Everything you always wanted to know about git*

*but were afraid to ask

git

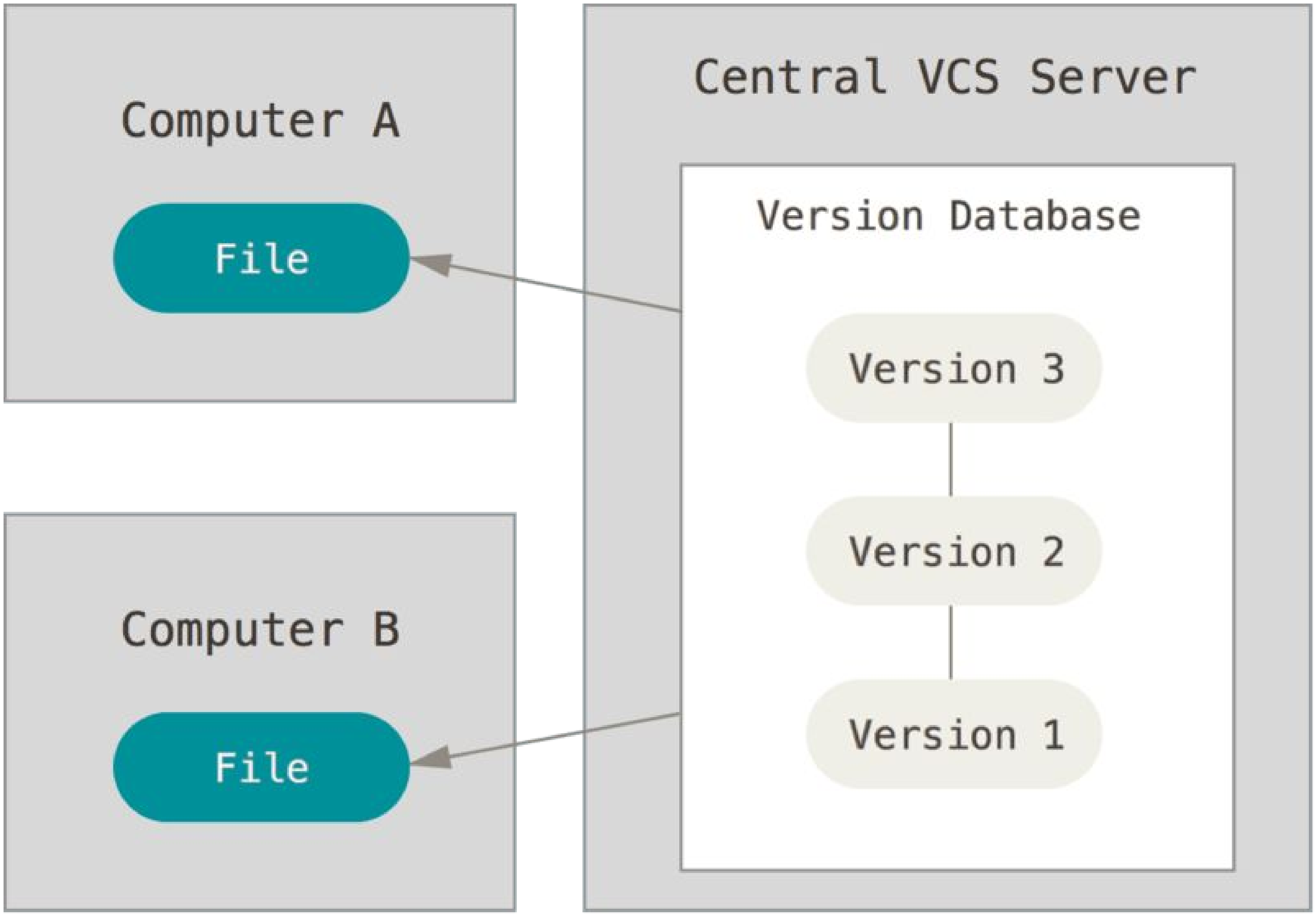
Distributed version control system (DVCS)

➔ Version control:
- keeps track of changes..

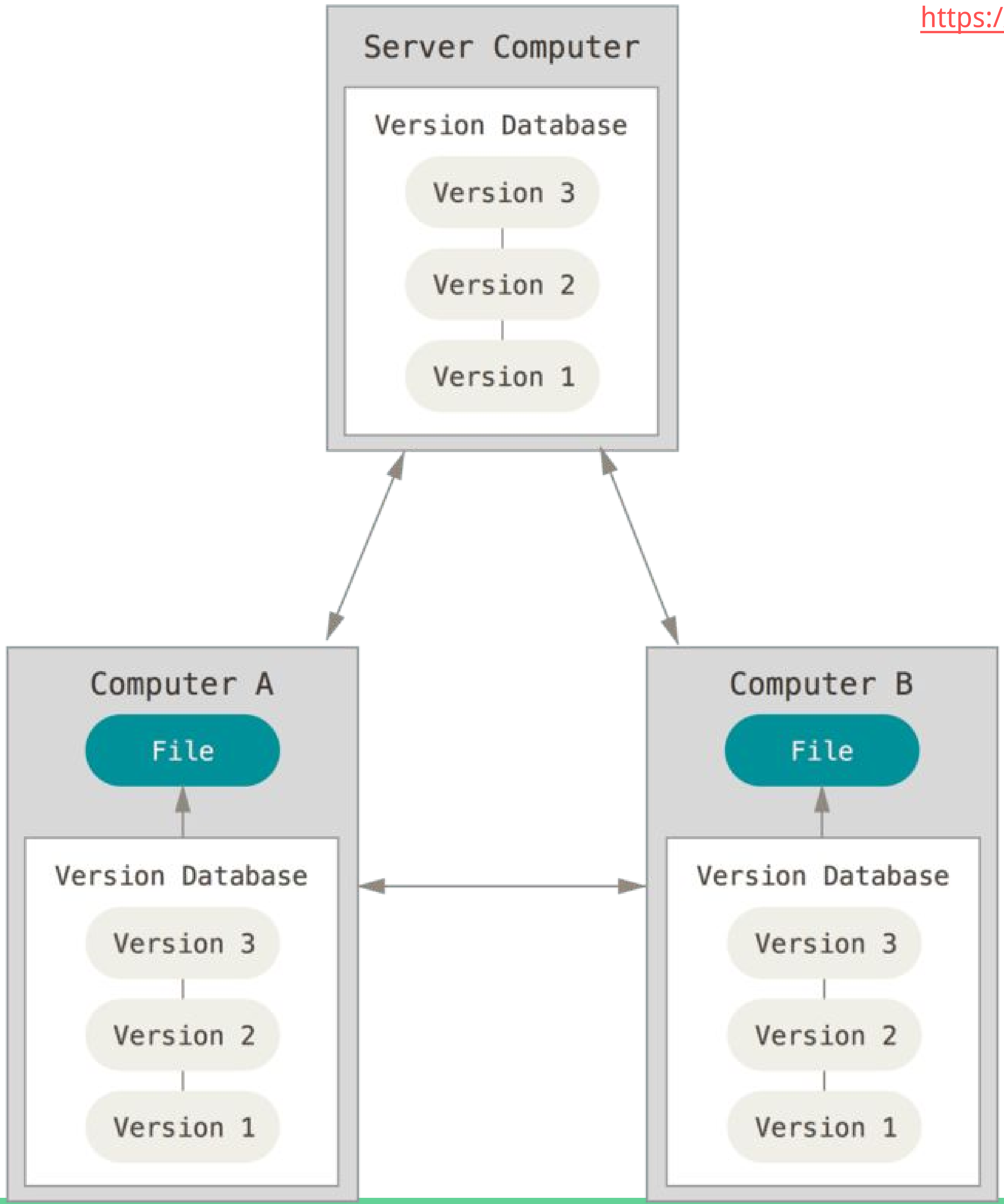
  ..if you tell it to
- Possible to restore old states / switch between states of the project

➔ Distributed:
- Main distinction to older VCSs
- No central server that stores information
- Every git repository stores history information locally
- Commits can be done without connection to the server
- Makes git extremely powerful.. but also adds complexity that user has to cope with

➔ Developed by Linus Torvalds for Linux Kernel development



from: “Pro Git” online at https://git-scm.com/book/en/v2/ (license CC BY-NC-SA 3.0)


Server Computer
Version Database
Version 3
Version 2
Version 1
Computer A
File
Version Database
Version 3
Version 2
Version 1
Computer B
File
Version Database
Version 3
Version 2
Version 1

Sounds complicated.. Why bother?

Looks familiar..?

```
$ ls -lrt
-rwx------ 1 megies users   3946 Oct 29  2015 data_processing_2016-10_fix_tapering.py
-rwx------ 1 megies users   1500 Jul 24  2015 data_processing_2016-10.py
-rwx------ 1 megies users   1451 Dec 16  2014 data_processing_new2_fixed.py
-rwx------ 1 megies users   1028 Jul 22  2015 data_processing_new2_final.py
-rwx------ 1 megies users   4096 Oct 14  2015 data_processing_new2_final2.py
-rwx------ 1 megies users   3450 Dec  8  2014 data_processing_new.py
-rwx------ 1 megies users   2211 Jan 27  2016 data_processing_new_with_data_download.py
-rwx------ 1 megies users   8366 Dec 15  2014 data_processing.py
-rw------- 1 megies users   9479 Dec  7  2015 data_processing_paper_version.py
-rw------- 1 megies users   9572 Dec  9  2015 data_processing_paper_version_SUBMITTED.py
-rw------- 1 megies users   3946 Oct 29  2015 data_processing_WORKS.py
```

“Manual” file/program versioning..

- wastes disk space
- costs hours over hours to figure out what was changed when and why when coming back to the project after some time

Sounds complicated.. Why bother?

Advantages:

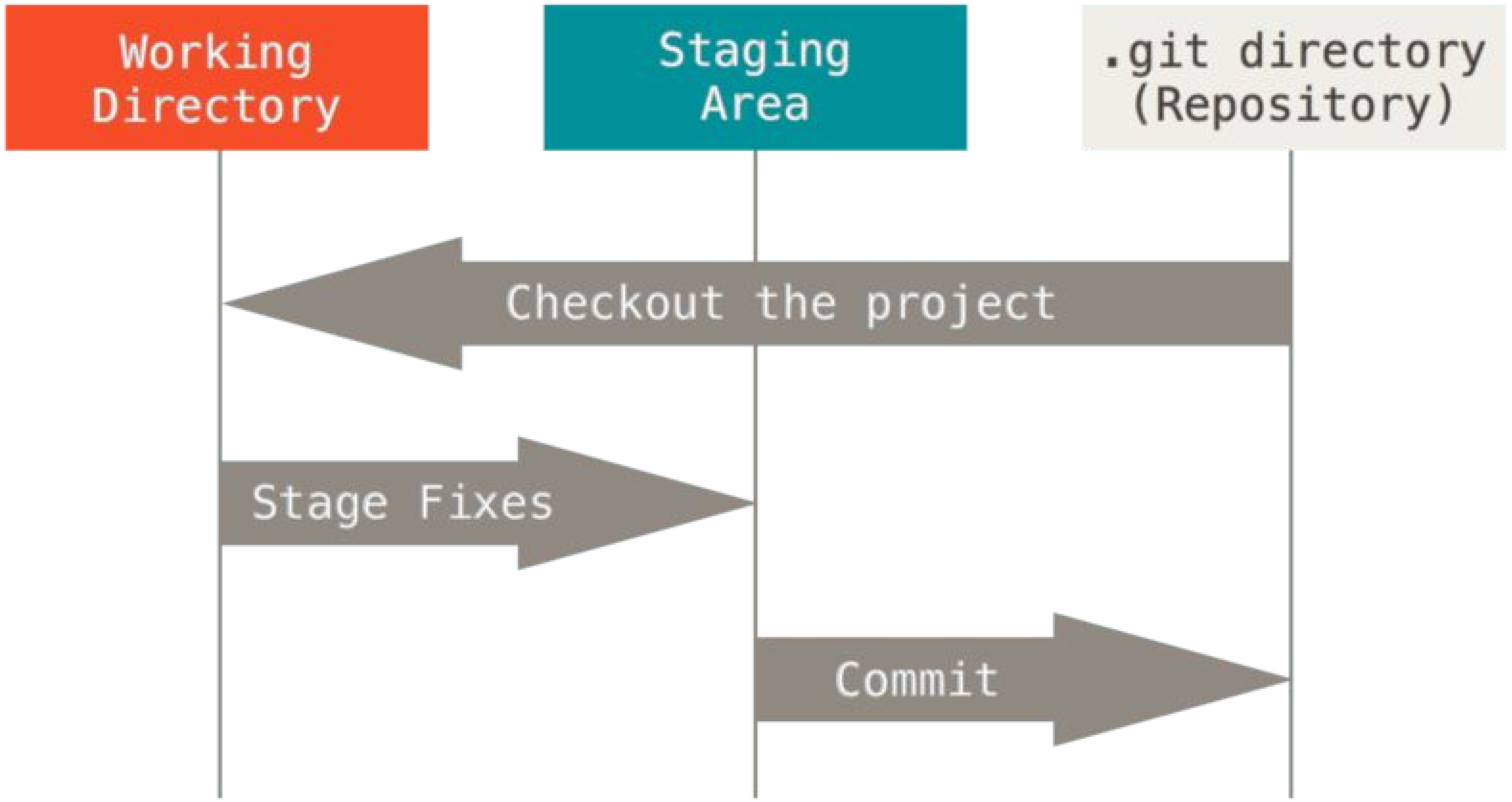
- clean, well organized and documented history
- old states of the project can be easily recovered
- easy synchronization to remotes brings backup solution for free
- multiple people can work on the same project simultaneously, changes can be merged later

git

What is a git repository?

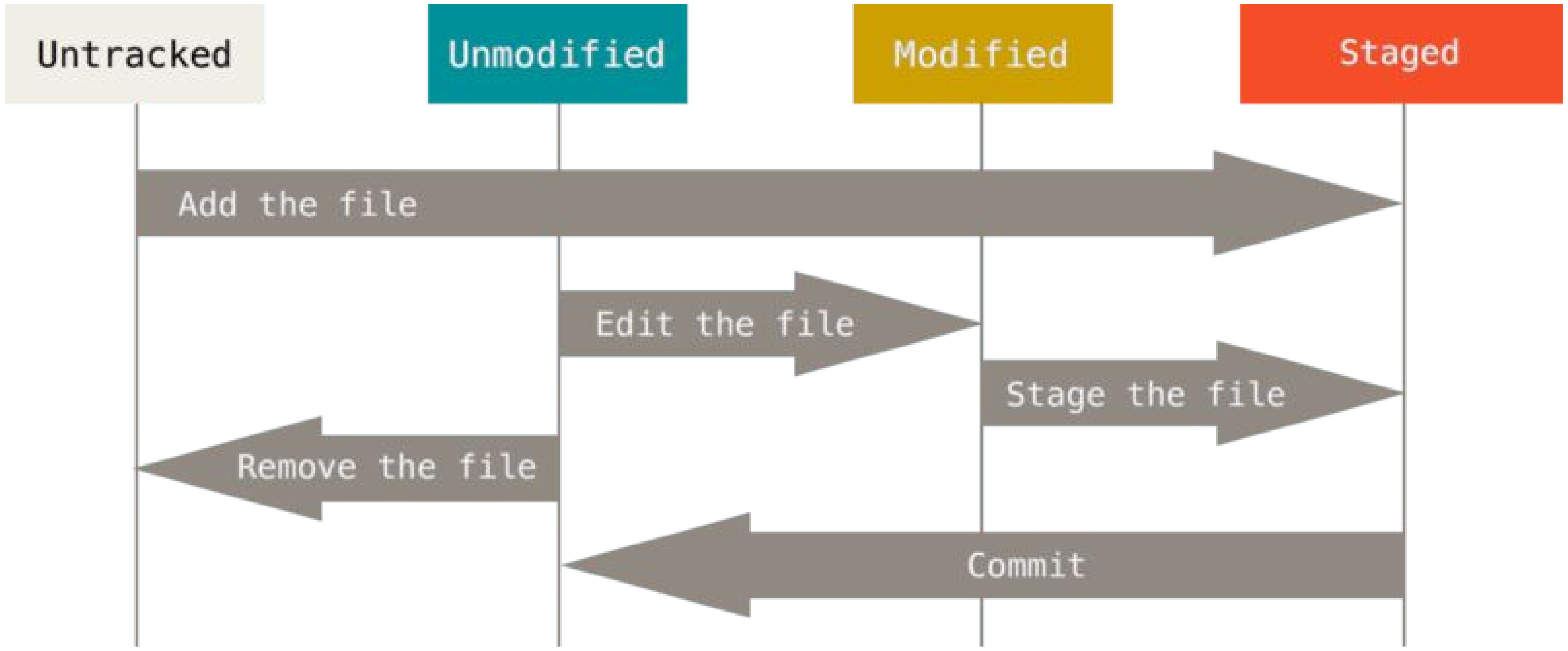
- Local folder on disk
- special directory `.git` in top-level
- Interaction/control over repository via command line interface when somewhere inside the repository
- when switching between revisions/branches/etc, git changes contents of local folder structure automatically
    - “removes” files not in the given revision
    - “adds” files not in the current revision
    - changes files that are in both revisions
    - whenever git can not perform this operation, it stops and tells you to take action

git



from: “Pro Git” online at https://git-scm.com/book/en/v2/ (license CC BY-NC-SA 3.0)

git



from: “Pro Git” online at https://git-scm.com/book/en/v2/ (license CC BY-NC-SA 3.0)

git

Glossary

- commit
  - a set of changes in the repository
  - identified globally by a unique hash
  - stores metadata on the commit: author, time of commit, commit message
- branch
  - a line of development in the project
  - a branch is like a “sign” that points to a commit, that represents the latest state of development in the branch
- remotes
  - related git repositories at different locations (other computer, github, …) that (in parts) share the identical history

git

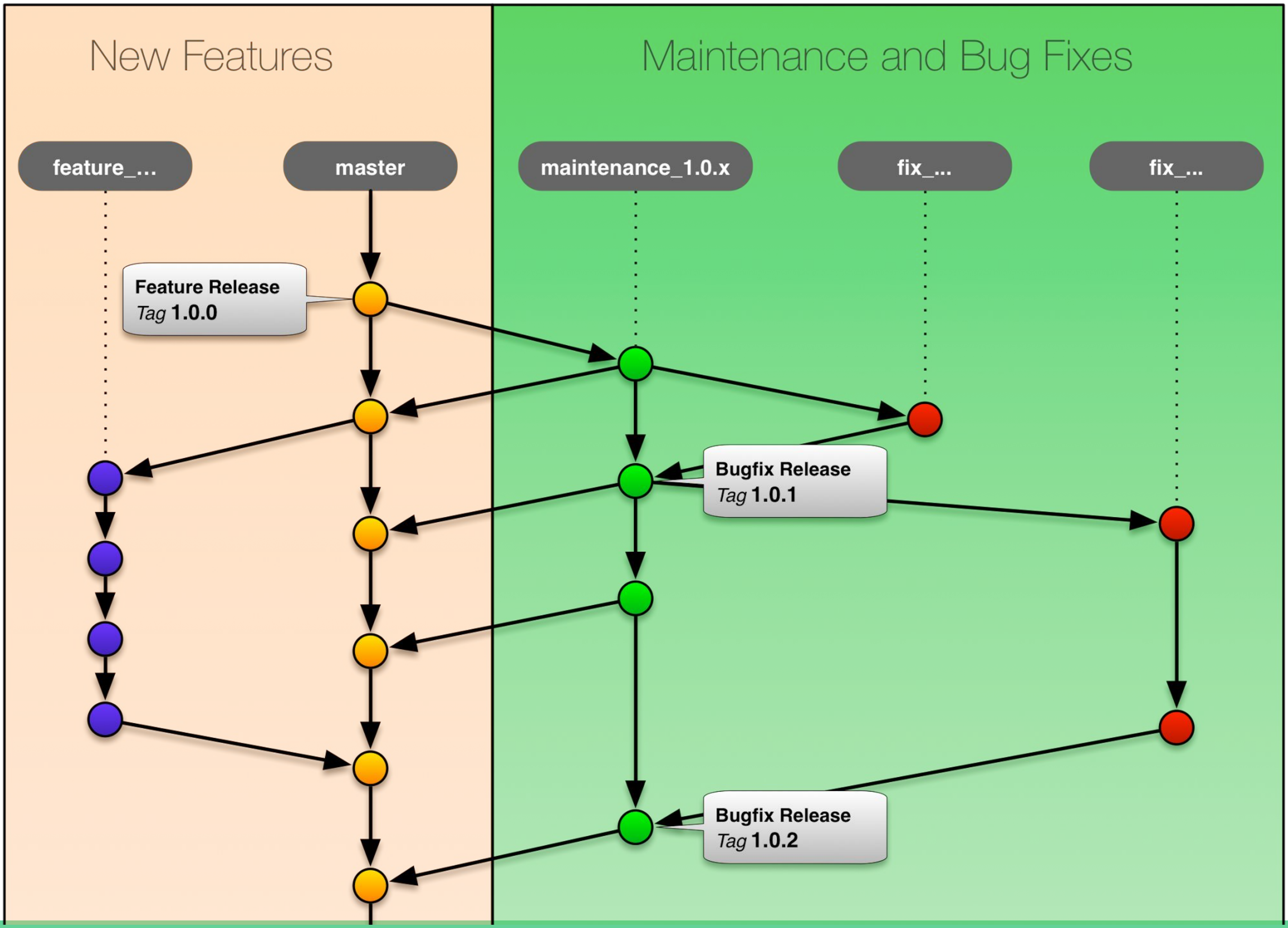
Why branches?

- branches help isolate development on separate topics
- branches help to keep the main line of development in a healthy state while working on new features etc.

Example..

- program that works on earthquake detection in near real time, needs one branch that is always working reliably
- new features might take some time to implement..
- sometimes bugs appear that need to be fixed immediately

from: https://github.com/obspy/obspy/wiki



git

Example of a simple use case: Local repository to keep track of a manuscript

- local repository only (no remotes)
- only work in one branch (“master”)
- commit specific versions, e.g. when handing manuscript to a collaborator/supervisor for comments or when submitting the manuscript for review

git

First steps..

```
$ mkdir my_first_git_repo
$ cd my_first_git_repo
my_first_git_repo$ git init
my_first_git_repo$ vi some_file
my_first_git_repo$ git add some_file
my_first_git_repo$ git commit
```

github

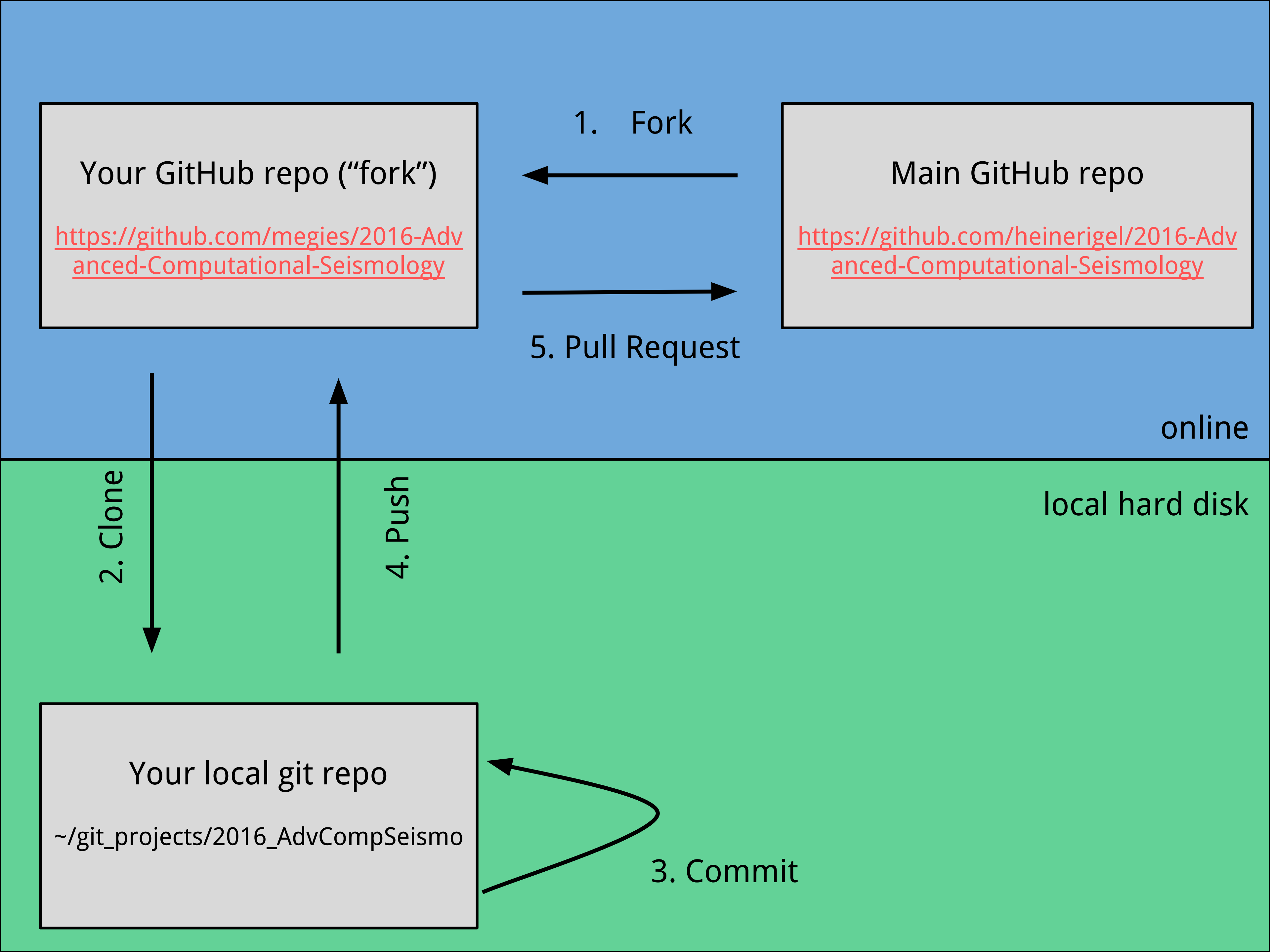
github puts the central server back into git

- github provides web hosting of (open source) git repositories
  - note: free unlimited private repositories for scientific accounts
- a git repository at github is in no way different than a git repository on your hard disk
- … but it adds additional features around it
  - bug/issue tracker
  - proposing changes to another person’s repository (aka “pull request”)
  - wiki
  - …
- having a central copy of the project online for everybody to synchronize with helps when working jointly on a project

github

Glossary

- fork
  - a repository connected to your account that shares (parts of) the history of another person's project
- pull request
  - letting another person know that you want to add some changes to their repository
  - requests that a branch from your fork be merged into a branch on another person's repository
- remote-tracking branch
  - a branch in your local repository on hard disk that ought to be synchronized with a branch in the repository at some other location, e.g. at github
  - if you tell git what branch at a remote should be used to synchronize against, git can tell you the status of synchronization (behind/ahead etc.)
- `$ git pull`
  - update your local branch with new commits that were added to the remote branch
- `$ git push`
  - update the remote branch with new commits you did locally


1. Fork
Your GitHub repo ("fork")
https://github.com/megies/2016-Advanced-Computational-Seismology
Main GitHub repo
https://github.com/heinerigel/2016-Advanced-Computational-Seismology
5. Pull Request
online
local hard disk
2. Clone
4. Push
Your local git repo
~/git_projects/2016_AdvCompSeismo
3. Commit

git -- Useful Commands

Getting information

```
$ git help

$ git status

$ git branch -vv

$ git remote -vv
```

git -- Useful Commands

Committing changes

```
$ git add

$ git add -p

$ git commit
```

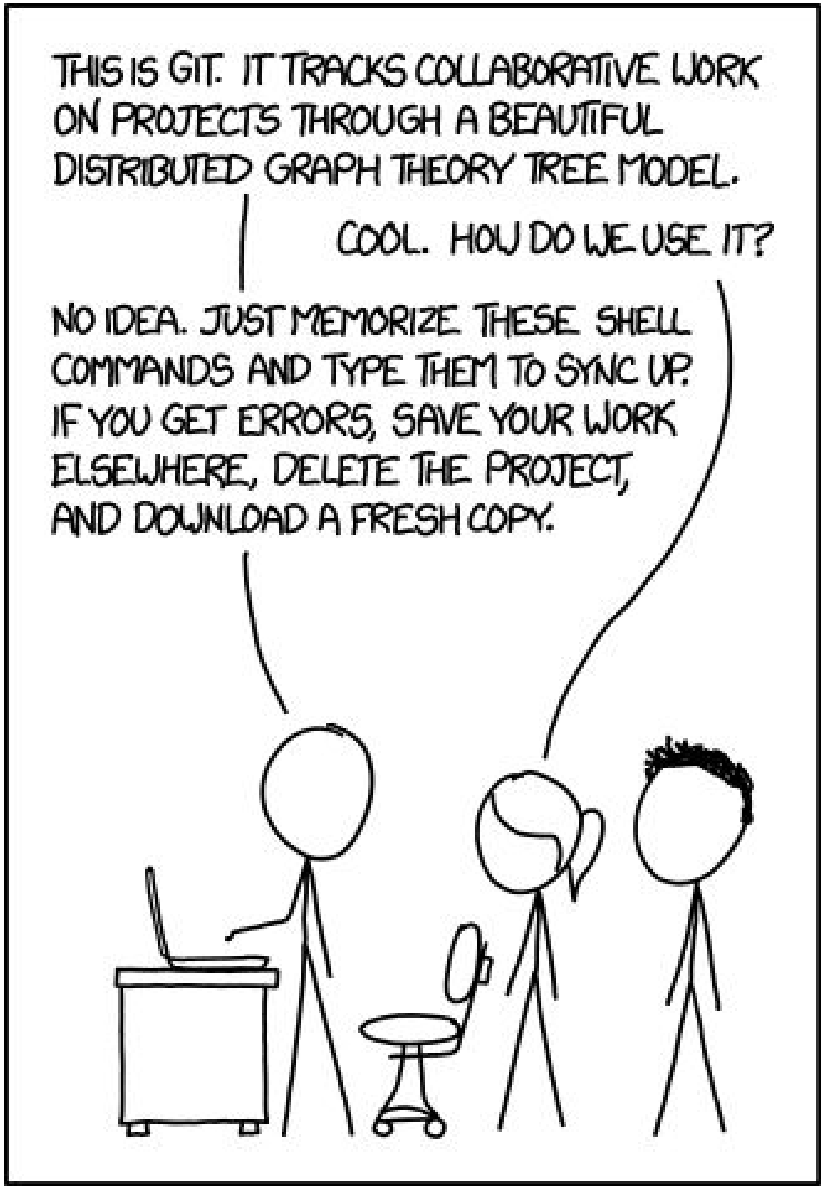
git -- Useful Commands

Synchronizing with remotes

```
$ git push --set-upstream <remote> <branch>

$ git push

$ git stash

$ git pull
```

from: “xkcd” at https://xkcd.com/1597/ (license:CC BY-NC 2.5)

Questions?


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

git(hub)

Example:

- fork seminar repository
- clone to local disk
- add some changes
- push those changes to the fork
- send a pull request