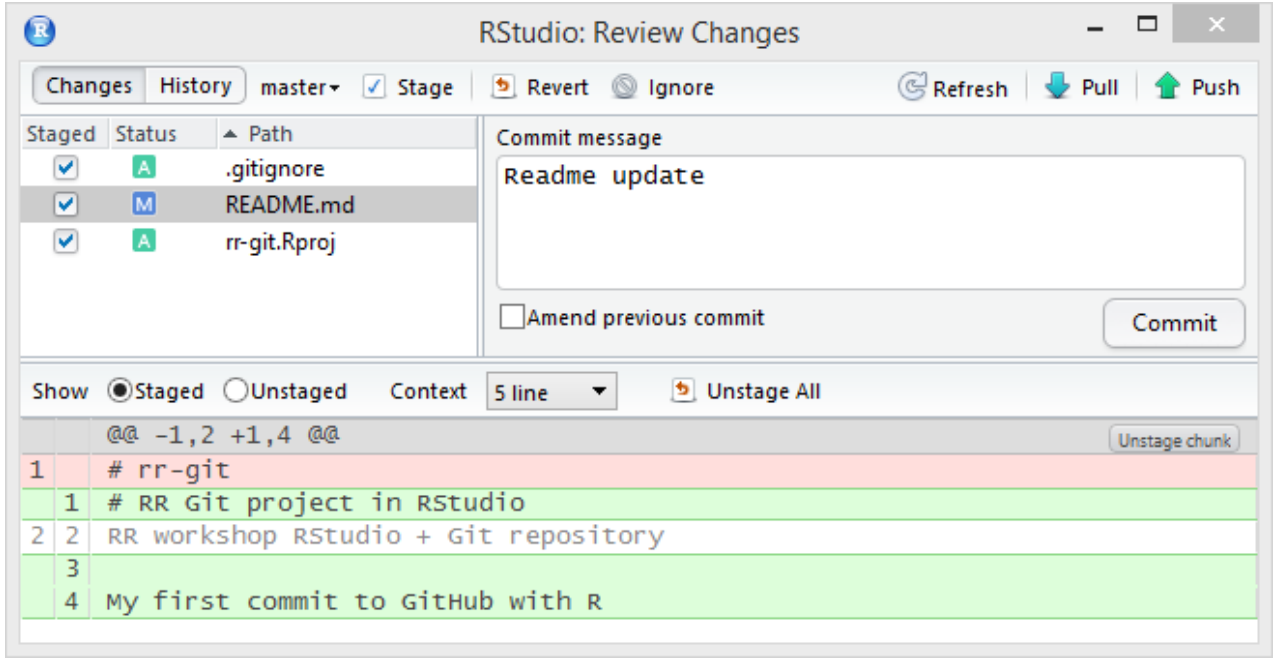


Interfaces to git

Git is a command line (console application)

- The complete features are only available via the command line!
- Simplified graphical interfaces written for novices, embedded in IDEs
- These actually just translate the actions to the command line application -> Experiment!

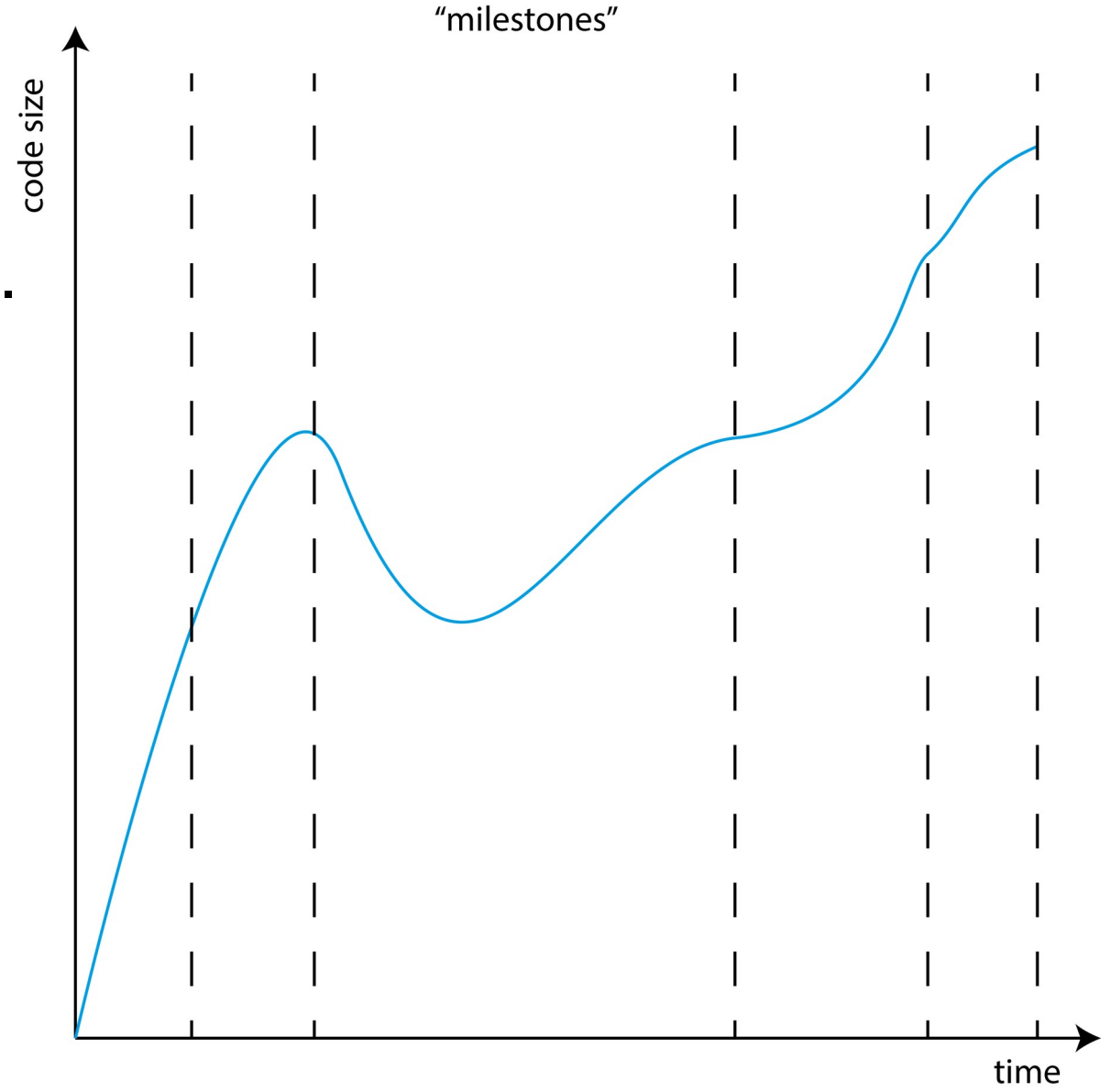




The basic use of git

Record snapshots of how a project develops.

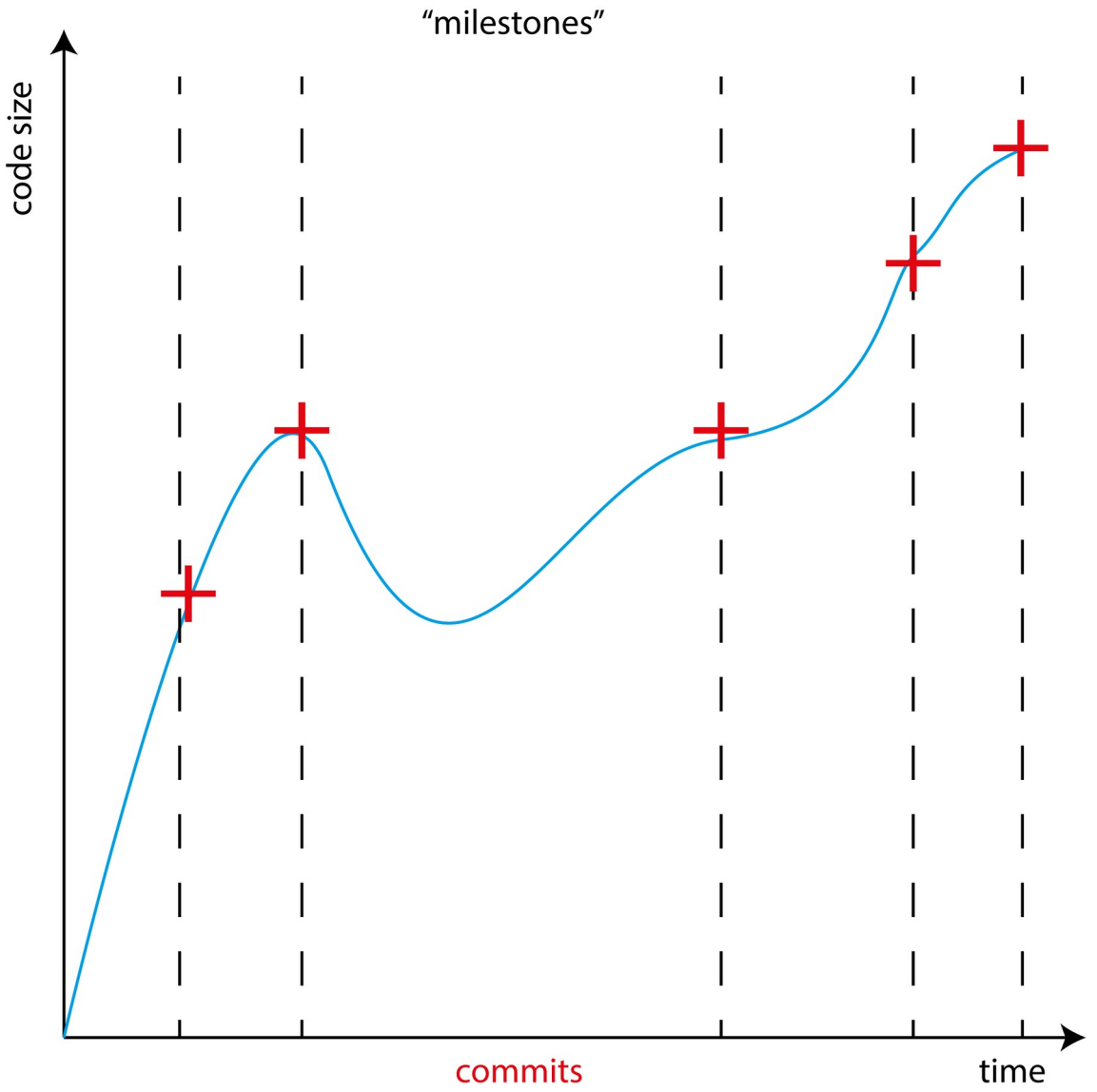
- Specific states of the code represent milestones:
  - Something works completely
  - Everything is cleaned up
  - Ready for further development
- In between these are transient states, when you are working on something but that is not yet done.



The basic use of git

Record snapshots of how a project develops.

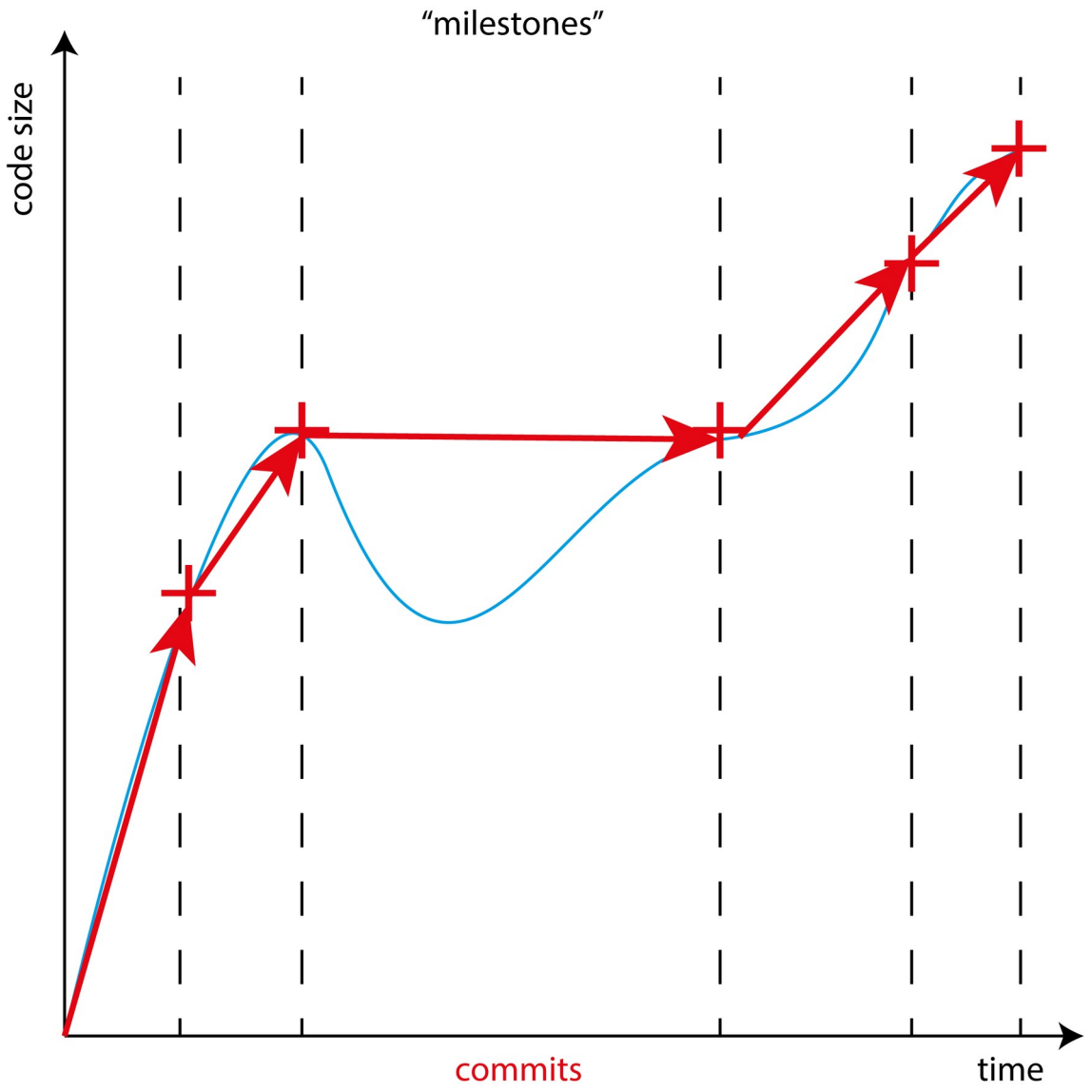
- These milesones can be saved and accessed at any time.
- These states are called as ‘commits’ in git’s terminology



The basic use of git

Record snapshots of how a project develop

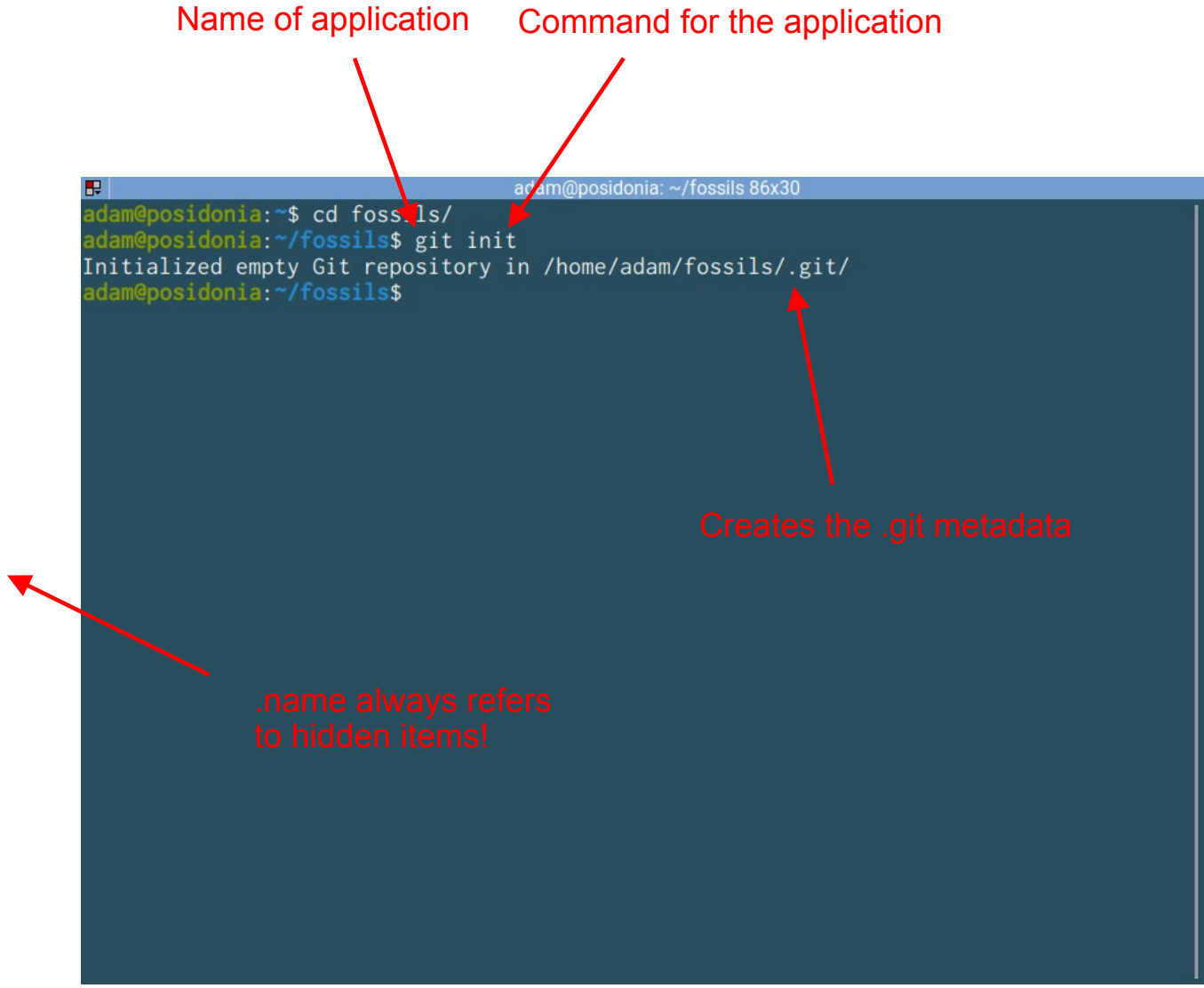
- Only the committed stages are recorded, the rest of the history is discarded
- The git repository is recorded as changes from one commit to the next



`git init`

Create a new git repository in current directory.

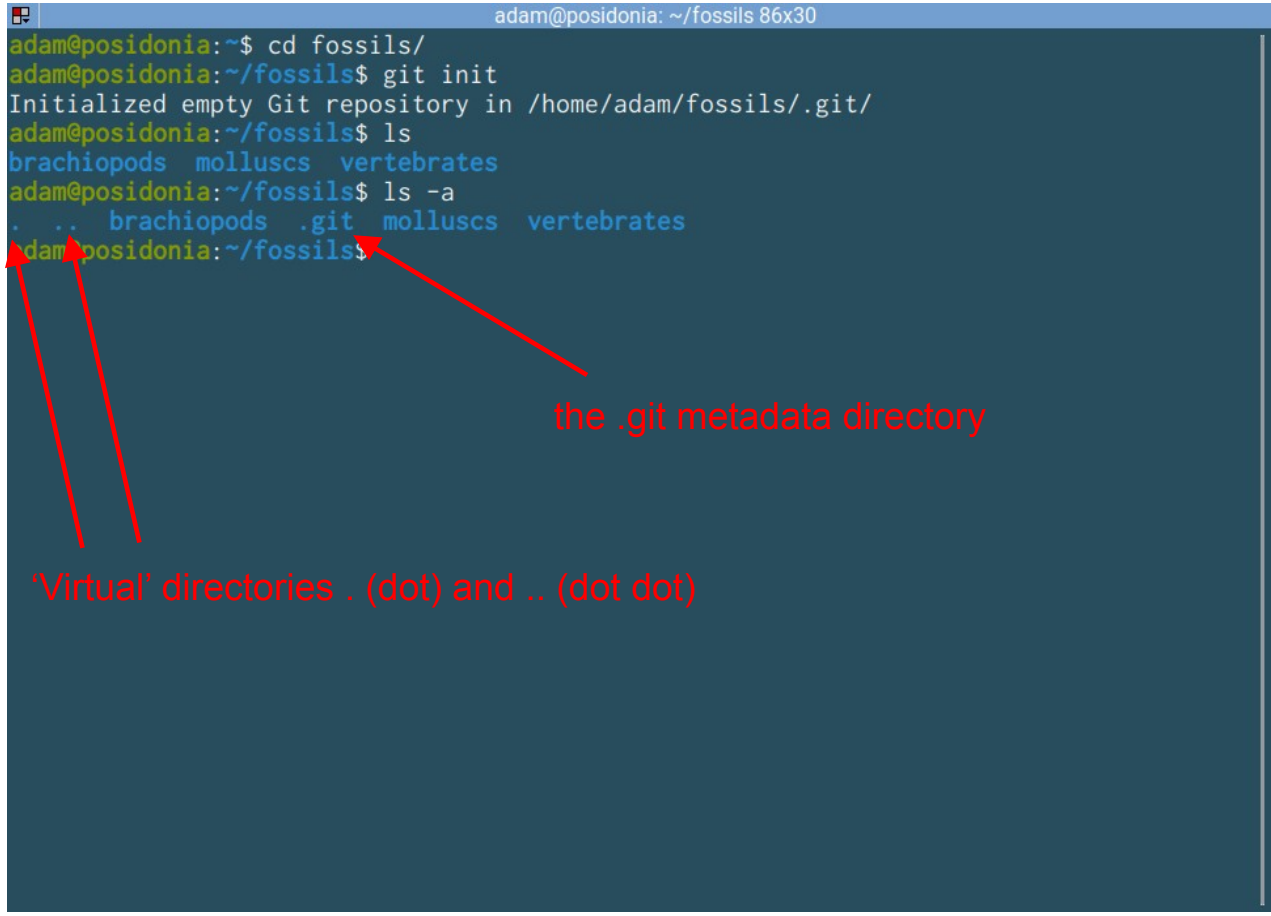
- A git repository is a directory with git metadata in it.
- The git metadata are in the `.git` directory



`ls␣-a`

List all files and directories in directory, including hidden items!

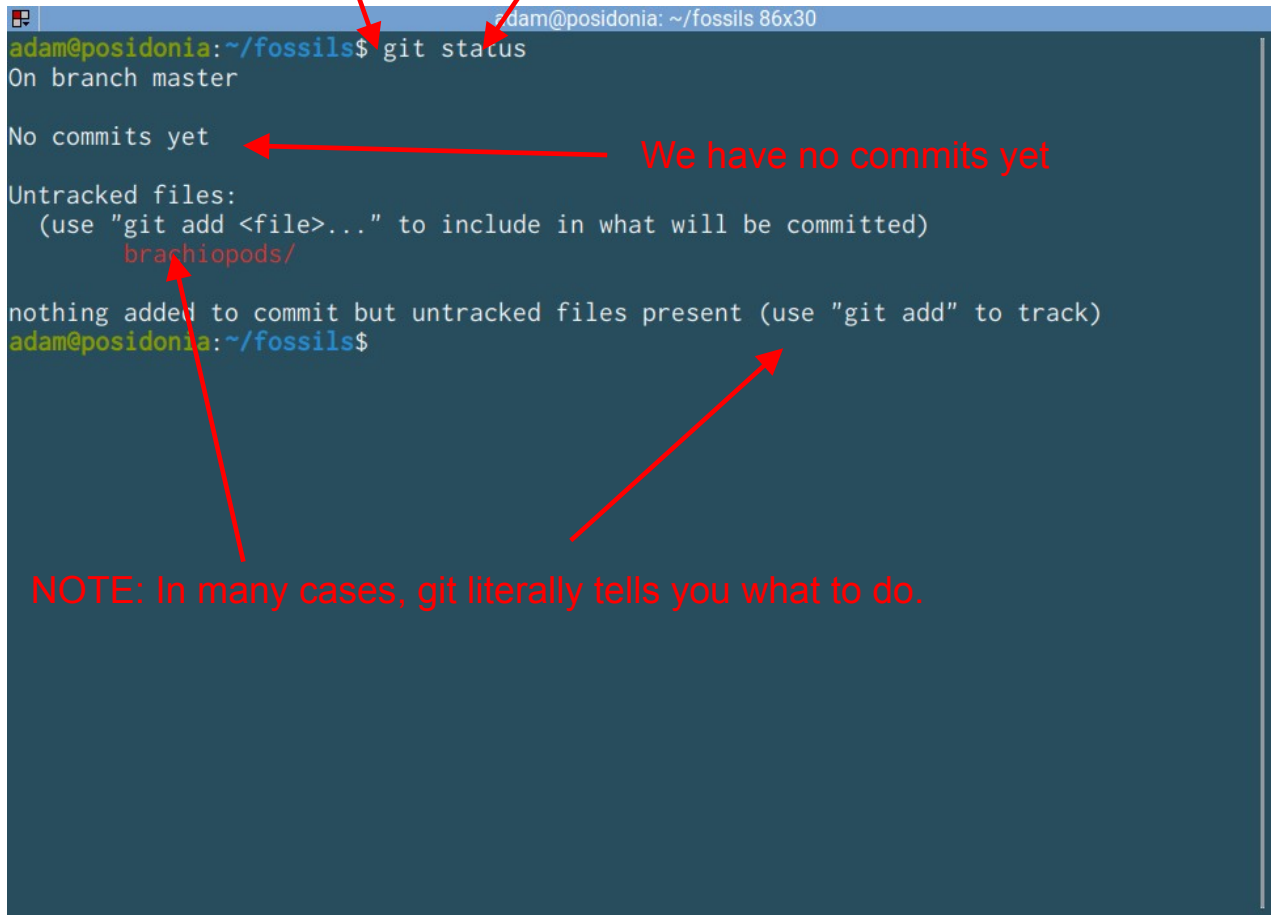
- The double dot (`..`) represents a way to refer to the previous directory, as we have seen earlier
- The single dot (`.`) represents a way to refer to the current directory.
- Note: `cd brachiopods` and `cd ./brachiopods` are the same!



git status

Show the status of the current respository

- A series of commits is called a ‘branch’. Simple repos use only one. There is always a current one
- Git has detected that there are things in the repo that are not registered.
- **Git can only detect files. Empty directories are not recorded!**



Staging

The preparation of a commit

- Commits are permanent, or are difficult to remove once done, so we have tools to make sure that they are ok
- Changes first have to be staged, before commiting. This allows us to include only specific changes in the commit, and to make sure that we are doing things ok.

Staging and commit (Airport)

Initial boarding pass control vs. boarding

- If you go through security you are staged to fly. You are expected to be on the plane, but you can still leave.
- If you board the plane and the cabin doors are closed, you are committed to a flight.

Getting staged



In the staged area, waiting to be commited



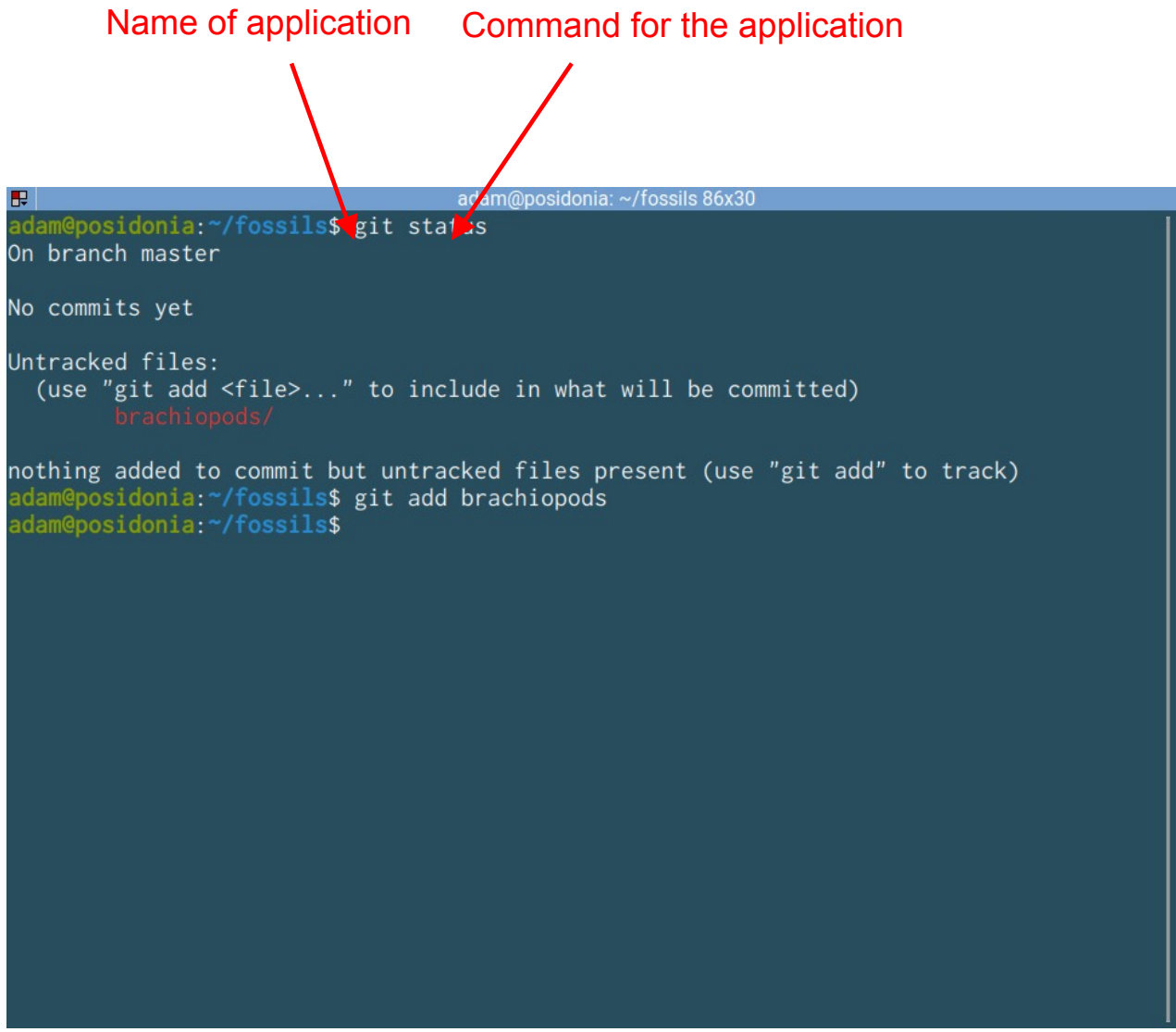
The commit



`git␣add␣<path>`

Stage the target file or directory.

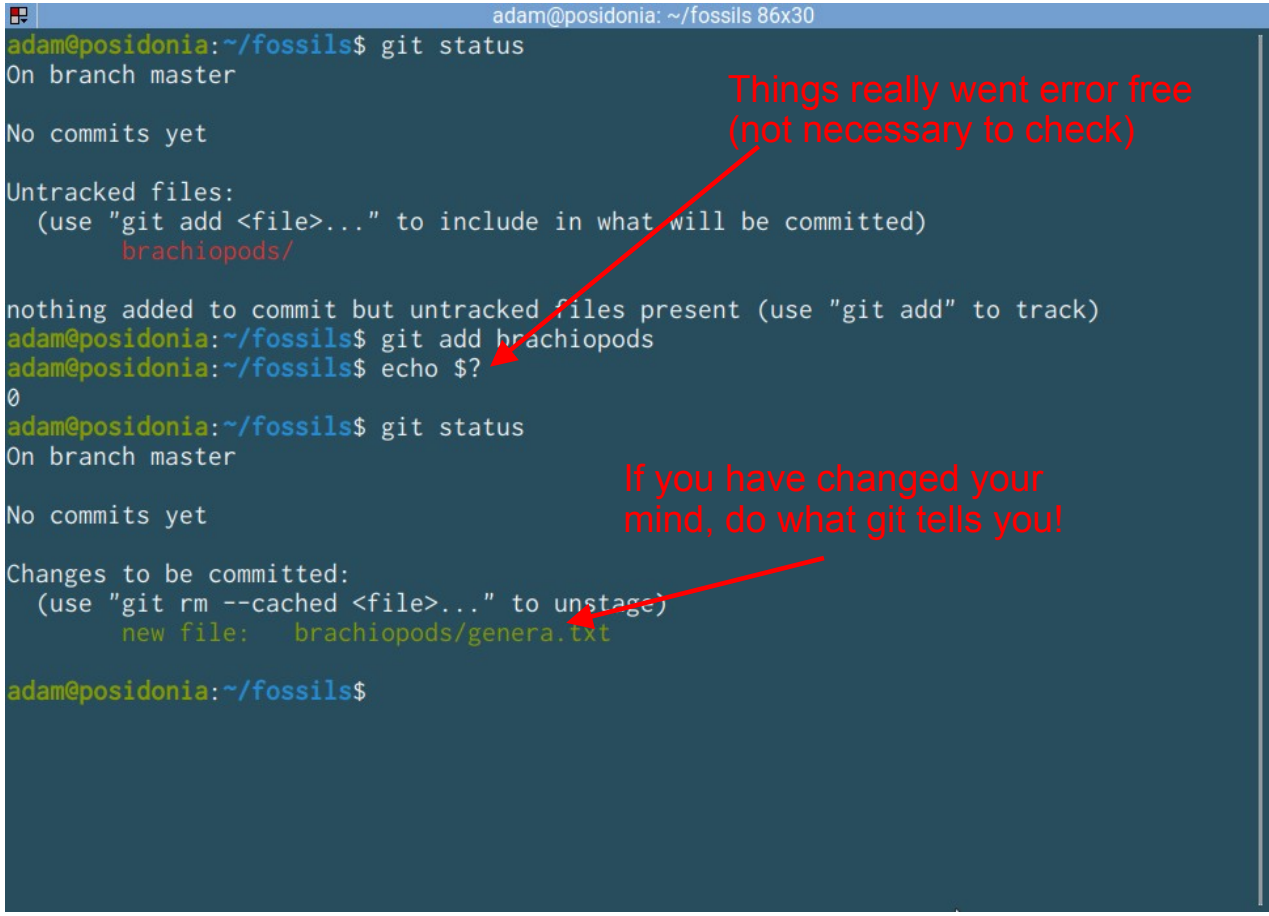
- Frequently this is an entire directory, including `.` (dot)
- If successful does not return anything, has to be checked with git status



`git status` (again)

Show status of repo

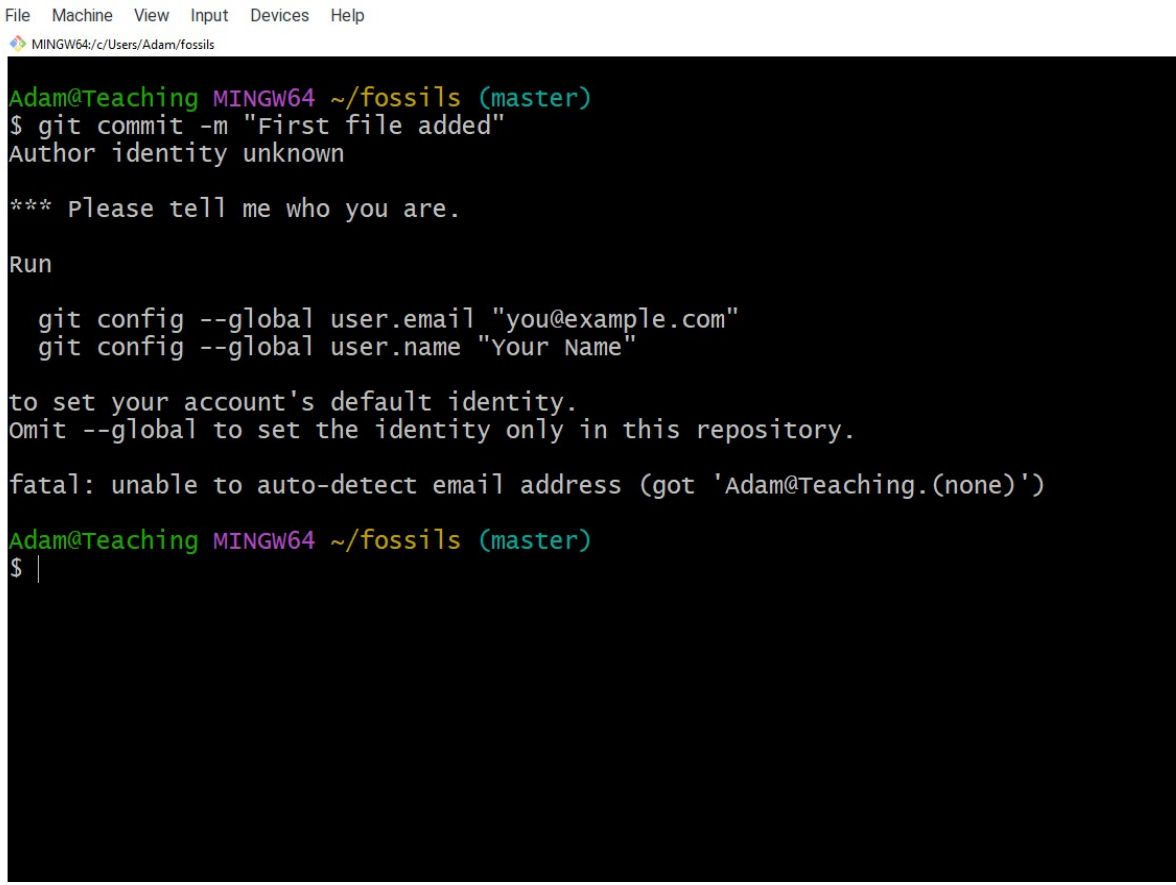
- There is just one file here which git finds.
- The file is now stages to be commited.



git␣commit␣-m␣<message>

First use not permitted without credentials!

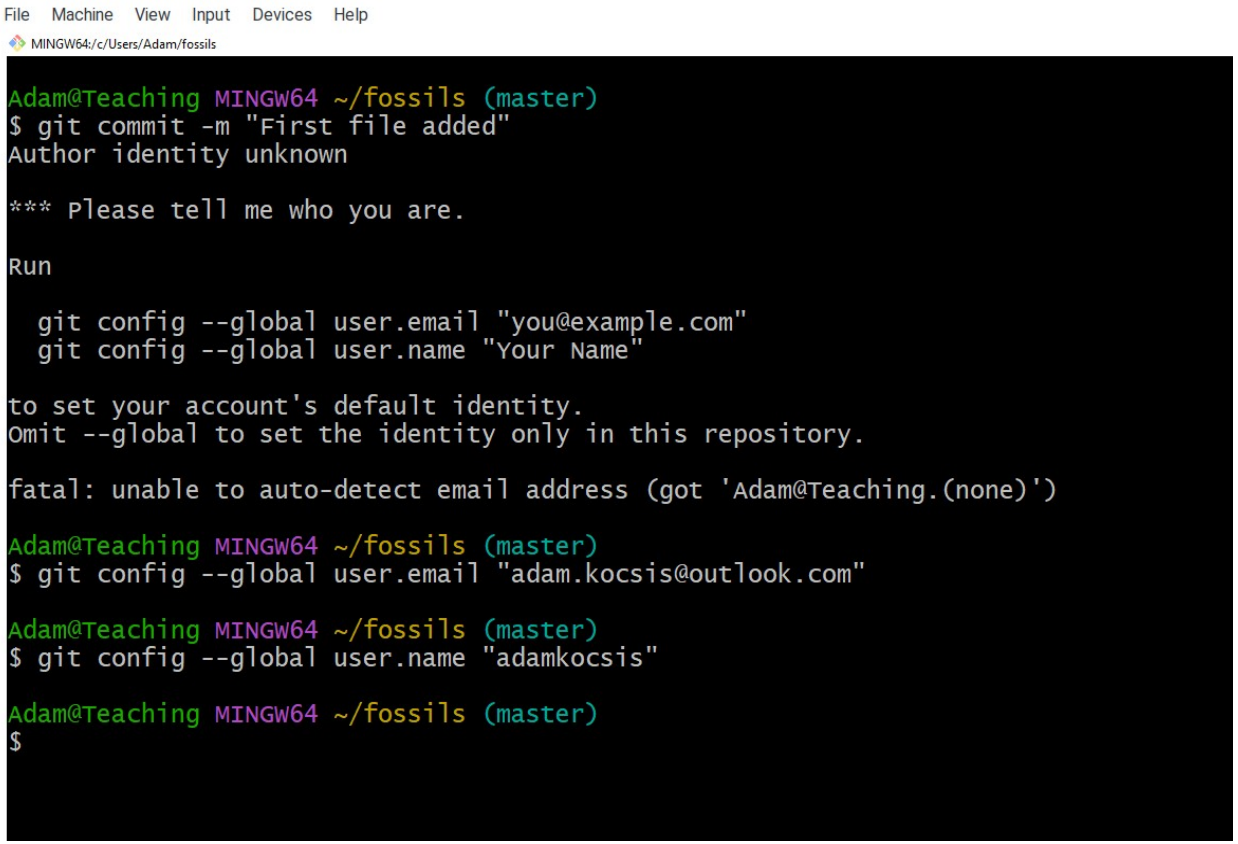
- You need to provide a user name and an email address with the git config command



`git␣config␣--global␣<what>␣<value>`

Configuring git

- `user.name` and `user.email`
- --global sets this for all your local git repositories
- Now you are ready to commit



`git commit -m <message>`

Now create a new commit

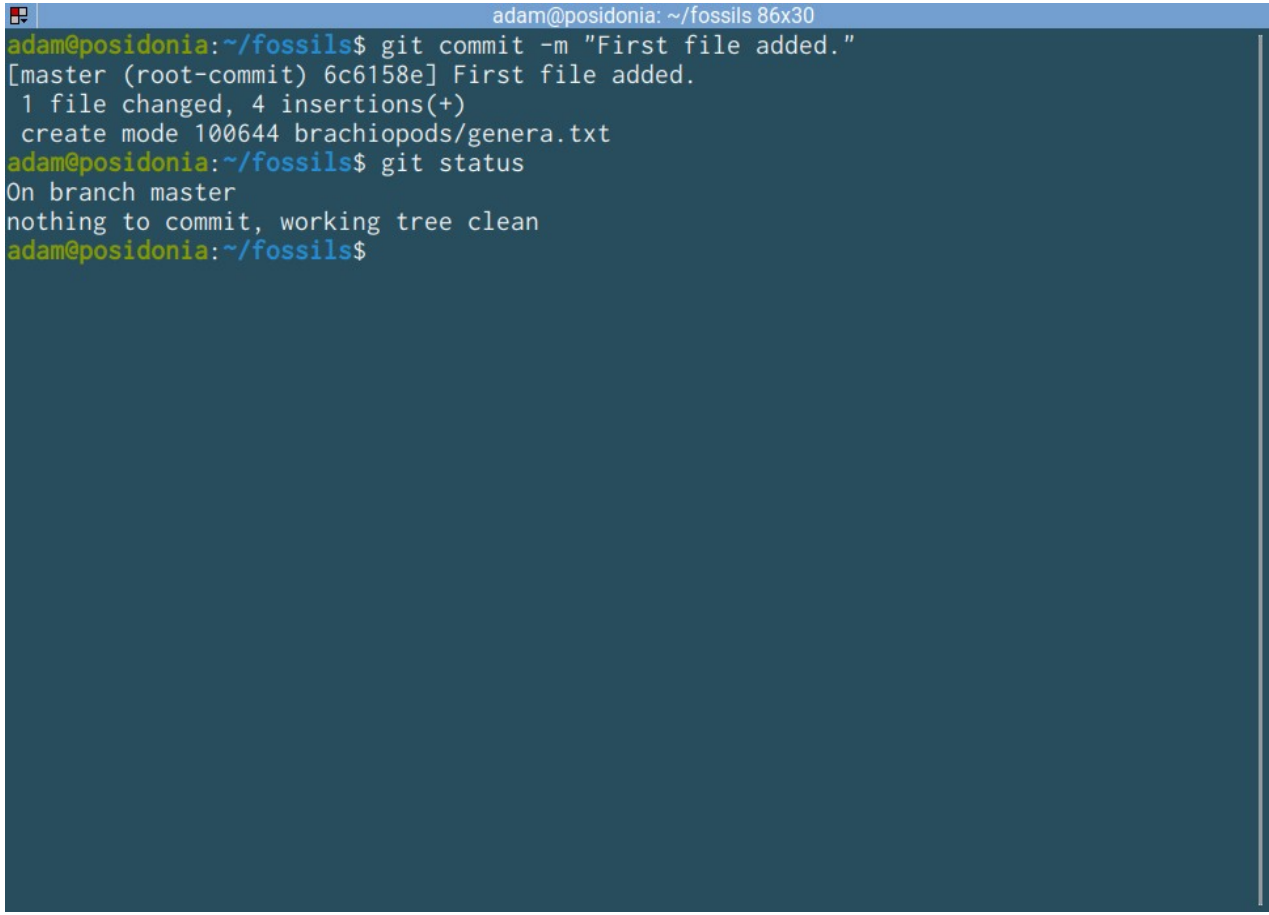
- Provide a message in **quotes**! This is the human readable description of what changed.
- Every commit gets a unique ‘hash’, a random set of characters that are used to identify unambiguously identify the commit



`git status` (yet again)

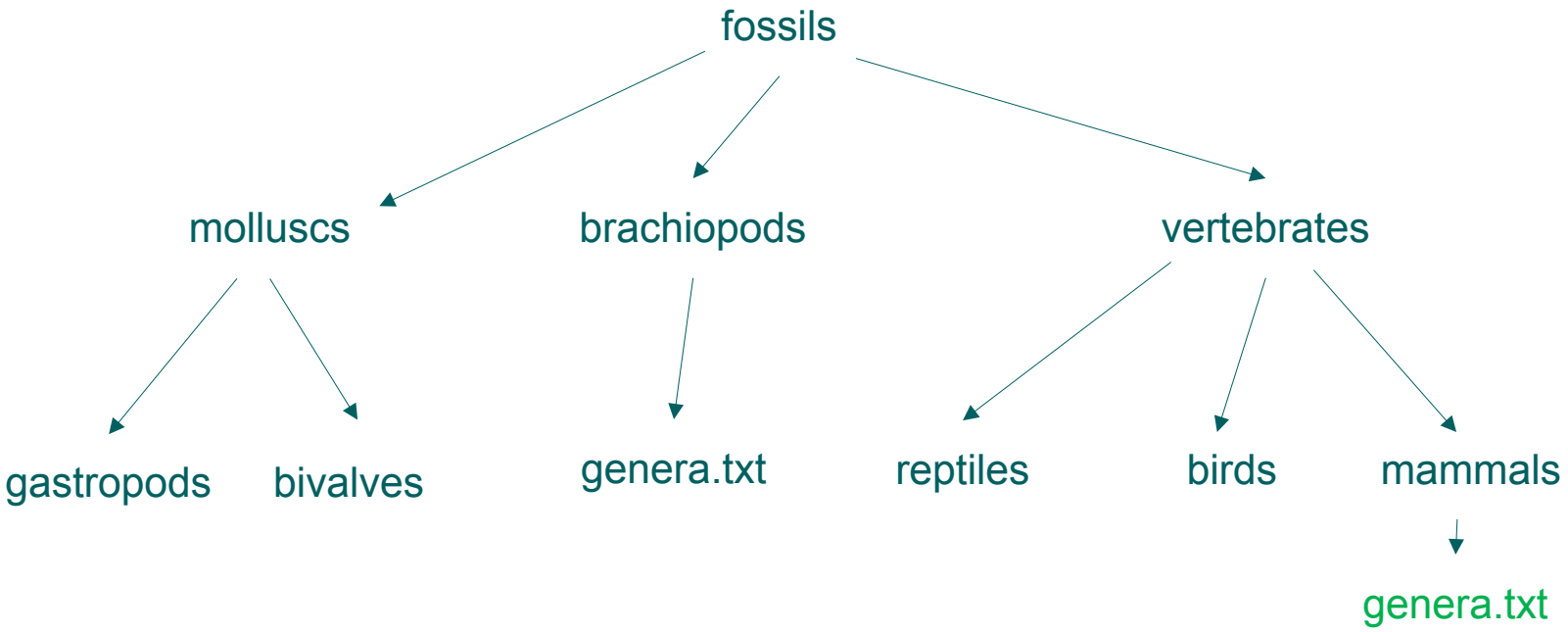
Nothing to be done.

- Create two new files

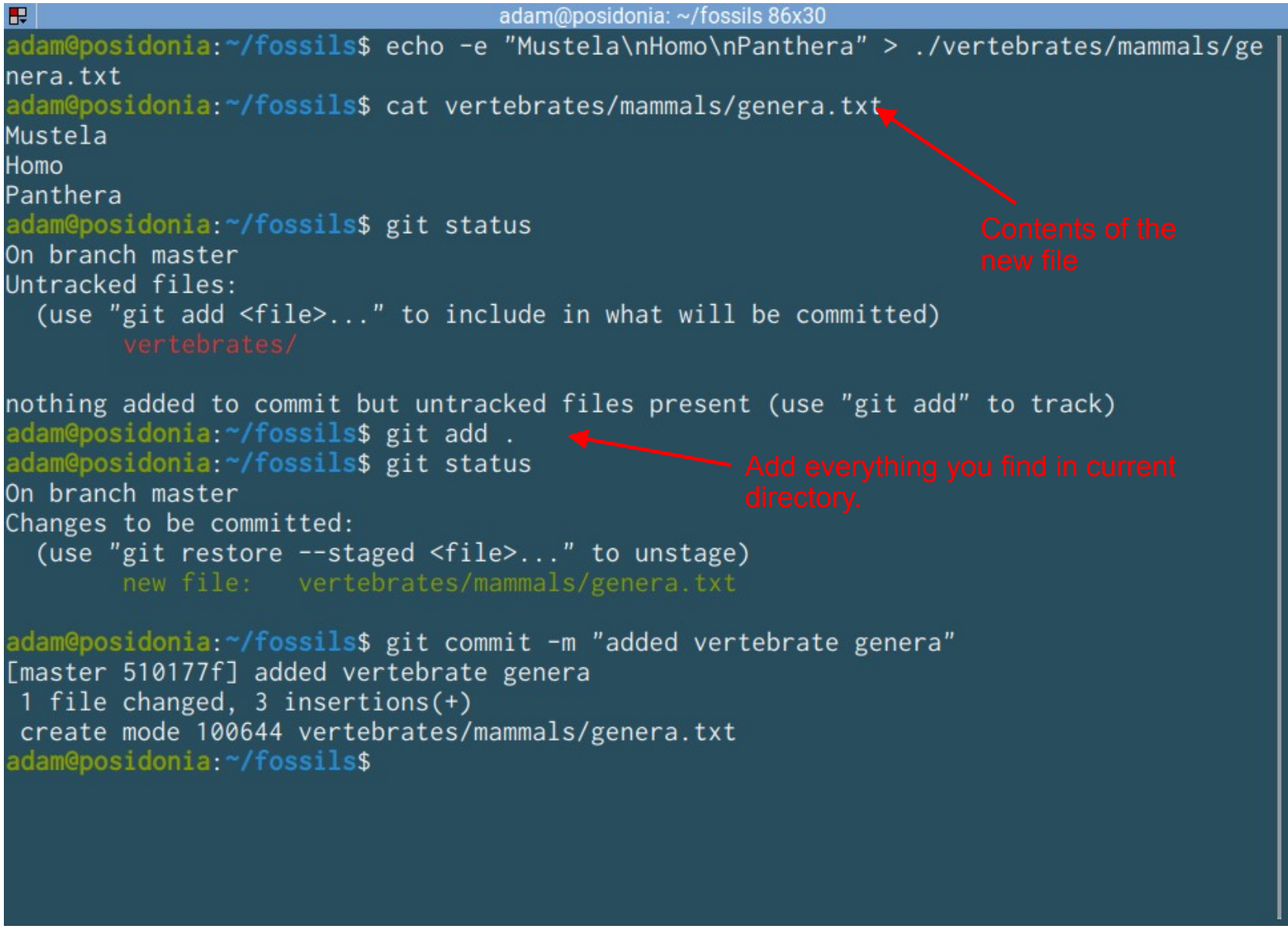


Demo!

1. Create a new file `genera.txt` in the `mammals` directory, and put the names of 3 mammalian genera in it!
2. Stage and commit the changes!

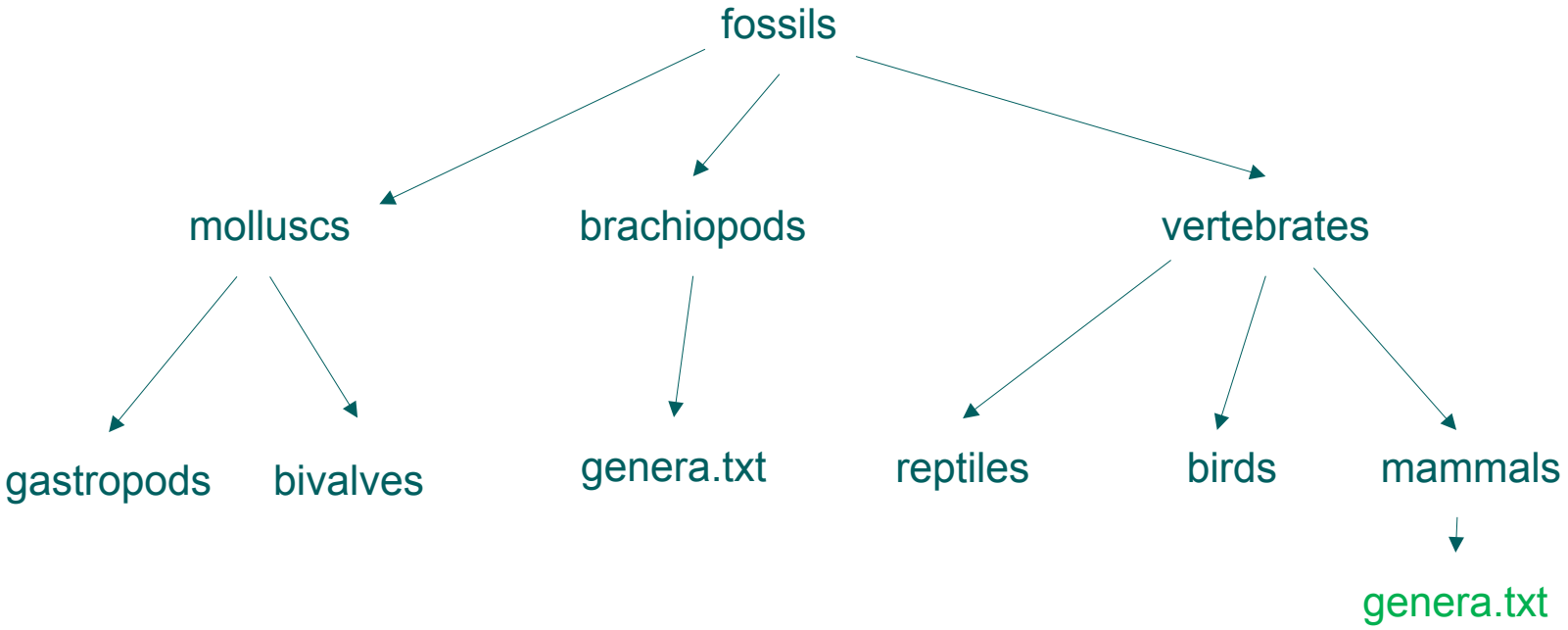


My solution


adam@posidonia: ~/fossils 86x30
adam@posidonia:~/fossils$ echo -e "Mustela\nHomo\nPanthera" > ./vertebrates/mammals/genera.txt
adam@posidonia:~/fossils$ cat vertebrates/mammals/genera.txt
Mustela
Homo
Panthera
adam@posidonia:~/fossils$ git status
On branch master
Untracked files:
(use "git add <file>..." to include in what will be committed)
vertebrates/
nothing added to commit but untracked files present (use "git add" to track)
adam@posidonia:~/fossils$ git add .
adam@posidonia:~/fossils$ git status
On branch master
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: vertebrates/mammals/genera.txt
adam@posidonia:~/fossils$ git commit -m "added vertebrate genera"
[master 510177f] added vertebrate genera
1 file changed, 3 insertions(+)
create mode 100644 vertebrates/mammals/genera.txt
adam@posidonia:~/fossils$
Contents of the new file
Add everything you find in current directory.

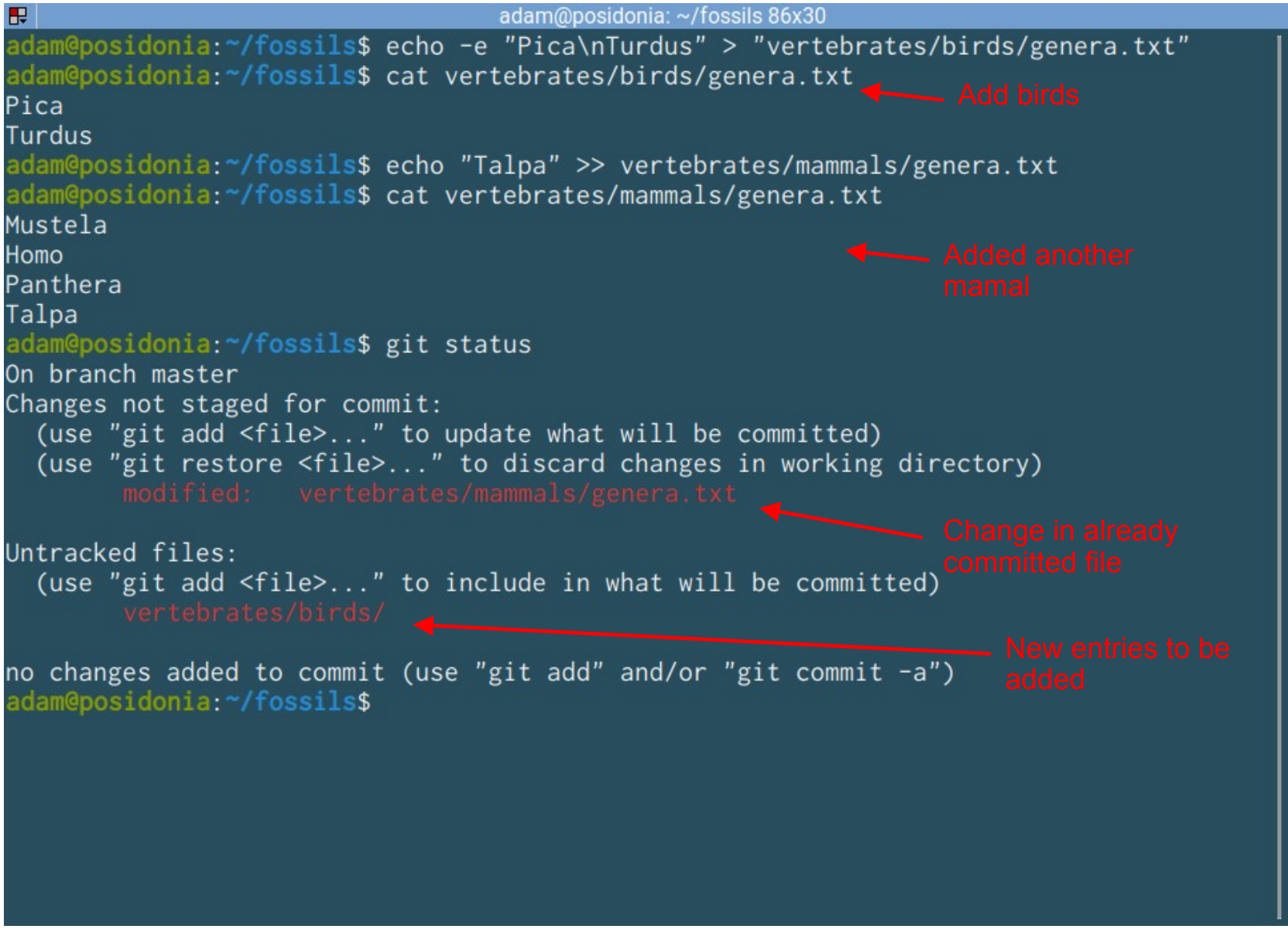
Demo!

1. Create a new file `genera.txt` in the `birds` directory, and put the names of 2 bird genera in it!
2. Add another genus to the mammals.
3. Try to commit only the birds!



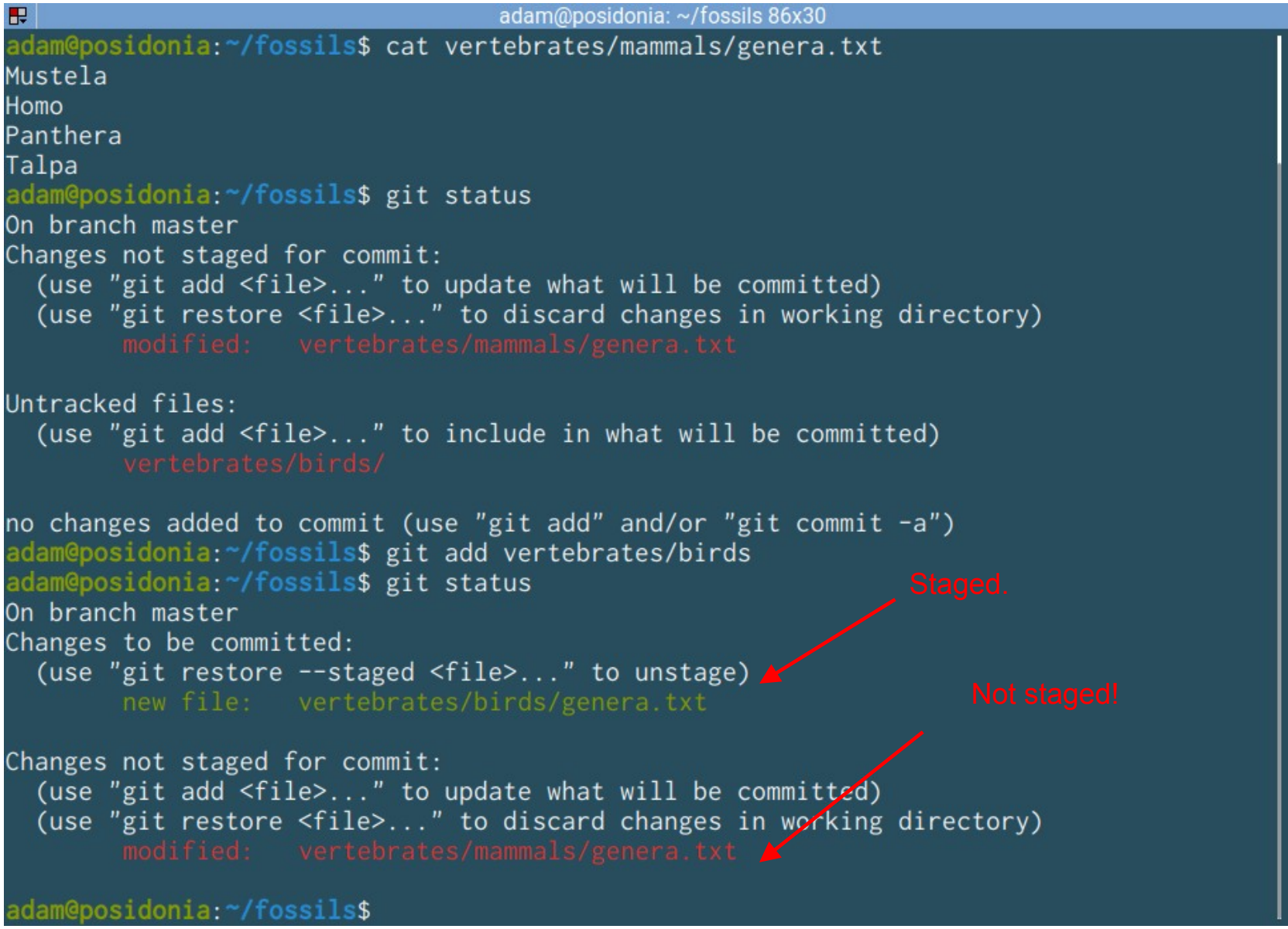
My solution

1. **Make the changes.**



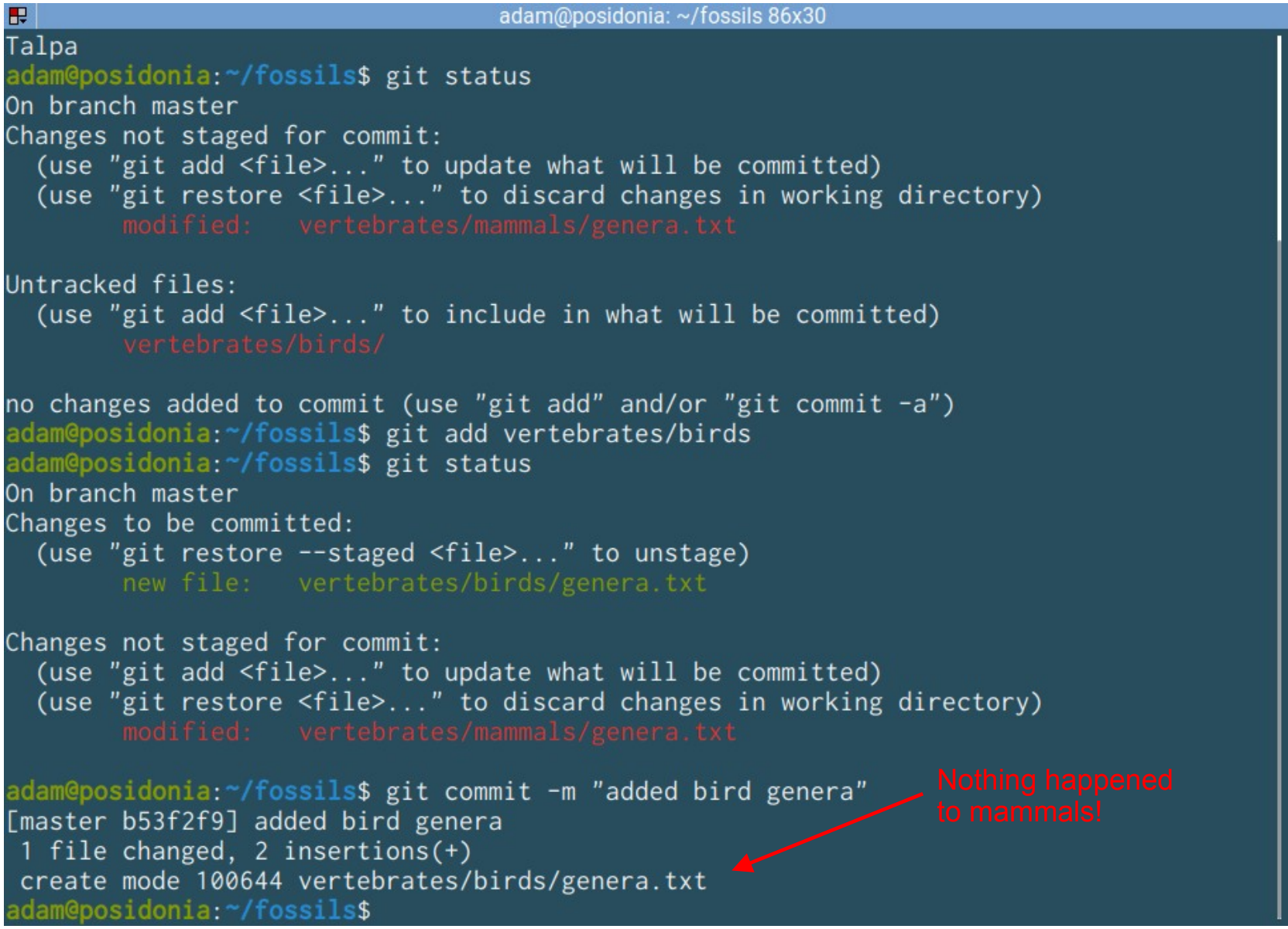
My solution

2. Stage only the birds.



My solution

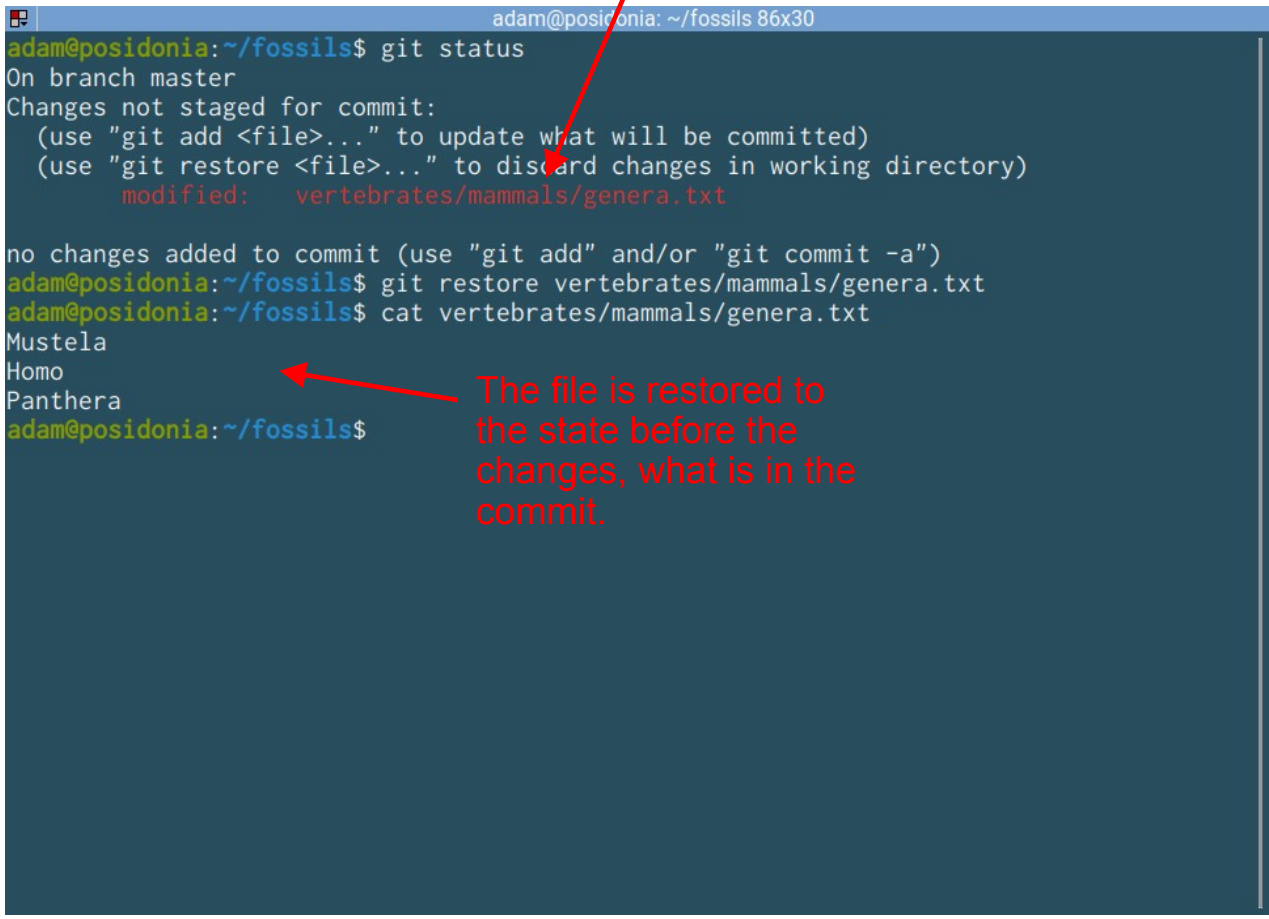
3. Stage only the birds.



git␣restore␣<path>

Discarding changes from previous commit

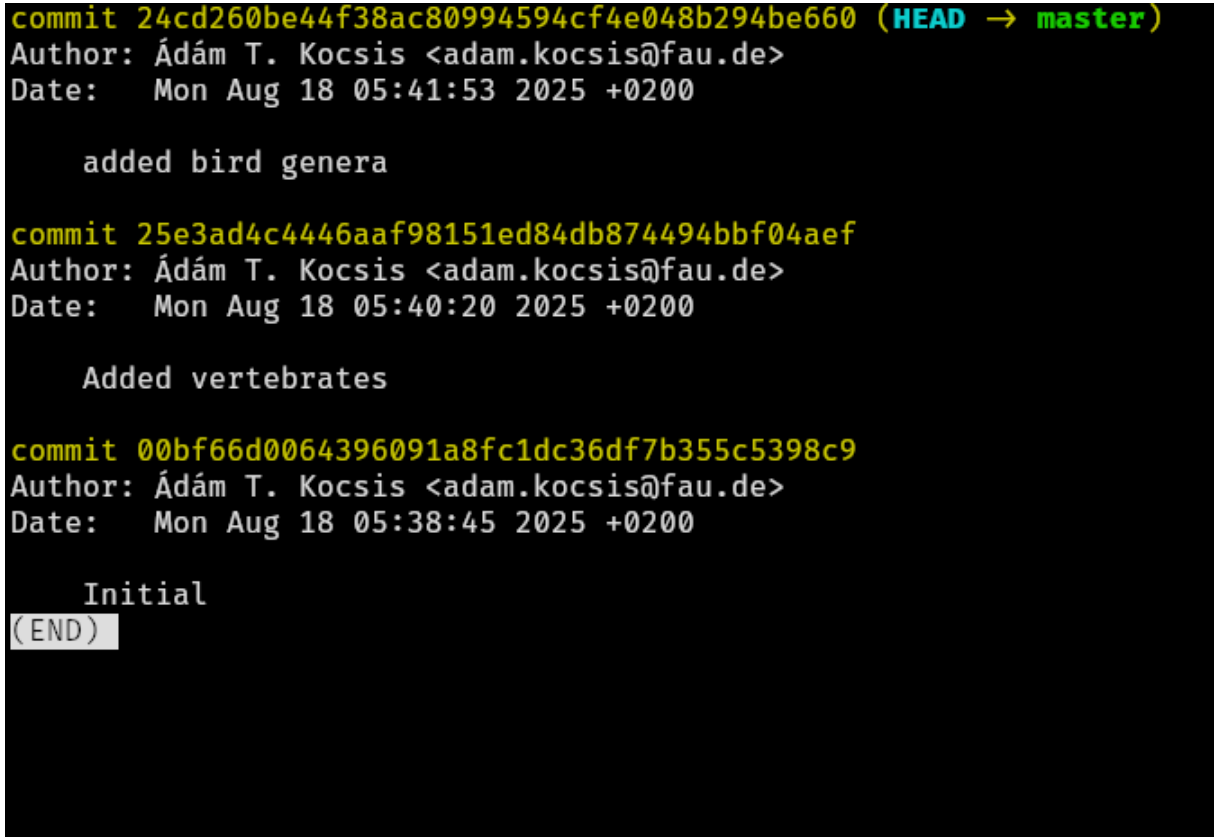
- We can commit the new mammal or discard it.
- You can correct unintended changes with this.
- What about even older changes?



git log

Accessing repository history

- Commits listed out in sequence.
- Every commit has a hexadecimal hash id
- Lists out metadata of the commit
- The ‘HEAD’ indicates where we are in the history



git checkout <hash>

Exploring history

- No need to write the entire hash
- Moves back in time so we can explore the past states
- `git checkout <branch name>` moves the head back to the most recent commit

birds/genera.txt is not there! →

Move head back →

birds/genera.txt reappears! →

```
 adam@TheStuff  ~/work/fossils  master  git log
 adam@TheStuff  ~/work/fossils  master  git checkout 25e3
Note: switching to '25e3'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 25e3ad4 Added vertebrates
 adam@TheStuff  ~/work/fossils  - 25e3ad4  ls vertebrates
mammals  reptiles
 adam@TheStuff  ~/work/fossils  - 25e3ad4  git checkout master
Previous HEAD position was 25e3ad4 Added vertebrates
Switched to branch 'master'
 adam@TheStuff  ~/work/fossils  master  ls vertebrates
birds  mammals  reptiles
 adam@TheStuff  ~/work/fossils  master
```

GitHub

and GitHub

GitHub

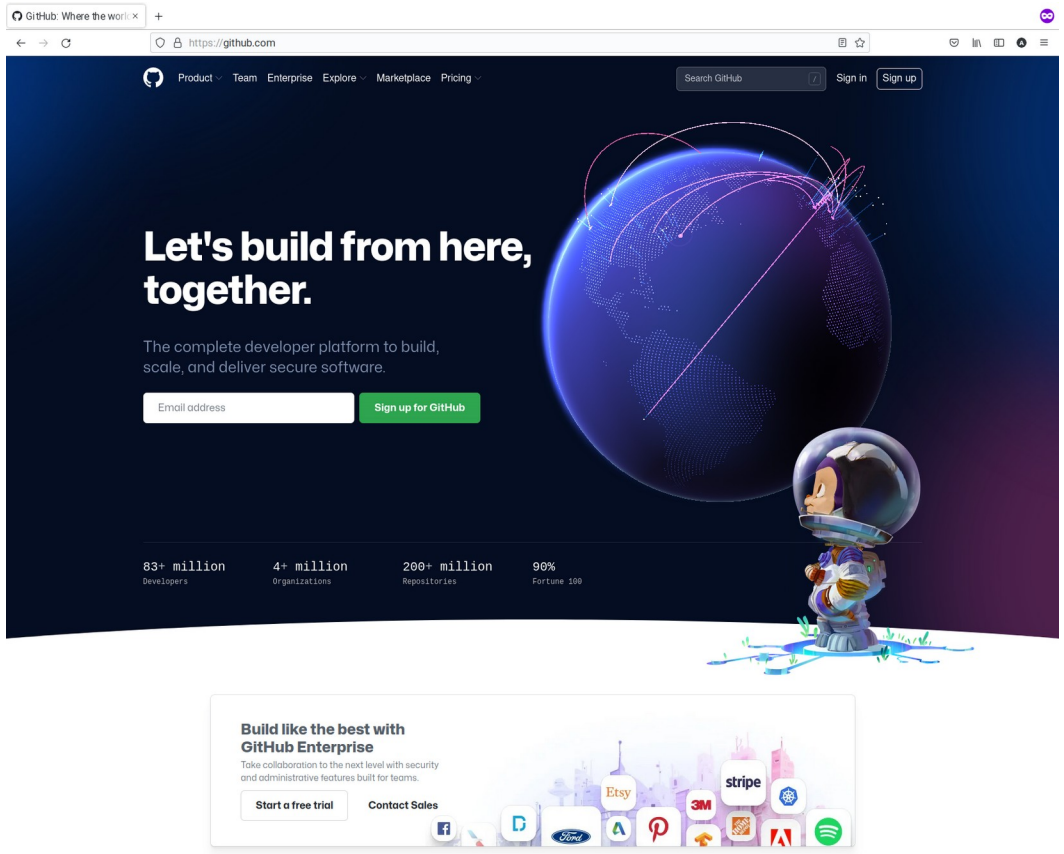
Where the world builds software (2008-)

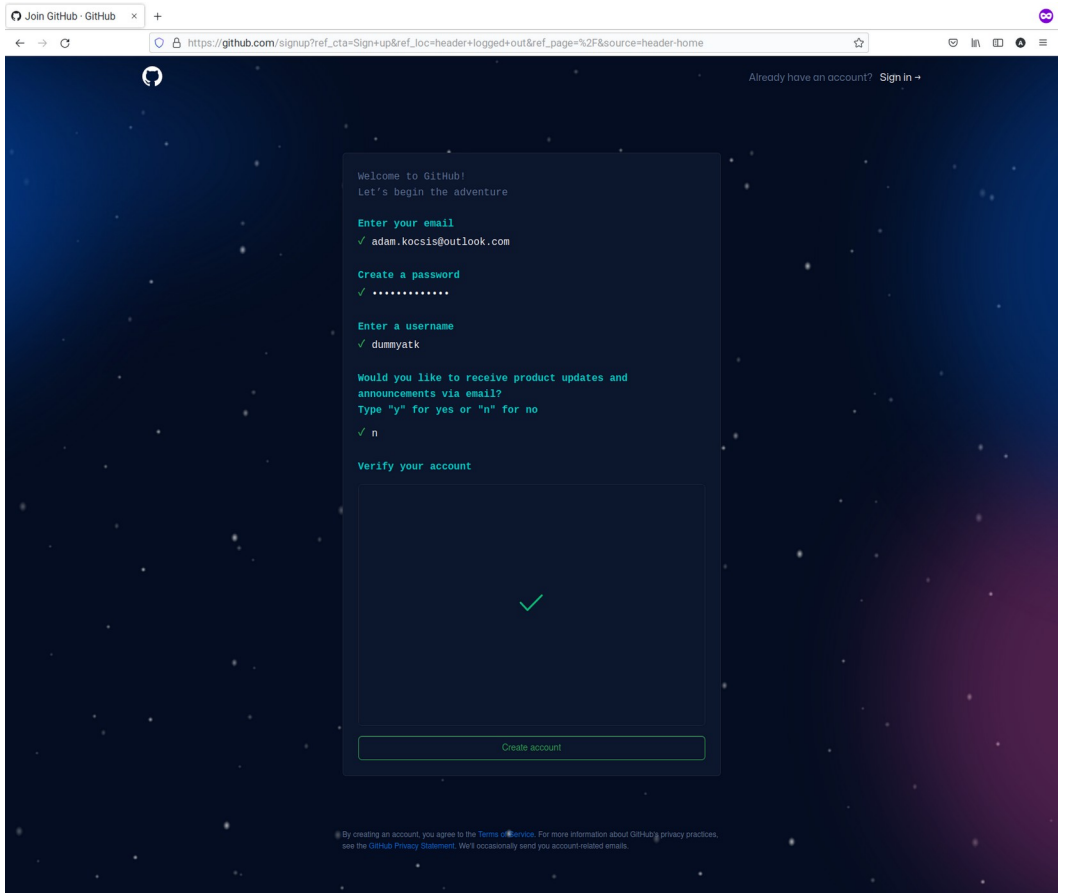


- Open source software development platform, places to store and share git repositories
- Currently owned by Microsoft
- Applications, packages, plugins, webpages and many more!
- Free and private repositories.
- Tools for CI/CD (continuous integration/deployment)

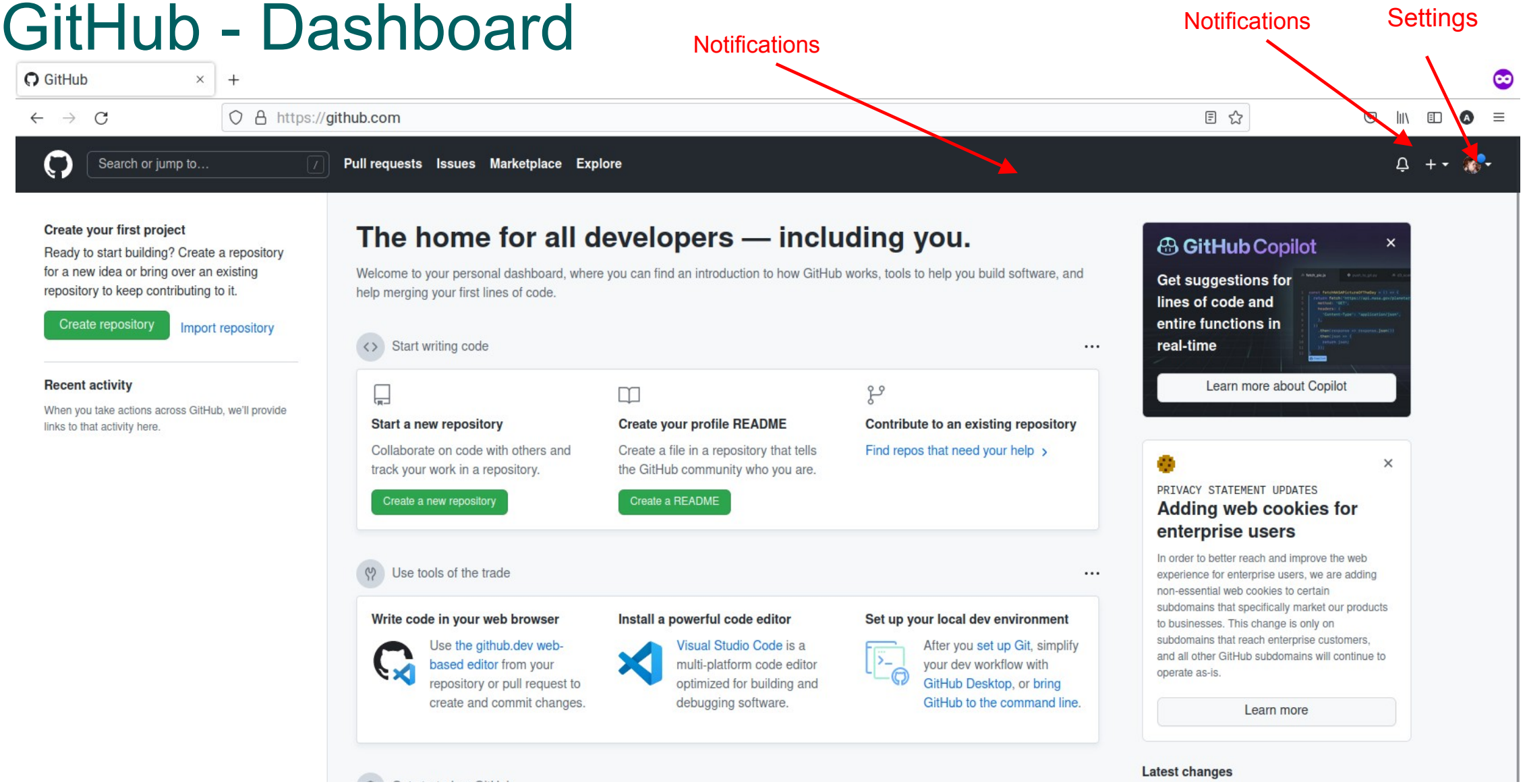
GitHub

Sign up if you haven’t yet!

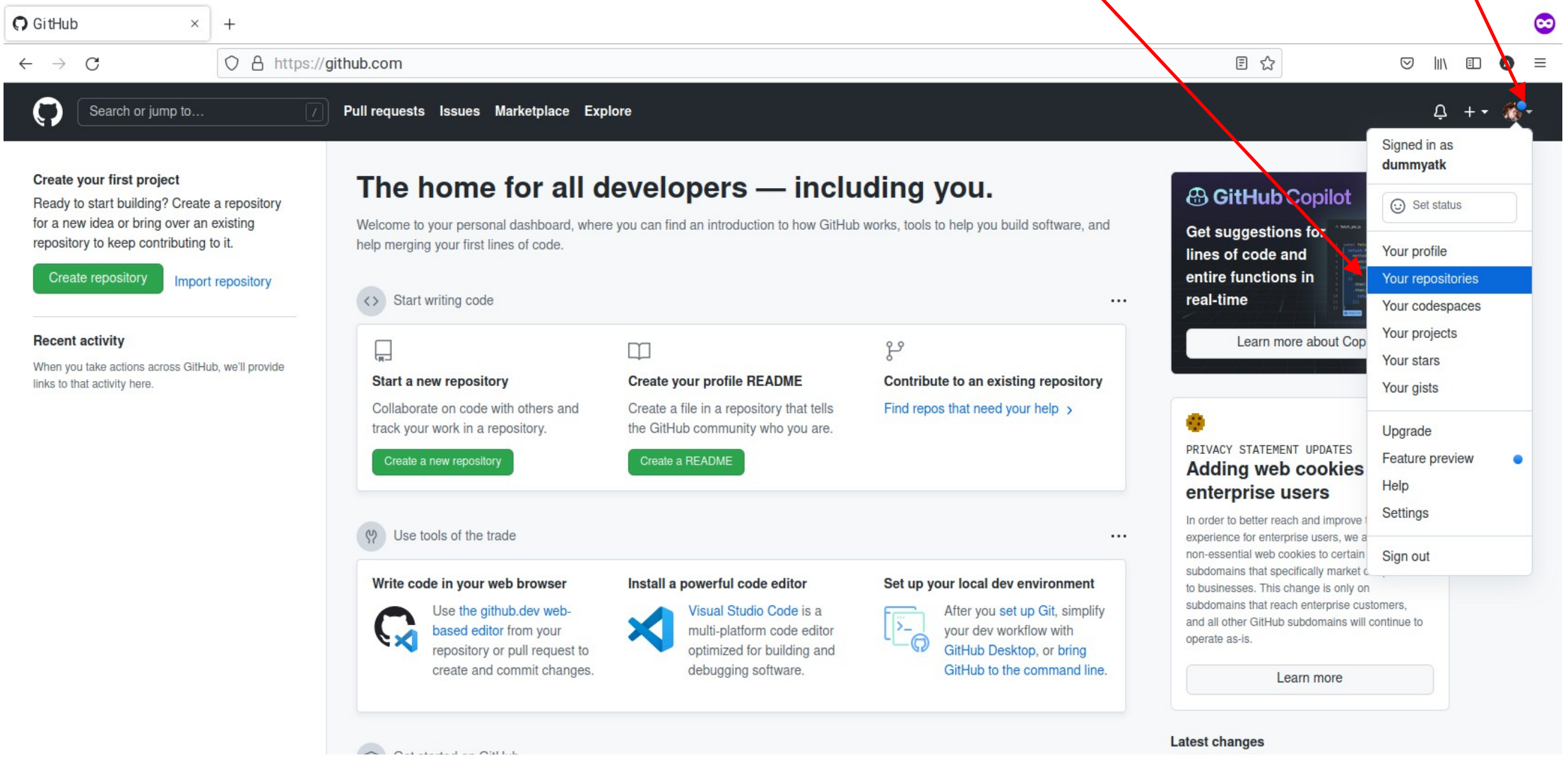




GitHub - Dashboard

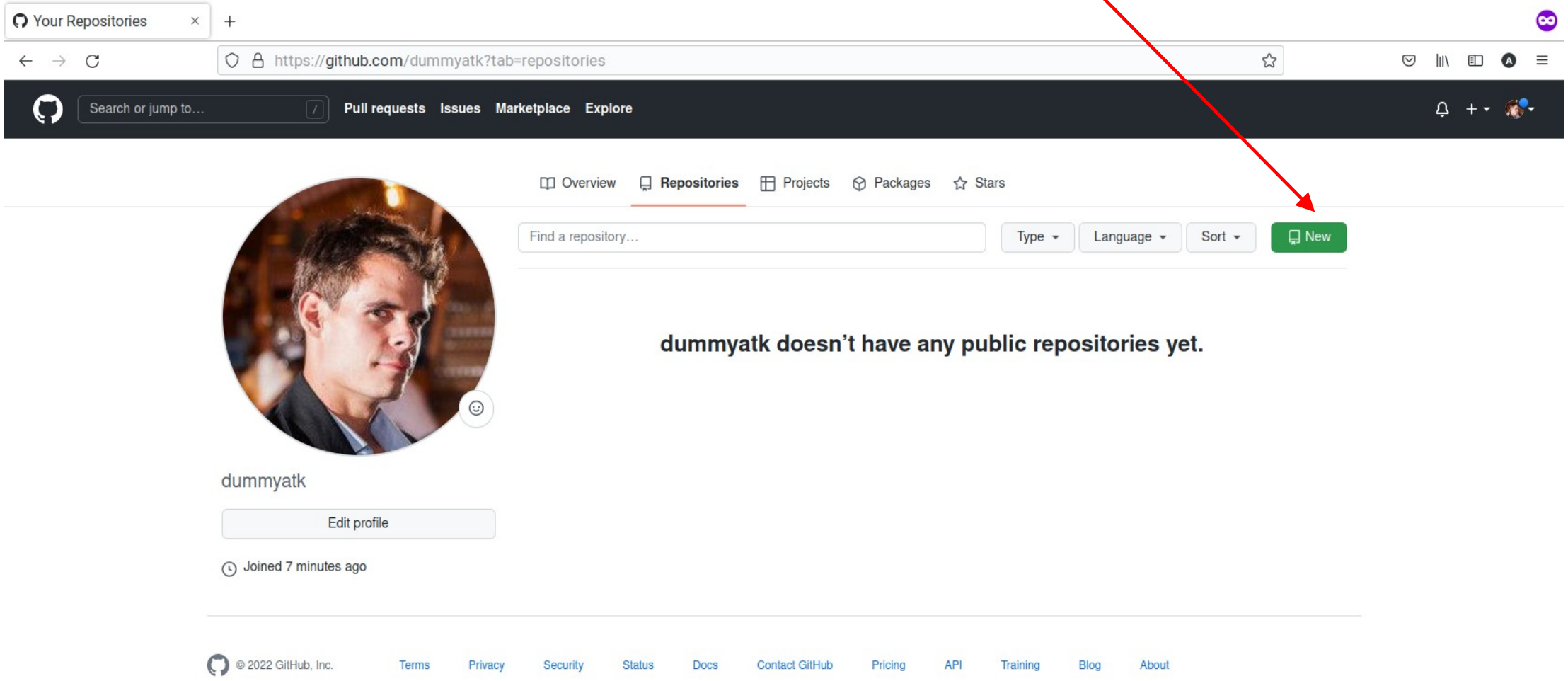


GitHub - Dashboard

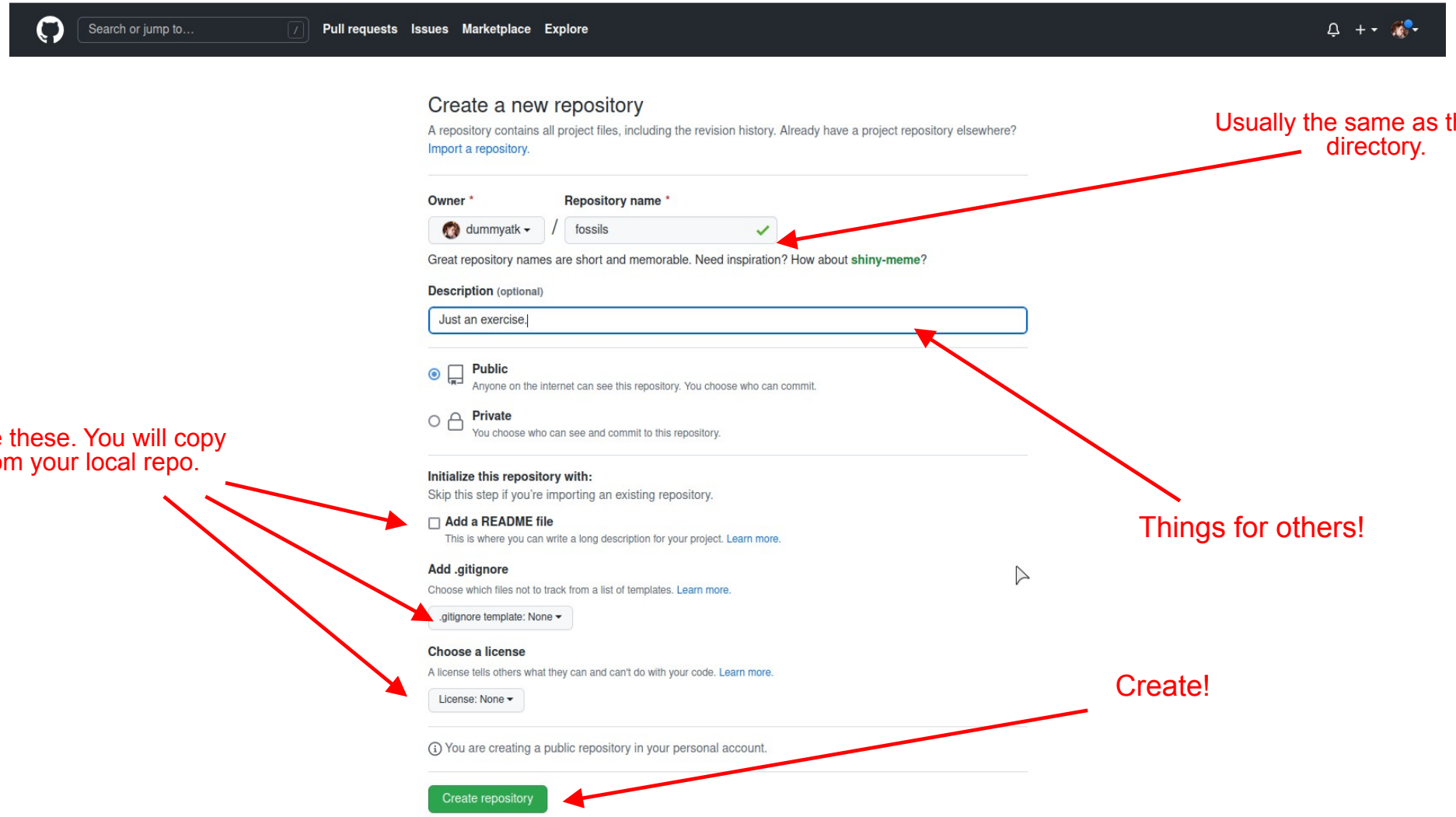


GitHub – Creating a new repo

Make a new repo



GitHub – Creating a new repo



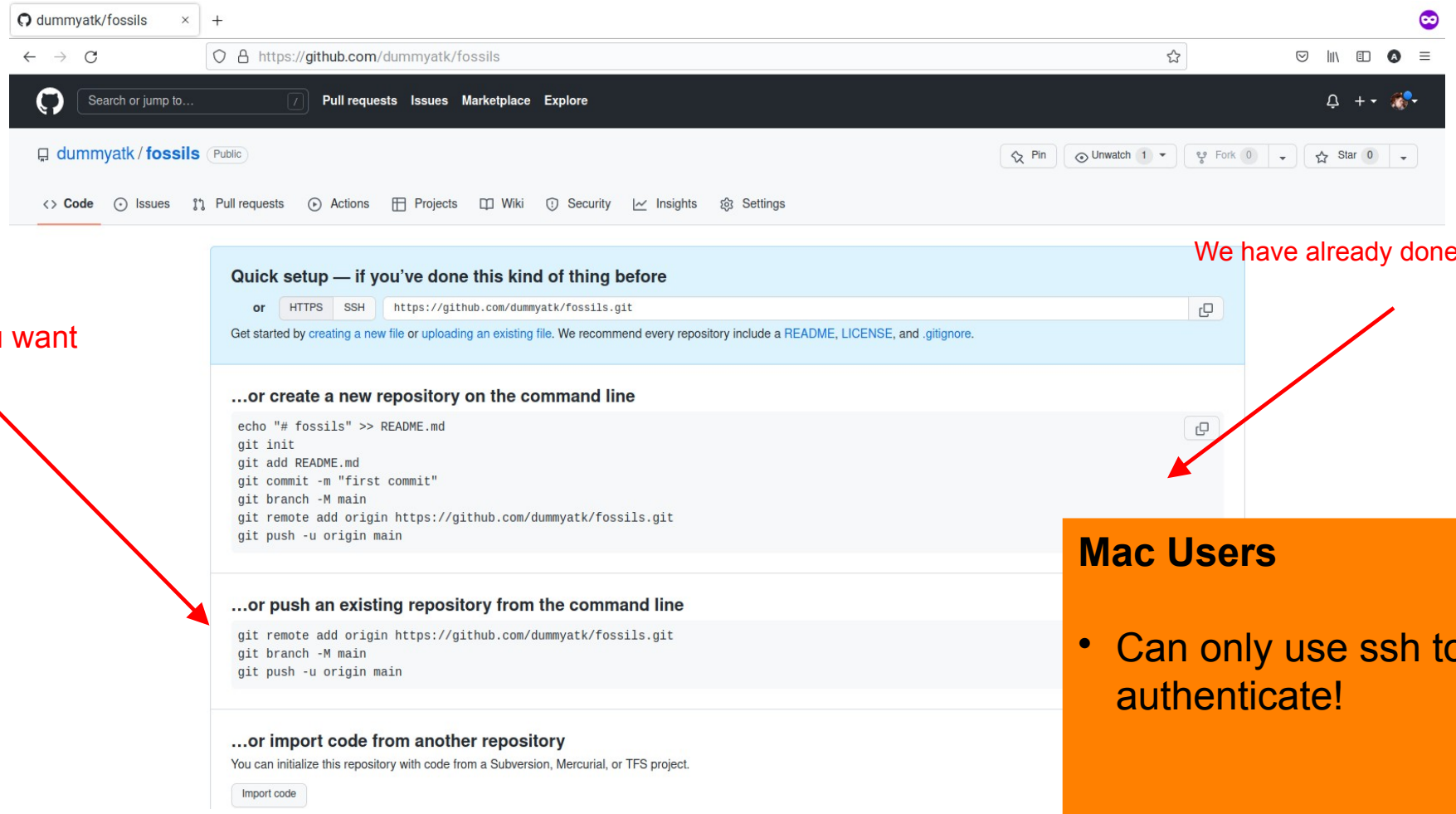
Usually the same as the local directory.

Things for others!

Do not change these. You will copy files over from your local repo.

Create!

GitHub – The fresh empty repo



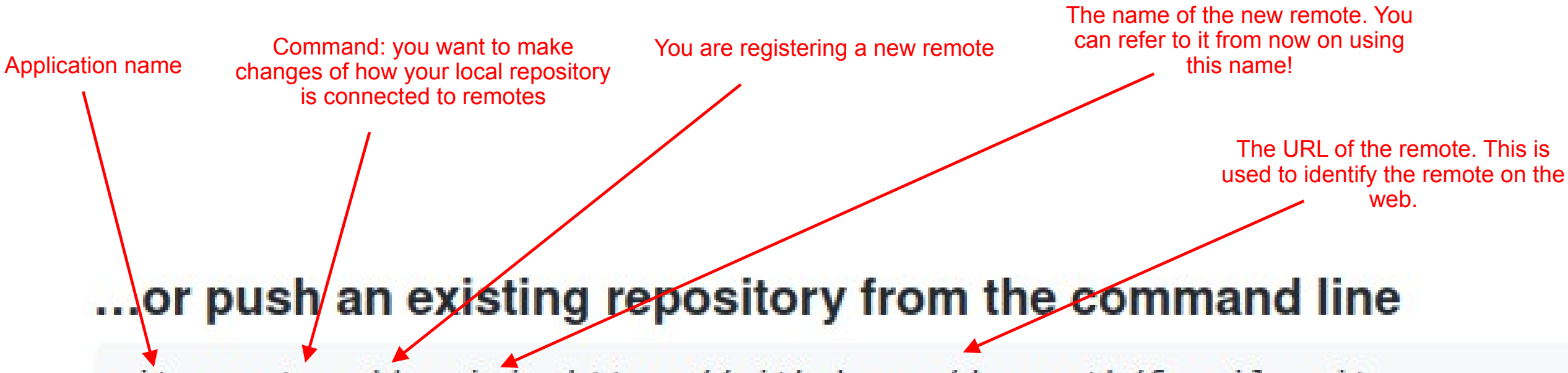
This is what you want

We have already done this mostly

Mac Users

- Can only use ssh to authenticate!

GitHub – Add new remote



...or push an existing repository from the command line

```
git remote add origin https://github.com/dummyatk/fossils.git
git branch -M main
git push -u origin main
```

GitHub – Rename current branch to main

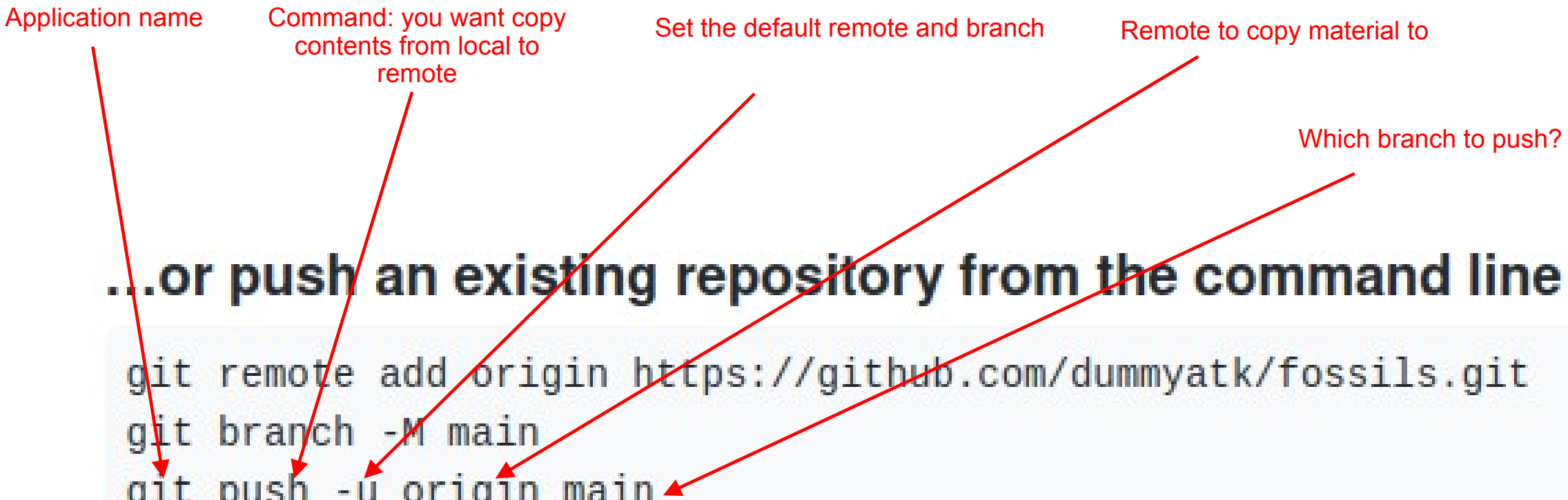


...or push an existing repository from the command line

```
git remote add origin https://github.com/dummyatk/fossils.git
git branch -M main
git push -u origin main
```

For political reasons, GitHub does not allow the use of the name master, hence this extra step.

GitHub – Pushing contents of branch to remote

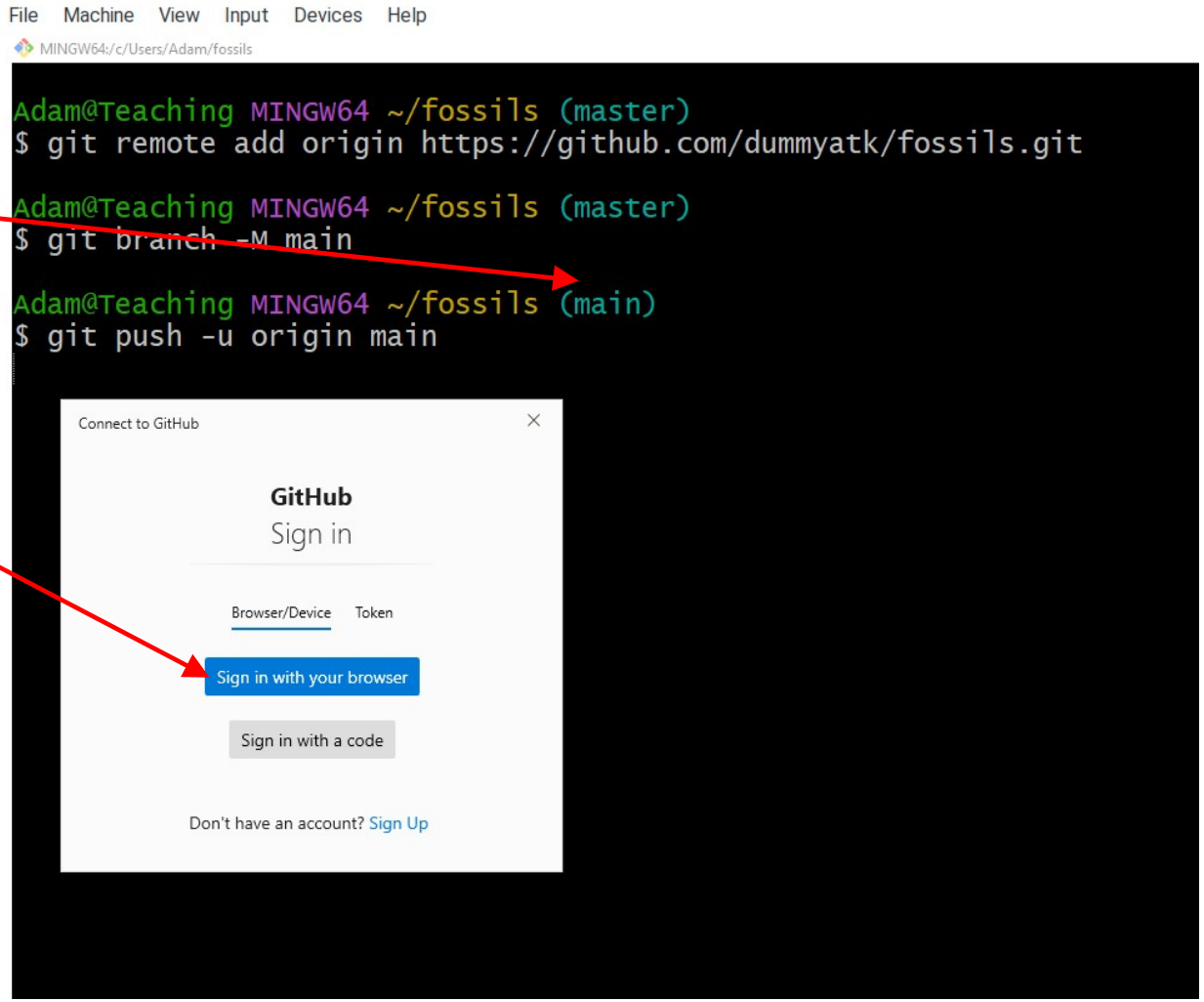


GitHub will ask for your credentials

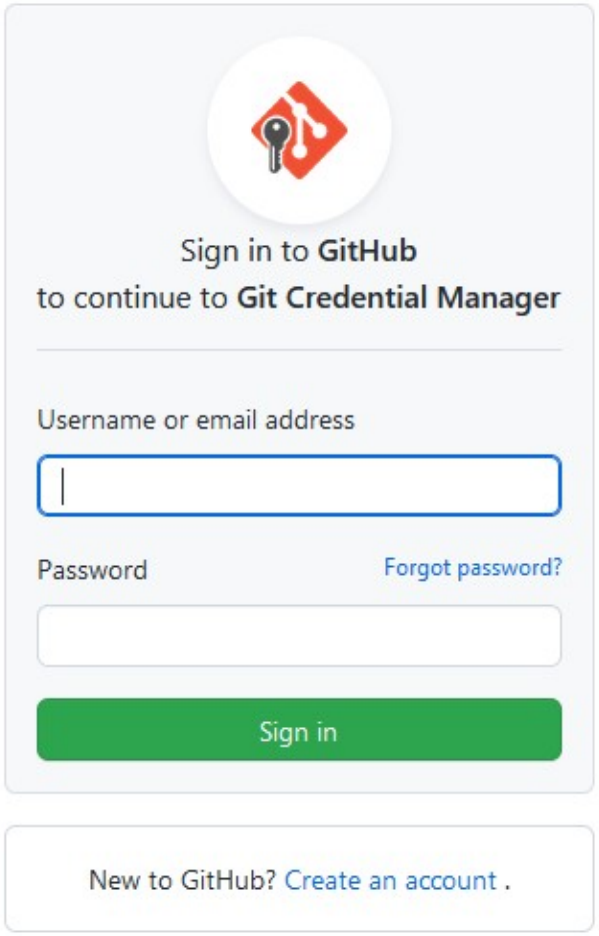
GitHub – Executing this and signing in on windows

Note branch name change

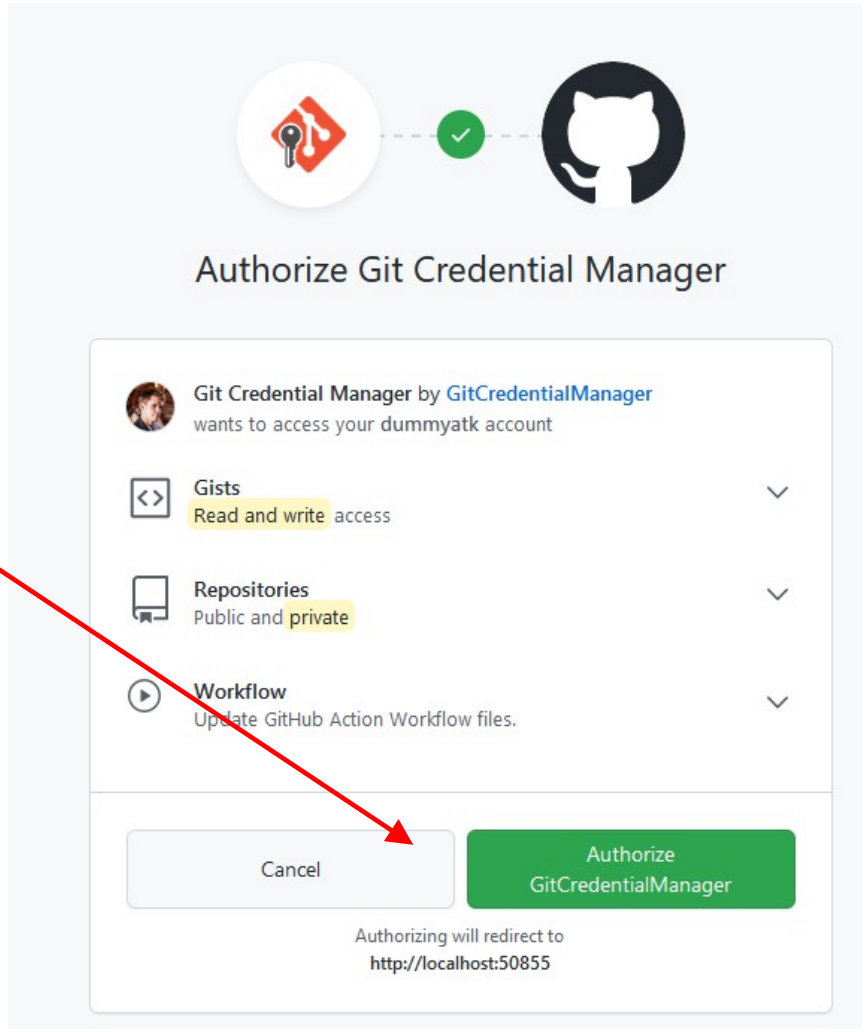
Most interactive sign in option available on Windows



GitHub – Executing this and signing in on windows



This is what you want

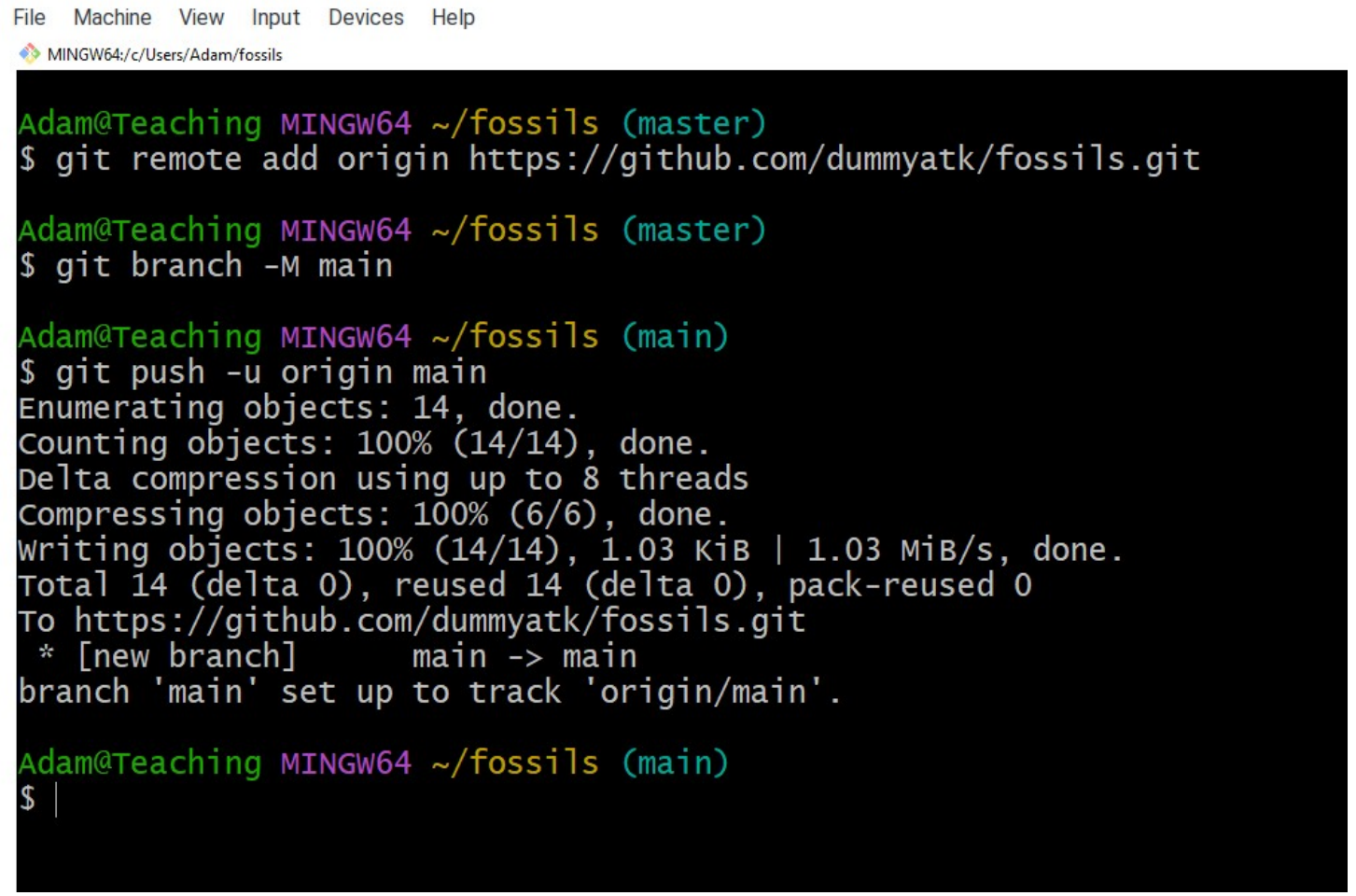


GitHub – Successful push

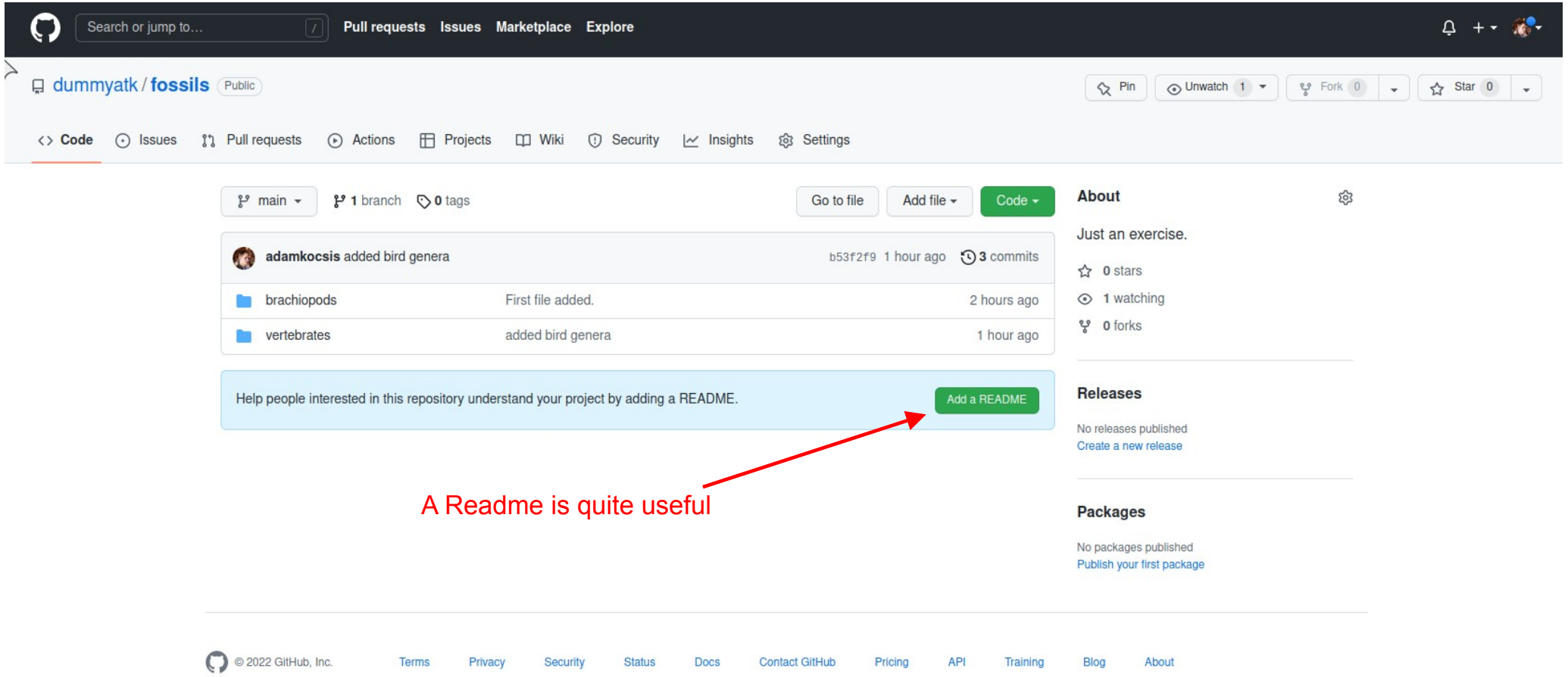
Transfer stats

New branch main is created on remote

And is now in sync with local

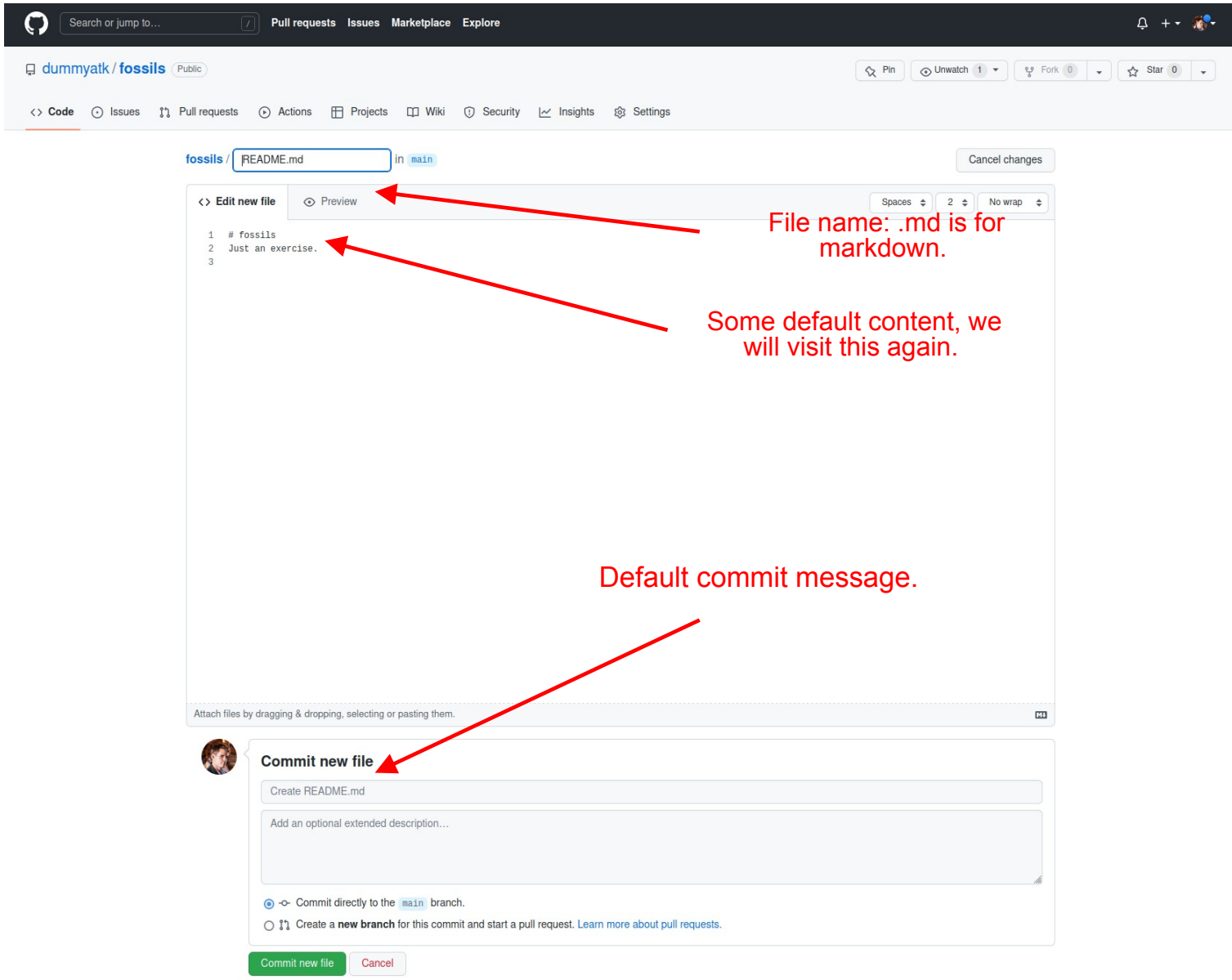


GitHub – Successful push

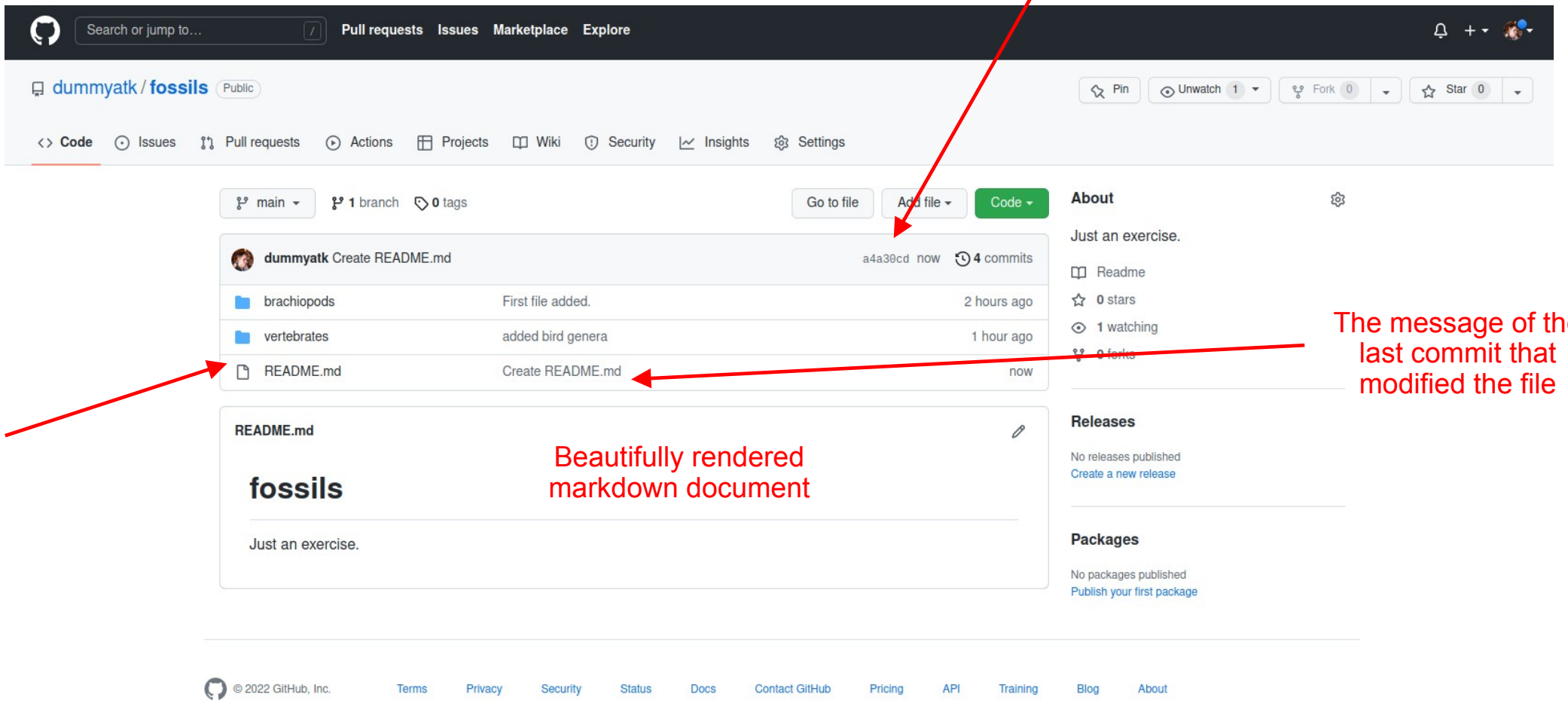


Writing a Readme

- Default format is **markdown (later)**
- You can work on files using GitHub's interface
- Save the defaults, by clicking on the green button
- Note that you are technically creating a new commit!



GitHub – Changing the remote



git␣pull␣<remote>␣<branch>

Pull changes from remote

- Just because you changed something on the remote server does not make things magically appear locally
- You have to pull the contents of the remote to have the new file that you just created!

