Karlsruhe Institute of Technology

OpenSense

KIT-Campus Alpin
IMK-IFU: Atmospheric Environmental Research

Reproducible science and programming with github

Maximilian Graf[1], Nico Blettner[1,2], Christian Chwala[1,2]

[1] Institute of Meteorology and Climate Research, Karlsruhe Institute of Technology, Campus Alpin, Garmisch-Partenkirchen, Germany
[2] Institute of Geography, University of Augsburg, Augsburg, Germany

Source: Flickr

Goals

- Understand the basics of git, github and mybinder
- Enable contribution to the sandbox environment

Agenda

1. Introduction to git and github (~60 min)
   - Explain concept
   - Learn a git/github workflow (hands-on)
2. Intro to mybinder (~15 min)
3. Short recap of the OpenSense sandbox environment (~10 min)

git

GitHub

What is git - what is github?

git	GitHub
version control software	version control tool
installed locally	hosted on web
command line tool	command line, GUI or web interface
tool to manage different versions of edits made to files in a git repository	a space to upload or copy a git repository
no external tool integration	supports third party tool integration (e.g. mybinder)

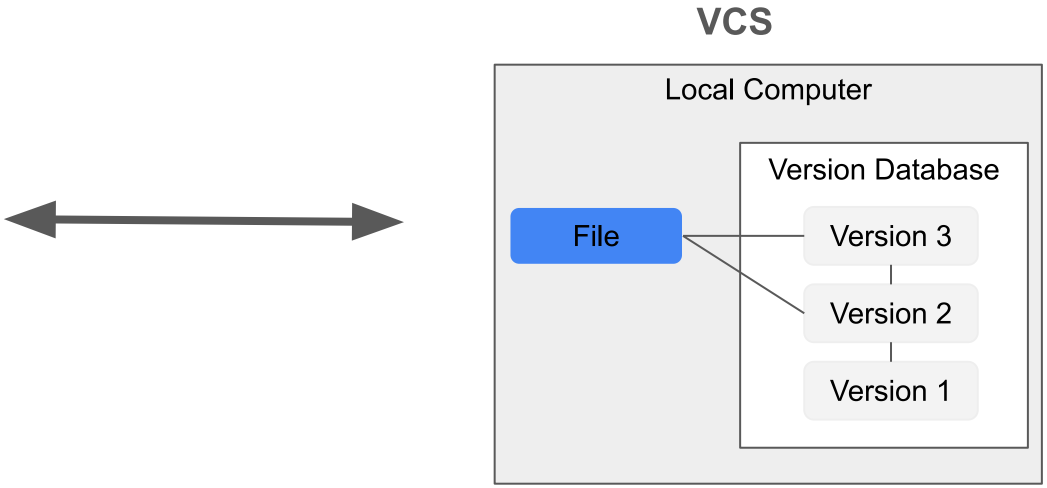
What is a version control system (VCS)?

A version control system (VCS) is a system that tracks changes to a file or set of files over time.

Single file, single person case

DIY versioning

- code_final.py
- code_final_new.py
- code_final_new2.py
- …



What is a version control system (VCS)?

A version control system (VCS) is a system that tracks changes to a file or set of files over time.

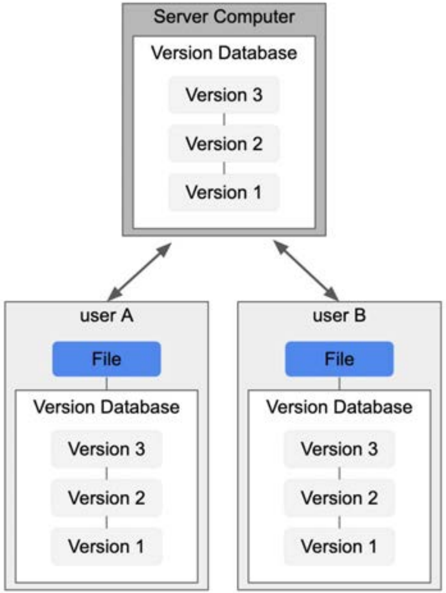
multi file, multi person case

DIY versioning

code_final.py	Author A
code_final_new.py	Author B
code_final_new2.py	Author A
readme.md	Author B
readme2.md	Author A
requirements.yml	Author C
…	

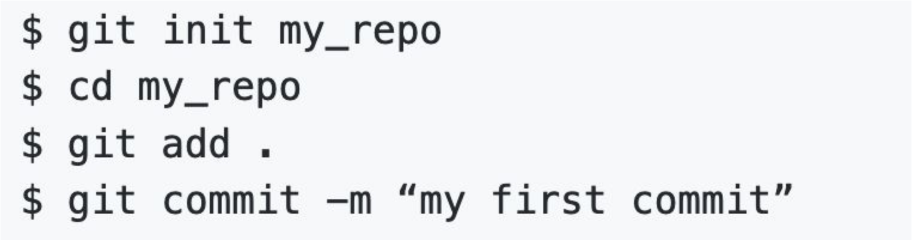
Who did what, when and why?

VCS

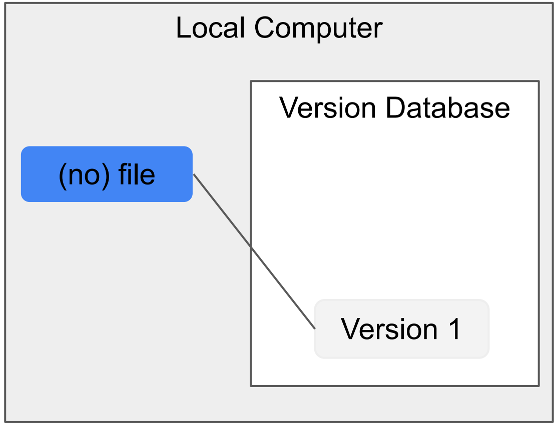


Local git workflow

Single file, single person case

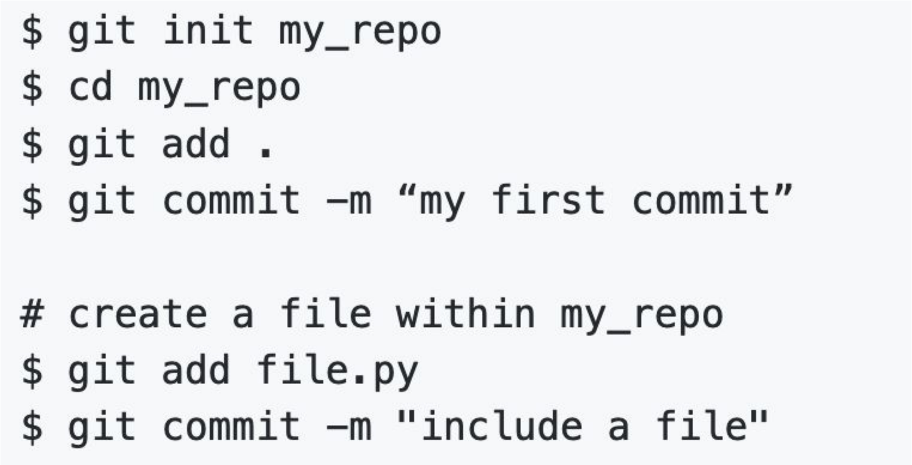


```
$ git init my_repo
$ cd my_repo
$ git add .
$ git commit -m “my first commit”
```



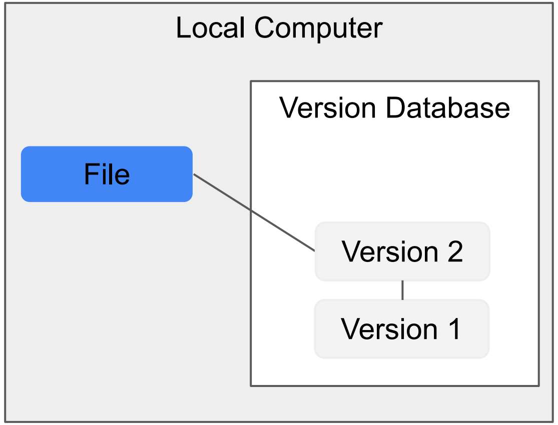
Local git workflow

Single file, single person case



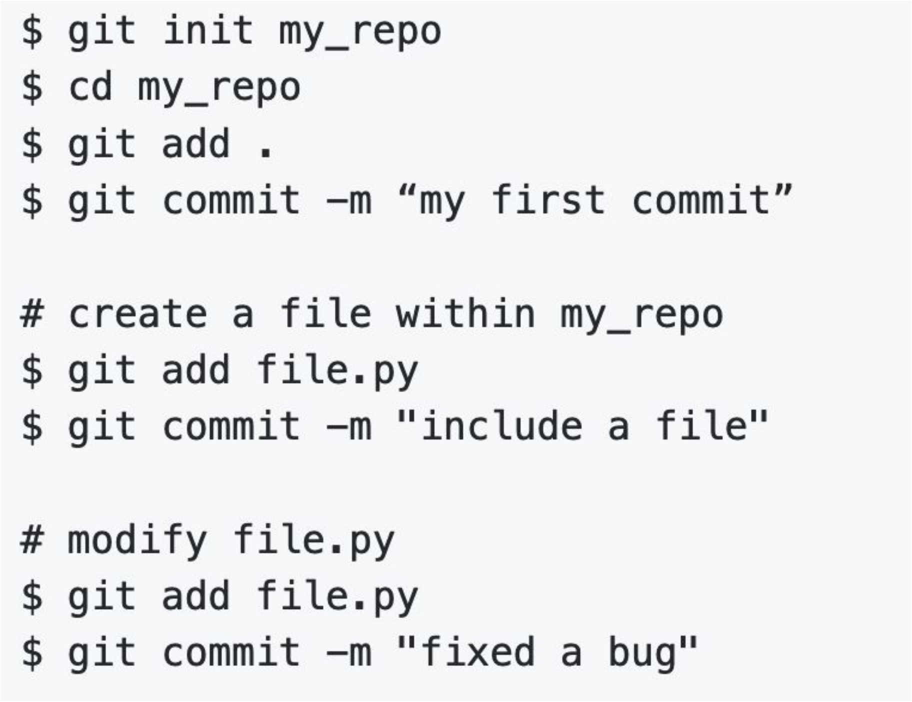
```
$ git init my_repo
$ cd my_repo
$ git add .
$ git commit -m “my first commit”

# create a file within my_repo
$ git add file.py
$ git commit -m "include a file"
```



Local git workflow

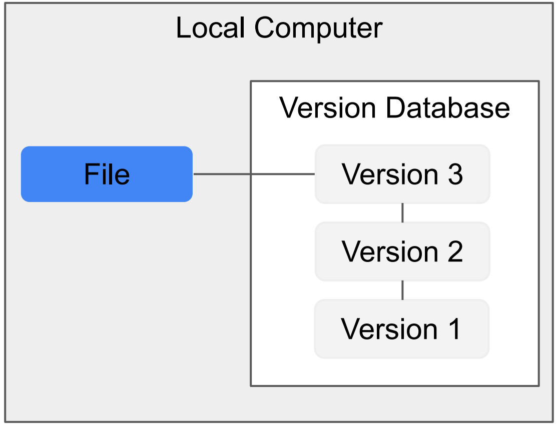
Single file, single person case



```
$ git init my_repo
$ cd my_repo
$ git add .
$ git commit -m “my first commit”

# create a file within my_repo
$ git add file.py
$ git commit -m "include a file"

# modify file.py
$ git add file.py
$ git commit -m "fixed a bug"
```



Local Computer

Version Database

File

Version 3

Version 2

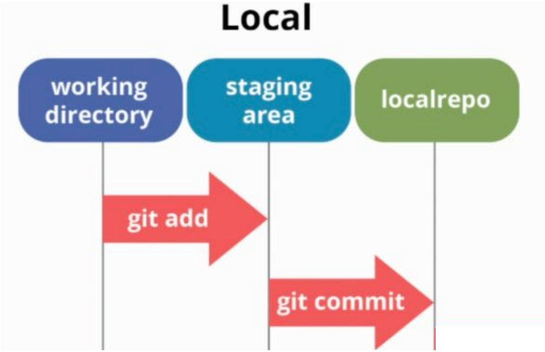
Version 1

“fixed a bug”

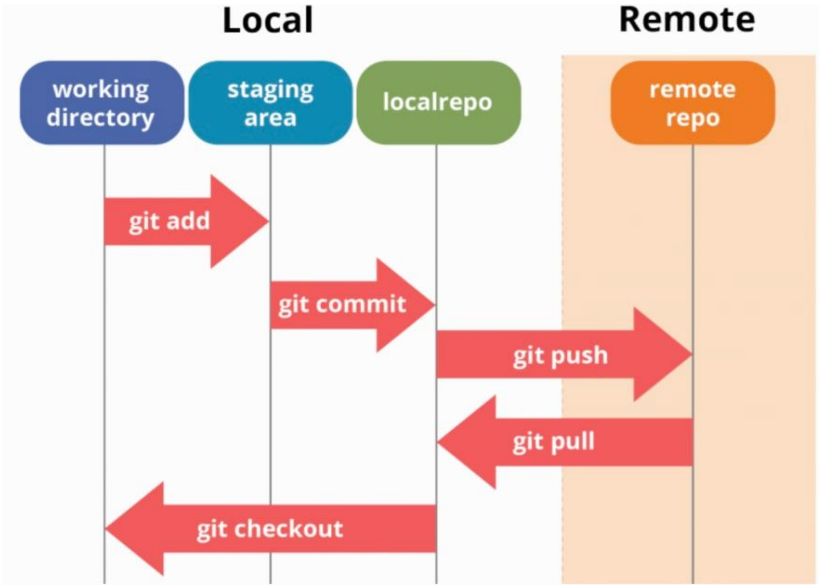
“include a file”

“my first commit”

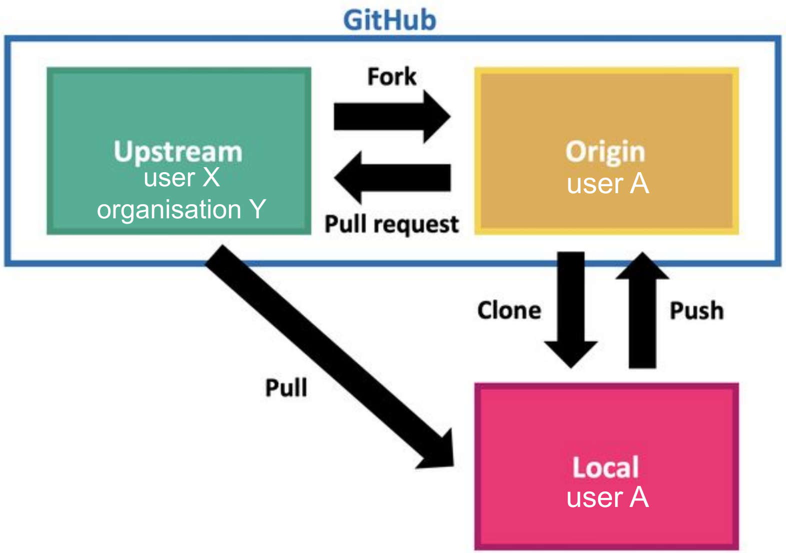
Local git workflow - another illustration

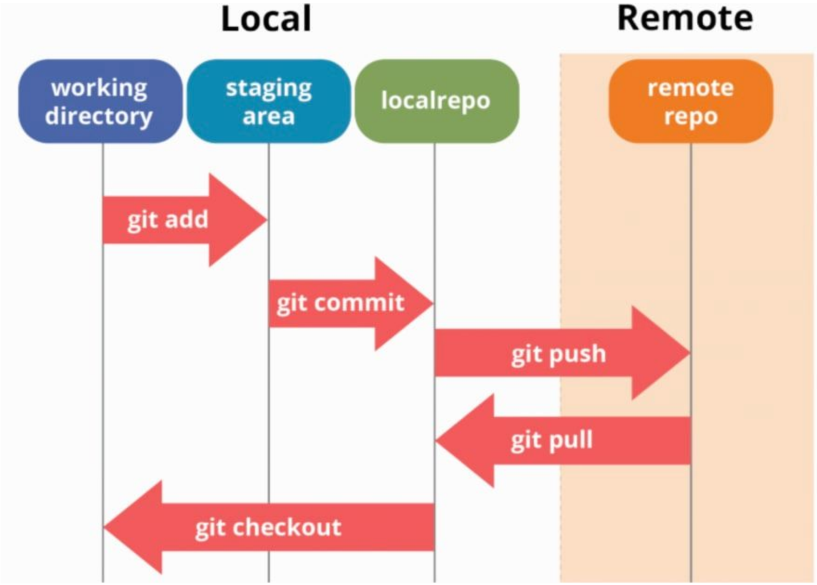


Adding github as remote server

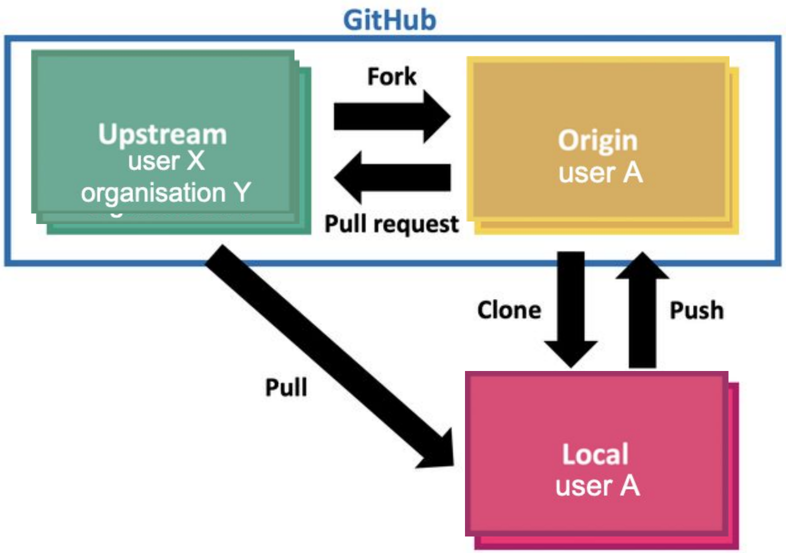


Adding github as remote server



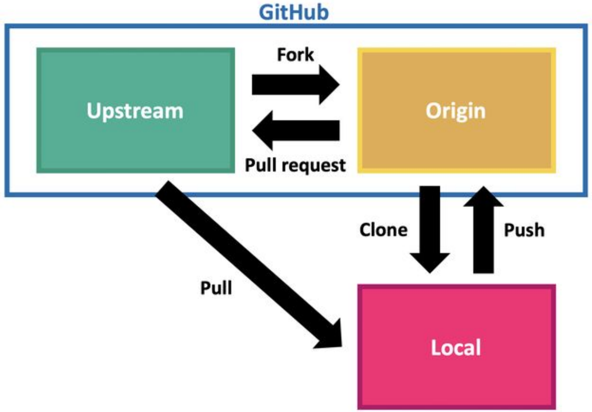


Concept of branches



- A branch is a new/separate version of the main repository
- The main branch is called “main” (earlier “master”) by default
- New branches get names typically describing the feature they should add
- Branches can be used to develop while keeping the stable version untouched

Workflow for today



Additional useful commands:

- Inspect status of files `git status`
- Show branches `git branch -a`

Get ready

1. Fork a GitHub repository: navigate to a repository on GitHub and click the Fork button.
2. Clone the forked repository locally: `git clone https://github.com/repository`.
3. Add remote called “upstream” pointing to the original repository: `git remote add upstream https://github.com/repository`.

Make changes and contribute to a repository

4. Checkout a new branch (here called “new_feature”): `git checkout -b new_feature`
5. Make desired changes to the local repository on this branch.
6. Track changes with git (use `git add / git remove` and `git commit`).
7. Push changes to your remote repository: `git push origin new_feature`.
8. Open a pull request on GitHub merging your changes with the upstream (original) repository.

Update local and origin

9. Once the pull request is accepted, you’ll want to pull those changes into your origin (forked repository). Change to master: `git checkout main` and pull: `git pull upstream main`. Then push to your origin `git push origin main`
10. Delete your feature branch using the GitHub website or, delete the local branch: `git branch -d new_feature`, and delete the remote: `git push origin --delete new_feature`.

Hands-on session

Exercise 1

- fork and clone following repository: https://github.com/OpenSenseAction/OpenSense_workshop_git_hub.git
- create a new branch called feature_1
- edit the readme.md file and remove remove_me.py
- add, commit and push the modified file to origin/feature_1 (your fork)

Exercise 2

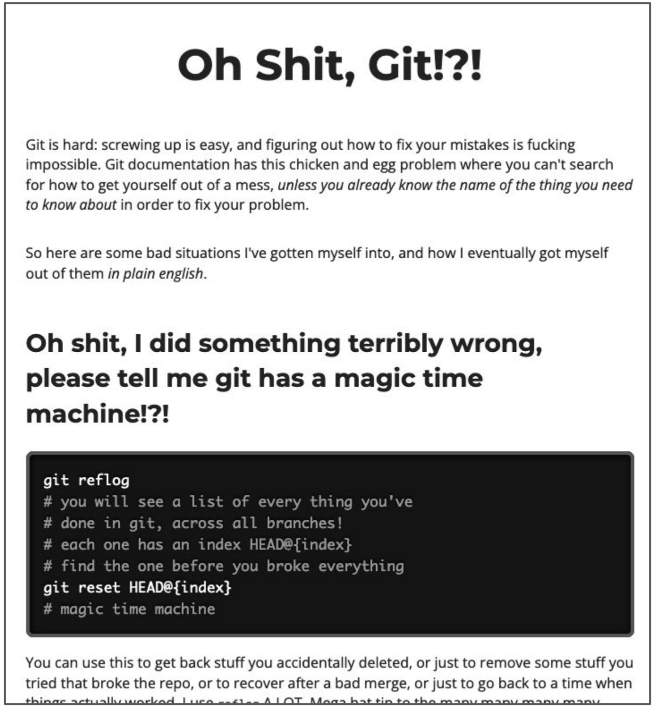
- locally create a new branch (checkout from your main branch!)
- in “data dir” copy the template.csv file and rename the copy with an individual name
- fill the table with some data
- push your commits to origin (add, commit, push)
- create a pull request

Exercise 3 (optional)

- add a new function to functions.py and open a pull request

Some resources

ohshitgit.com



Oh Shit, Git!?!

Git is hard: screwing up is easy, and figuring out how to fix your mistakes is fucking impossible. Git documentation has this chicken and egg problem where you can't search for how to get yourself out of a mess, *unless you already know the name of the thing you need to know about* in order to fix your problem.

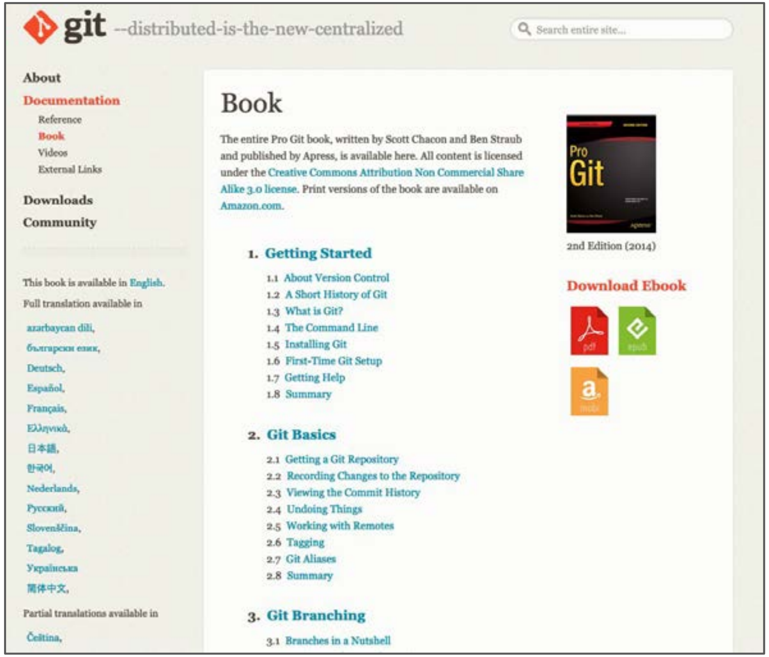
So here are some bad situations I've gotten myself into, and how I eventually got myself out of them *in plain english*.

Oh shit, I did something terribly wrong, please tell me git has a magic time machine!?!

```
git reflog
# you will see a list of every thing you've
# done in git, across all branches!
# each one has an index HEAD@{index}
# find the one before you broke everything
git reset HEAD@{index}
# magic time machine
```

You can use this to get back stuff you accidentally deleted, or just to remove some stuff you tried that broke the repo, or to recover after a bad merge, or just to go back to a time when things actually worked. I use reflog A LOT. Mega hat tip to the many many many many

git-scm.com/book/



About

Documentation

Reference

Book

Videos

External Links

Downloads

Community

This book is available in English.

Full translation available in

azarbaycan dili,
български език,
Deutsch,
Español,
Français,
Ελληνικά,
日本語,
한국어,
Nederlands,
Русский,
Slovenščina,
Tagalog,
Українська
简体中文,

Partial translations available in

Čeština,

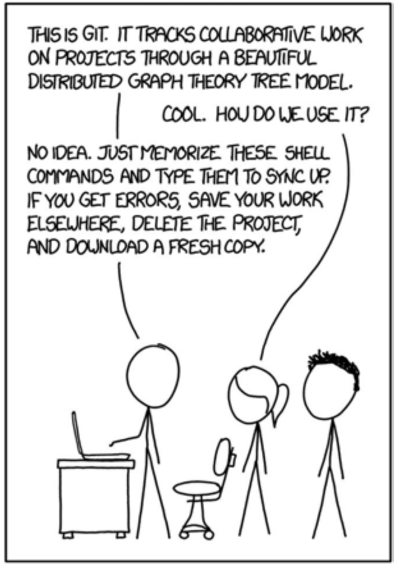
Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the Creative Commons Attribution Non Commercial Share Alike 3.0 license. Print versions of the book are available on Amazon.com.

2nd Edition (2014)

Download Ebook

1. Getting Started
   - 1.1 About Version Control
   - 1.2 A Short History of Git
   - 1.3 What is Git?
   - 1.4 The Command Line
   - 1.5 Installing Git
   - 1.6 First-Time Git Setup
   - 1.7 Getting Help
   - 1.8 Summary
2. Git Basics
   - 2.1 Getting a Git Repository
   - 2.2 Recording Changes to the Repository
   - 2.3 Viewing the Commit History
   - 2.4 Undoing Things
   - 2.5 Working with Remotes
   - 2.6 Tagging
   - 2.7 Git Aliases
   - 2.8 Summary
3. Git Branching
   - 3.1 Branches in a Nutshell



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.

https://xkcd.com/1597/

The binder project

Software project to package and share interactive, reproducible environments

Software stack:

- BinderHub: open-source tool that deploys the Binder service in the cloud
- repo2docker: generates reproducible Docker images from a git repository

Prerequisite in a repository:

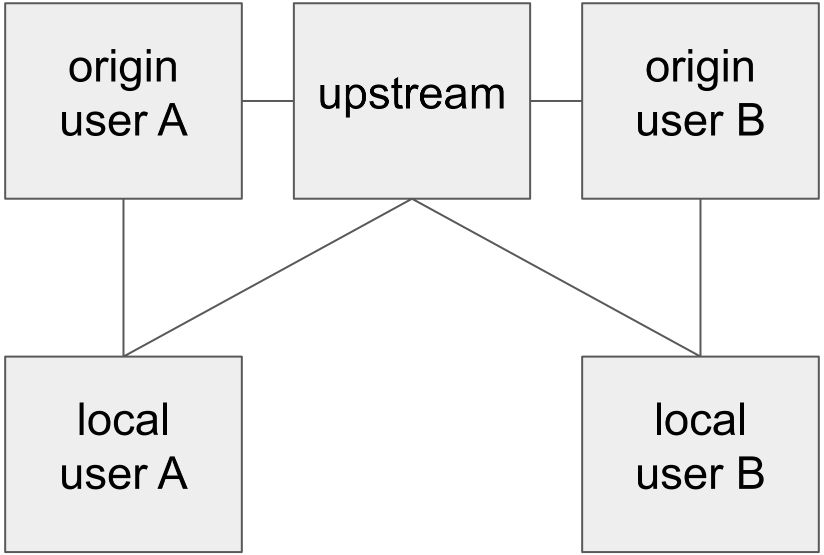
- code
- environment.yml file

Reproducible science and programming with github

Maximilian Graf[1], Nico Blettner[1,2], Christian Chwala[1,2]

[1] *Institute of Meteorology and Climate Research, Karlsruhe Institute of Technology, Campus Alpin, Garmisch-Partenkirchen, Germany*
[2] *Institute of Geography, University of Augsburg, Augsburg, Germany*

Source: Flickr



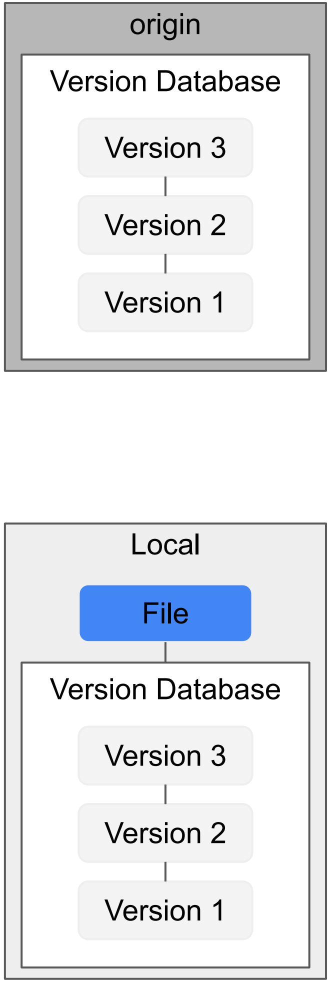
origin user A

upstream

origin user B

local user A

local user B



origin

Version Database

Version 3

Version 2

Version 1

Local

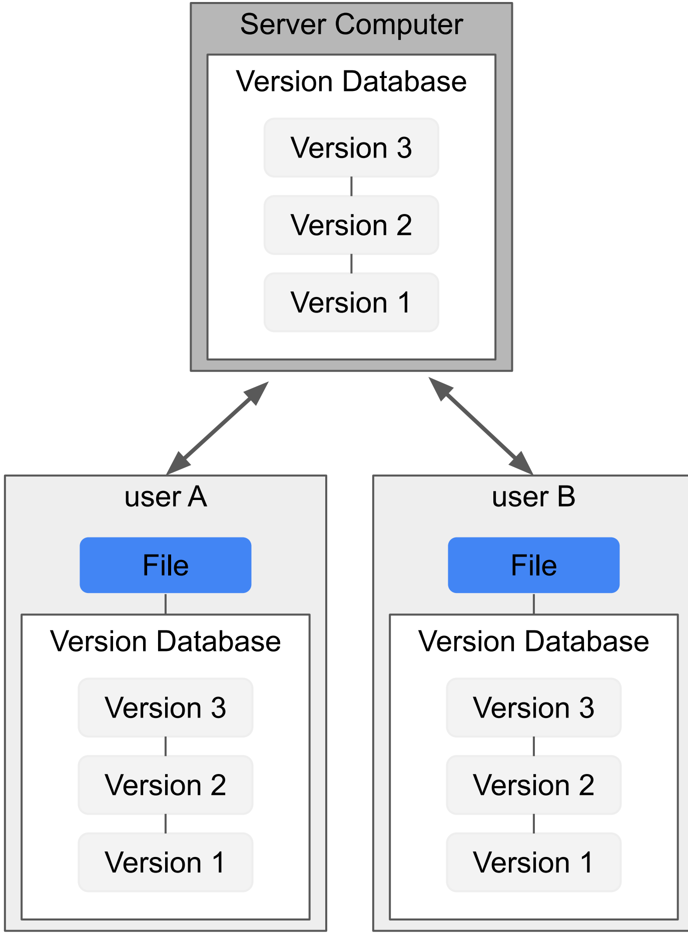
File

Version Database

Version 3

Version 2

Version 1



Server Computer

Version Database

Version 3

Version 2

Version 1

user A

File

Version Database

Version 3

Version 2

Version 1

user B

File

Version Database

Version 3

Version 2

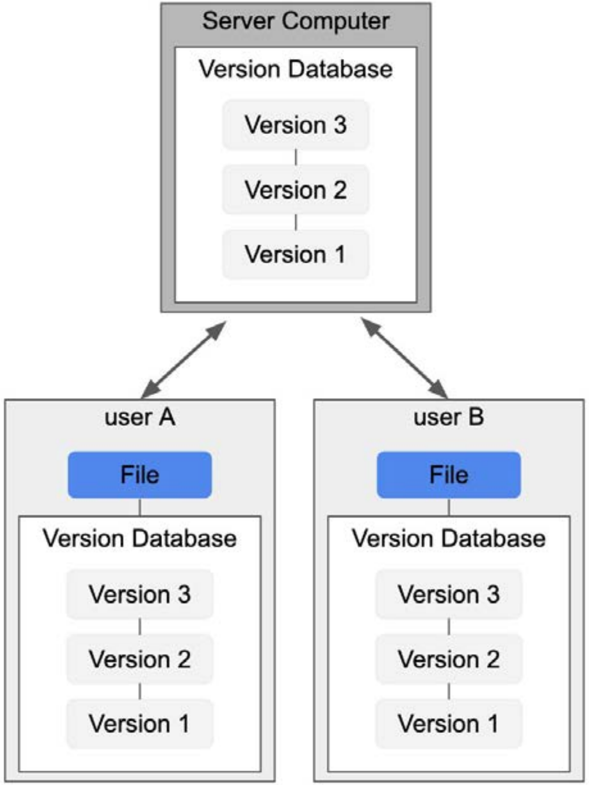
Version 1

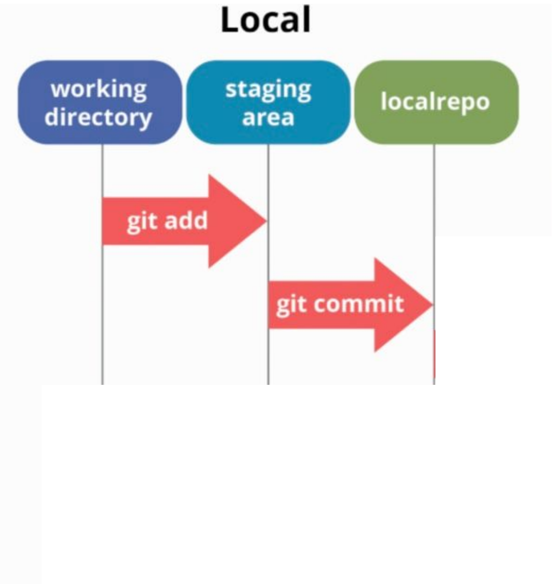
Prerequisites

You will need:

- github account (https://github.com/)
- installation of git
    - if not already available on Mac/Linux: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git;
    - on Windows:https://phoenixnap.com/kb/how-to-install-git-windows go through the steps (you can leave all option on default until “Launch git GUI” and then jump to the final required step “Configure GitHub Credentials”
- ssh key for github (https://docs.github.com/en/authentication/connecting-to-github-with-ssh/about-ssh)

Adding github as remote server


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


Local
working directory
staging area
localrepo
git add
git commit