2024 TdP

Git & GitHub

Quick introduction

Fulvio Corno
Giuseppe Averta
Carlo Masone
Francesca Pistilli

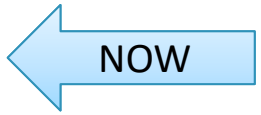
Goal

- What is Revision Control?
- What is Git?
- What is GitHub?
- How to access Revision Control with Git and GitHub from within Eclipse?
- What are the Eclipse workflows useful in this course?

Version Control Systems

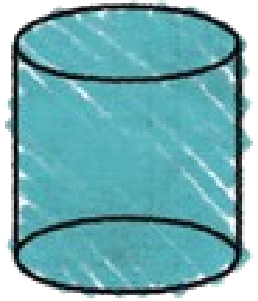
Record changes to a file or a set of files over time so that you can recall specific versions later

- Three generations:
  - Local (RCS, SCCS)
  - Centralized (CVS, Subversion, Team Foundation Server)
  - Distributed (Git, Mercurial)



Basic Concepts

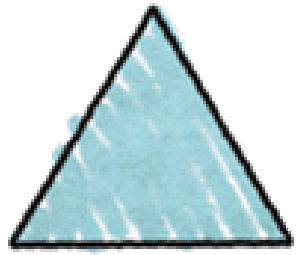
- **Repository**
  - place where you store all your work
  - contains every version of your work that has ever existed
    - files
    - directories layout
    - history
  - can be shared with the whole team



Basic Concepts

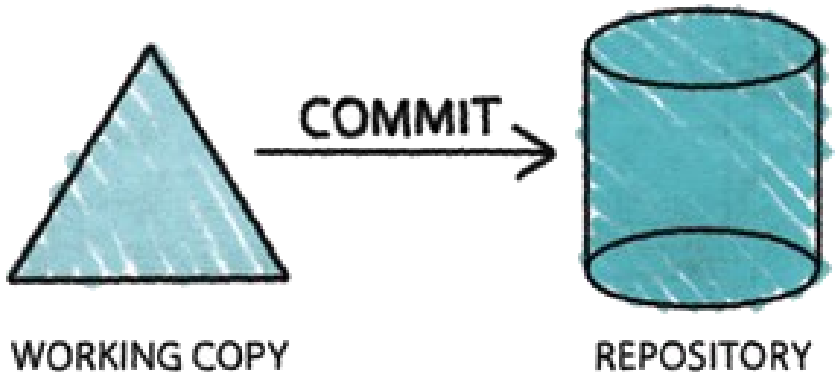
Working copy

- a snapshot of the repository used for… working
- the place where changes happens
- private, not shared with the team
- it also contains some metadata so that it can keep track of the state of things
  - has a file been modified?
  - is this file new?
  - has a file been deleted?



WORKING COPY

Basic Concepts



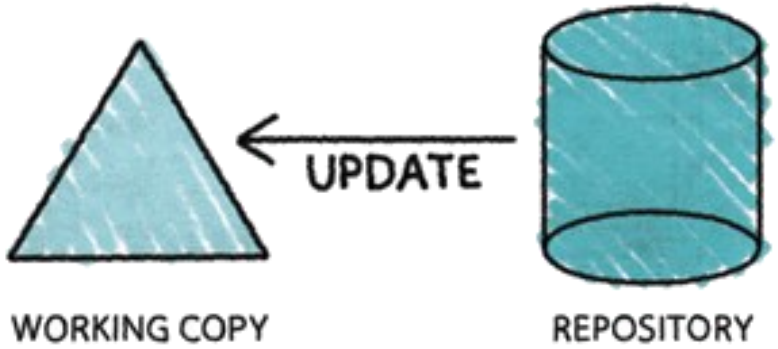
Commit

- the operation that modifies the repository
- atomically performed by modern version control tools
  - the integrity of the repository is ensured
- it is typical to provide a log message (or comment) when you commit
  - to explain the changes you have made
  - the message becomes part of the history of the repository

Basic Concepts

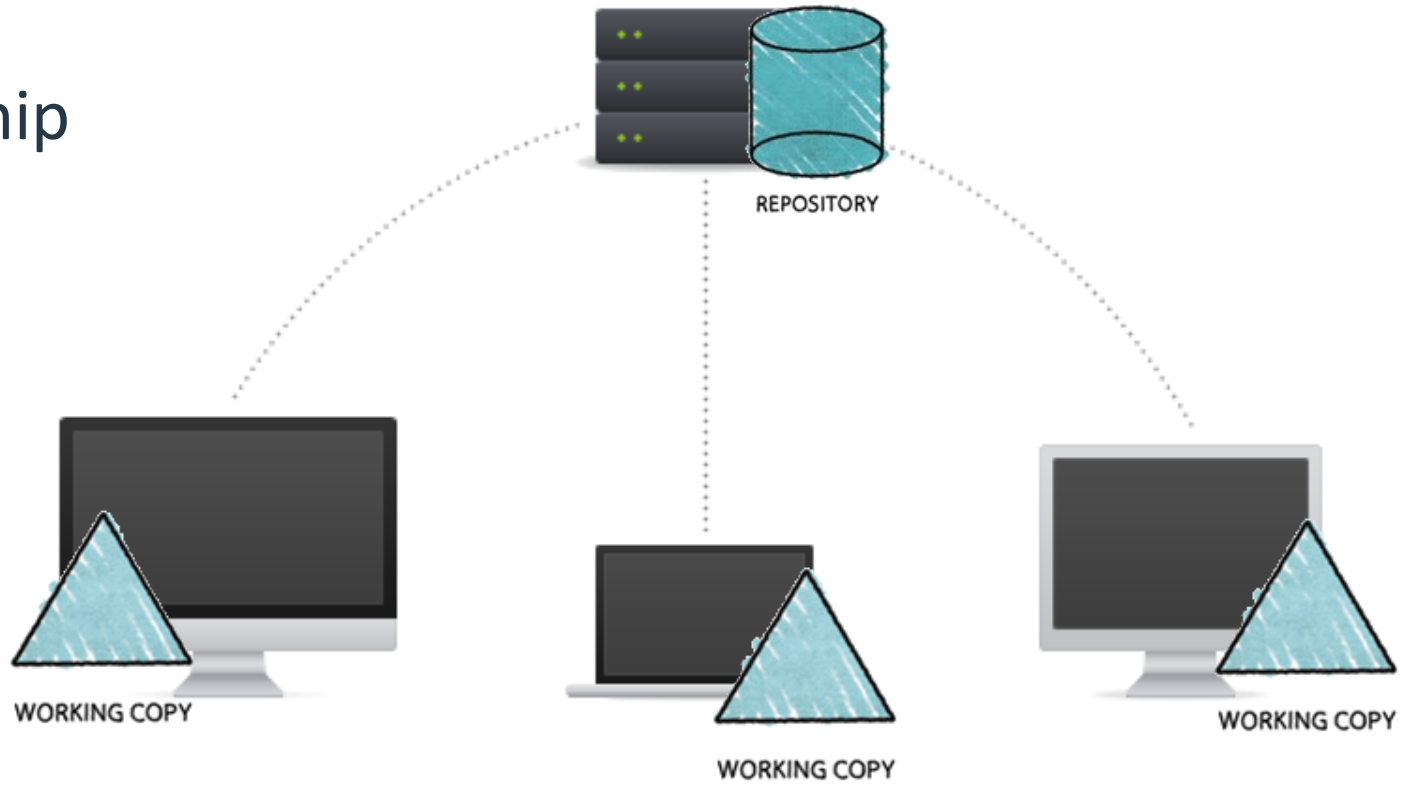
Update

- update the working copy with respect to the repository
  - apply changes from the repository
  - merge such changes with the ones you have made to your working copy, if necessary



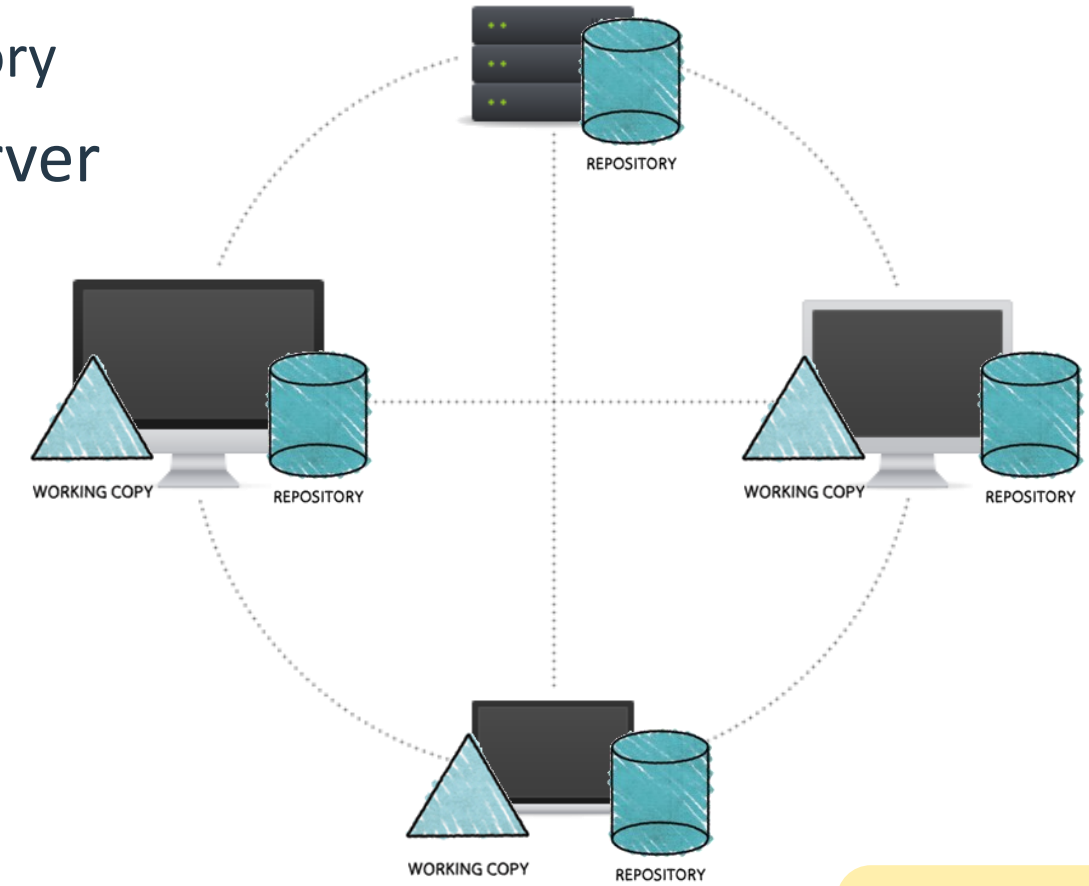
Centralized Version Control

- one central repository
- client-server relationship



Distributed Version Control

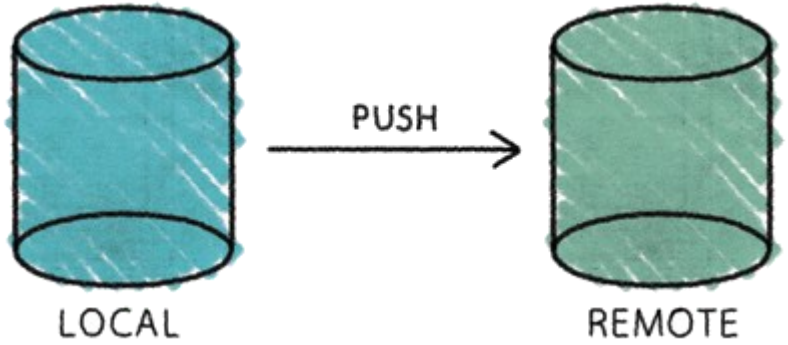
- clients and server have the full copy of the repository
  - local repositories ‘clone’ a remote repository
- it is possible to have more than one server



More Basic Concepts

Push

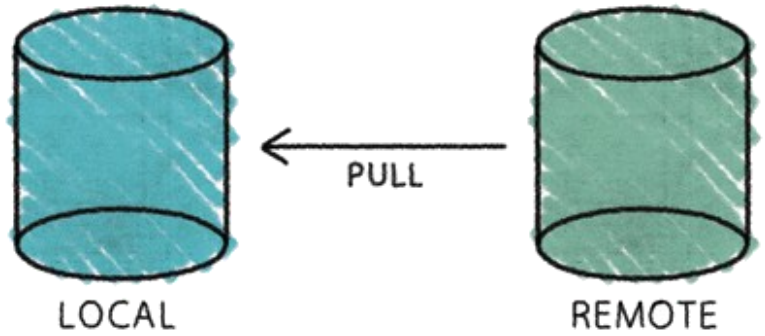
- copy changesets from a local repository instance to a remote one
  - synchronization between two repository instances



More Basic Concepts

Pull

- copy changesets from a remote repository instance to a local one
  - synchronization between two repository instances



Introducing… Git

- Distributed Version Control System
- Born
  - on 2005 for the Linux kernel project
  - to be used via command line
- Website: http://git-scm.com
- Highlights:
  - free and open source
  - strong support for non-linear development
  - fully distributed
  - efficient handling of large projects
  - cryptographic authentication of history

Getting started with Git

- Standard installations
  - http://git-scm.com/downloads
- Available for all the platform
- Git Graphical Applications
  - http://git-scm.com/downloads/guis
  - Suggestion: GitExtensions, SourceTree, Fork
- For this course, Git is
  - integrated in PyCharm

Hosted Git

- To have (at least) one remote repository
  - alternative: set up your own Git server!
- Most popular:
  - **GitHub, https://github.com/**
  - Bitbucket, https://bitbucket.org/
  - GitLab, https://about.gitlab.com/gitlab-com/

GitHub

- Slightly different than other code-hosting sites
  - instead of being primarily based on the project, it is user-centric
  - social coding
- Owned by Microsoft
  - free account to host as many open source project as you want
  - free plans for students
    - https://education.github.com

For Labs

- Create a personal GitHub account
  - You will have “education” discounts if you use your University e-mail
  - https://education.github.com
- Try Git!
  - http://try.github.io/
  - 15 minutes tutorial

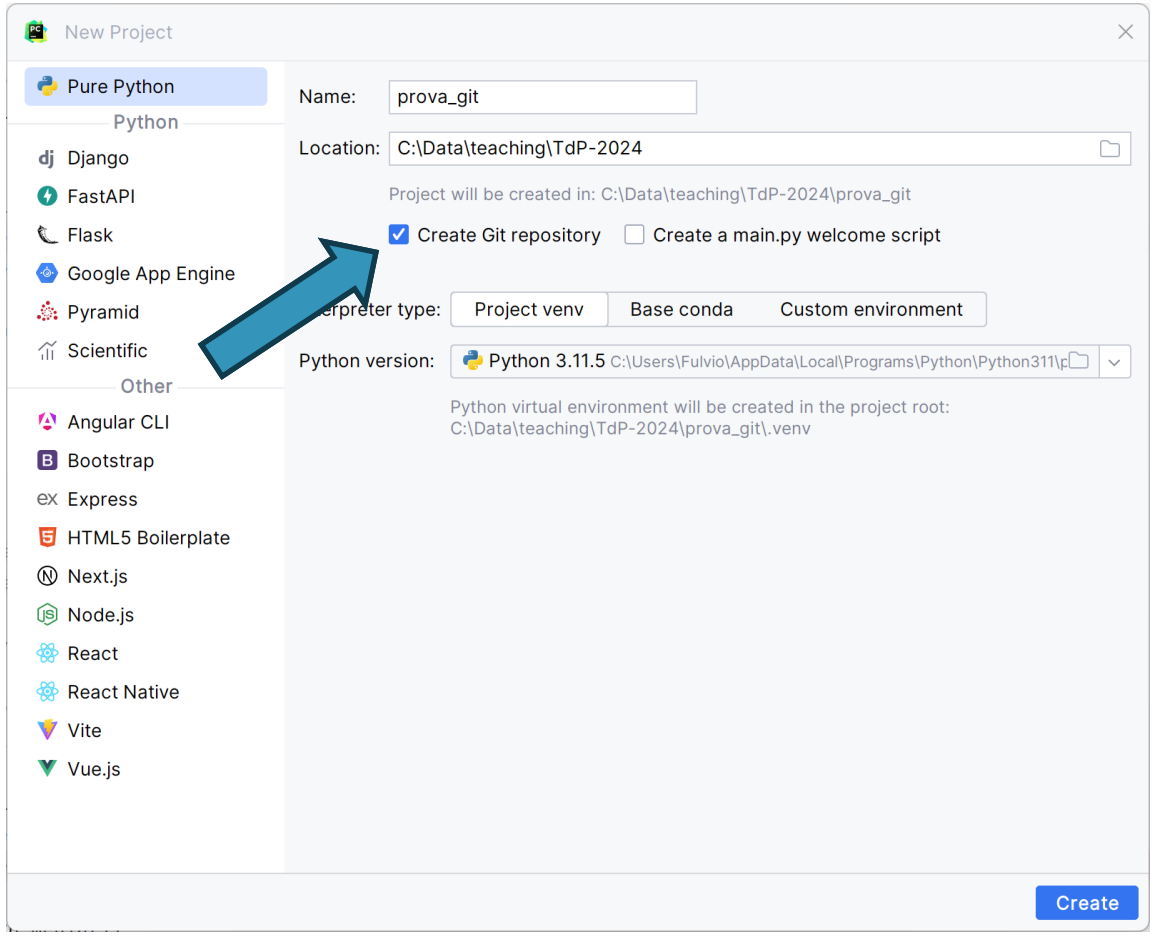
Quick introduction to Git & GitHub

GITHUB-BASED WORKFLOWS

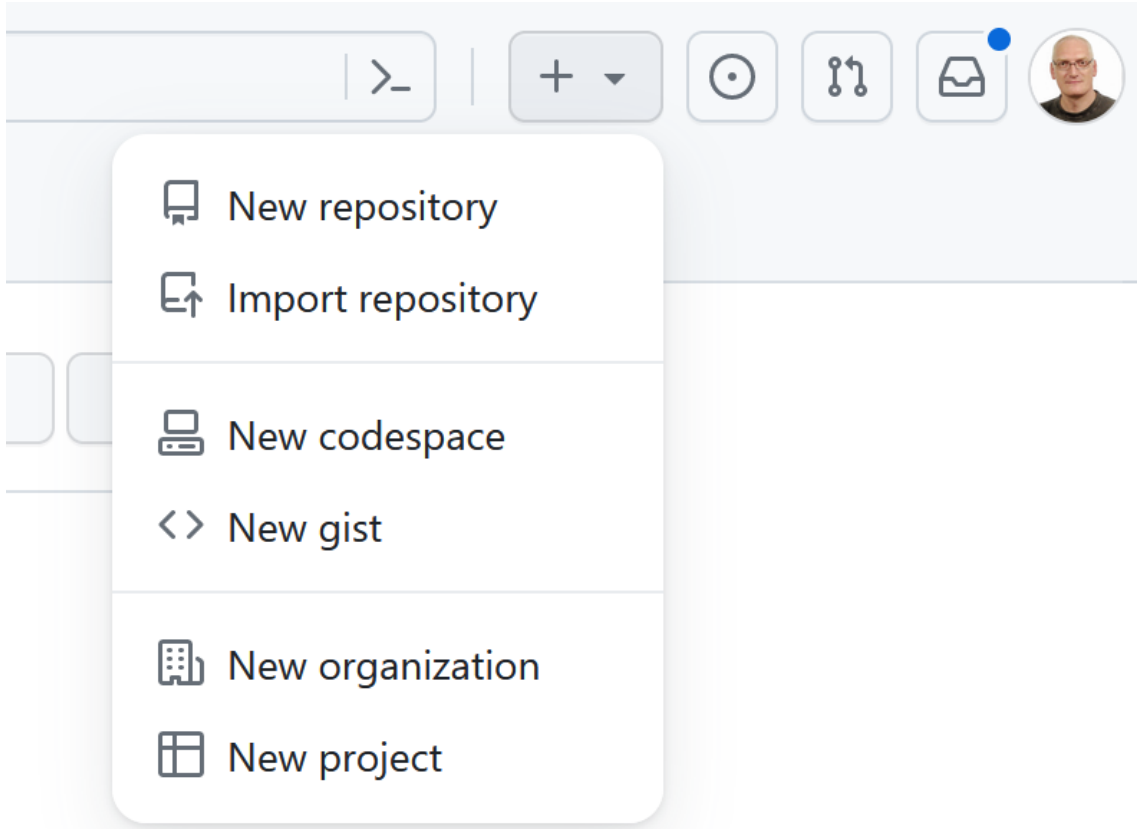
Workflow 1: “Create new project”

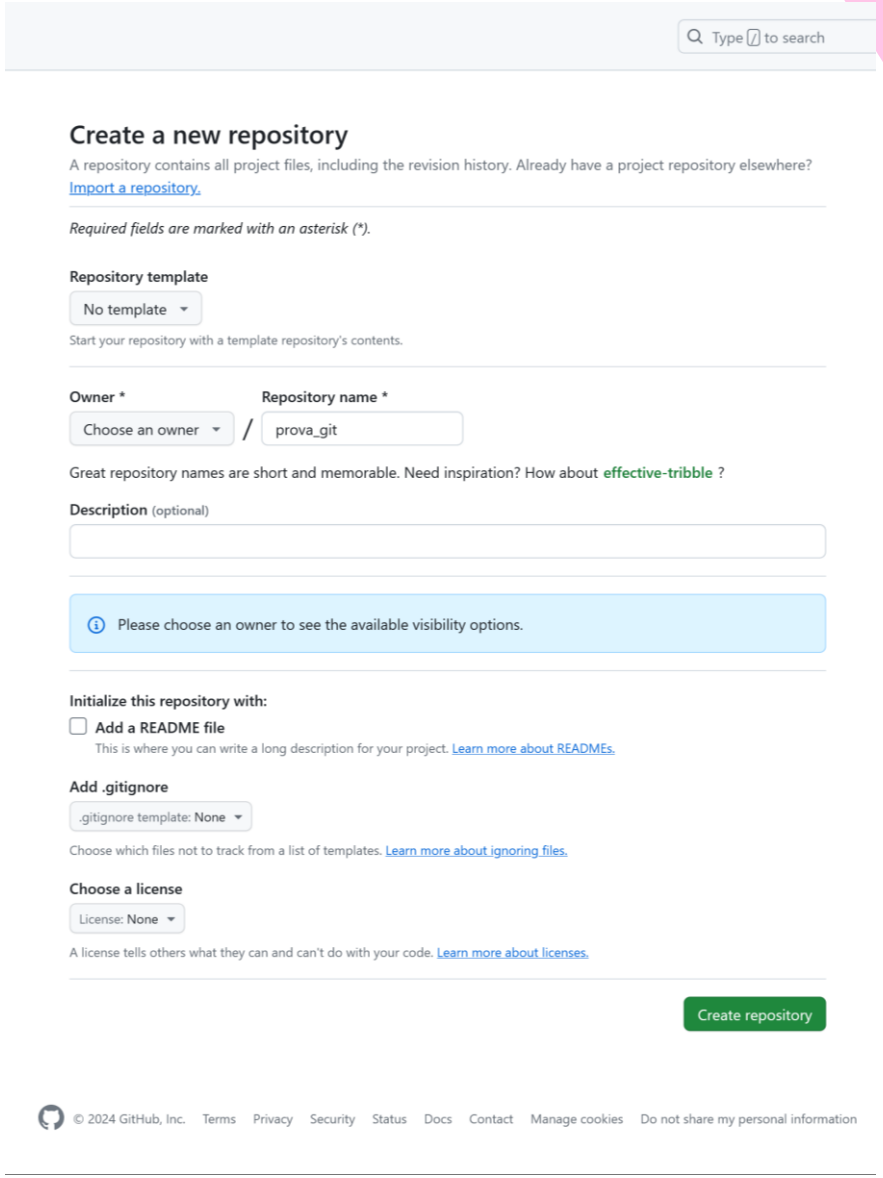
1. Create a project in PyCharm
    1. File/New Project…
    2. Select “Create Git Repository”
2. Create and edit Python files
3. Create a new (empty) project in GitHub
    1. Copy the Project URL
4. Push changes (**Commit&push**)
    1. The first time, you must Define Remote

Create new project in PyCharm

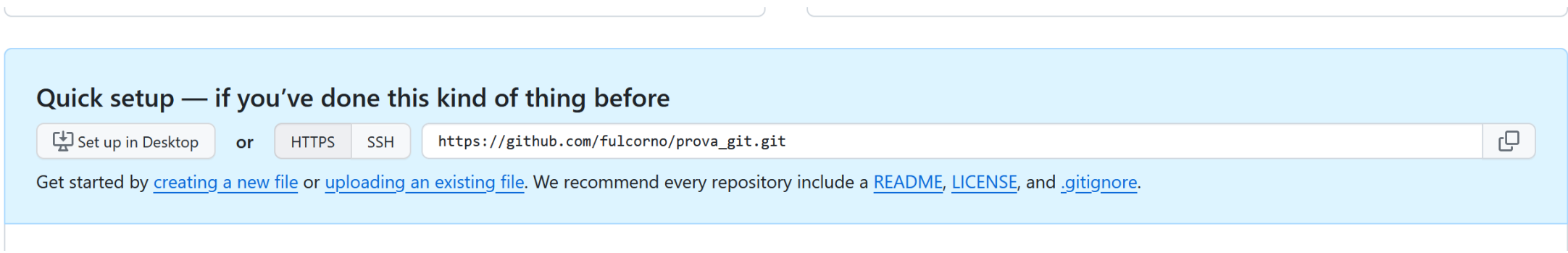


Create Repository in GitHub

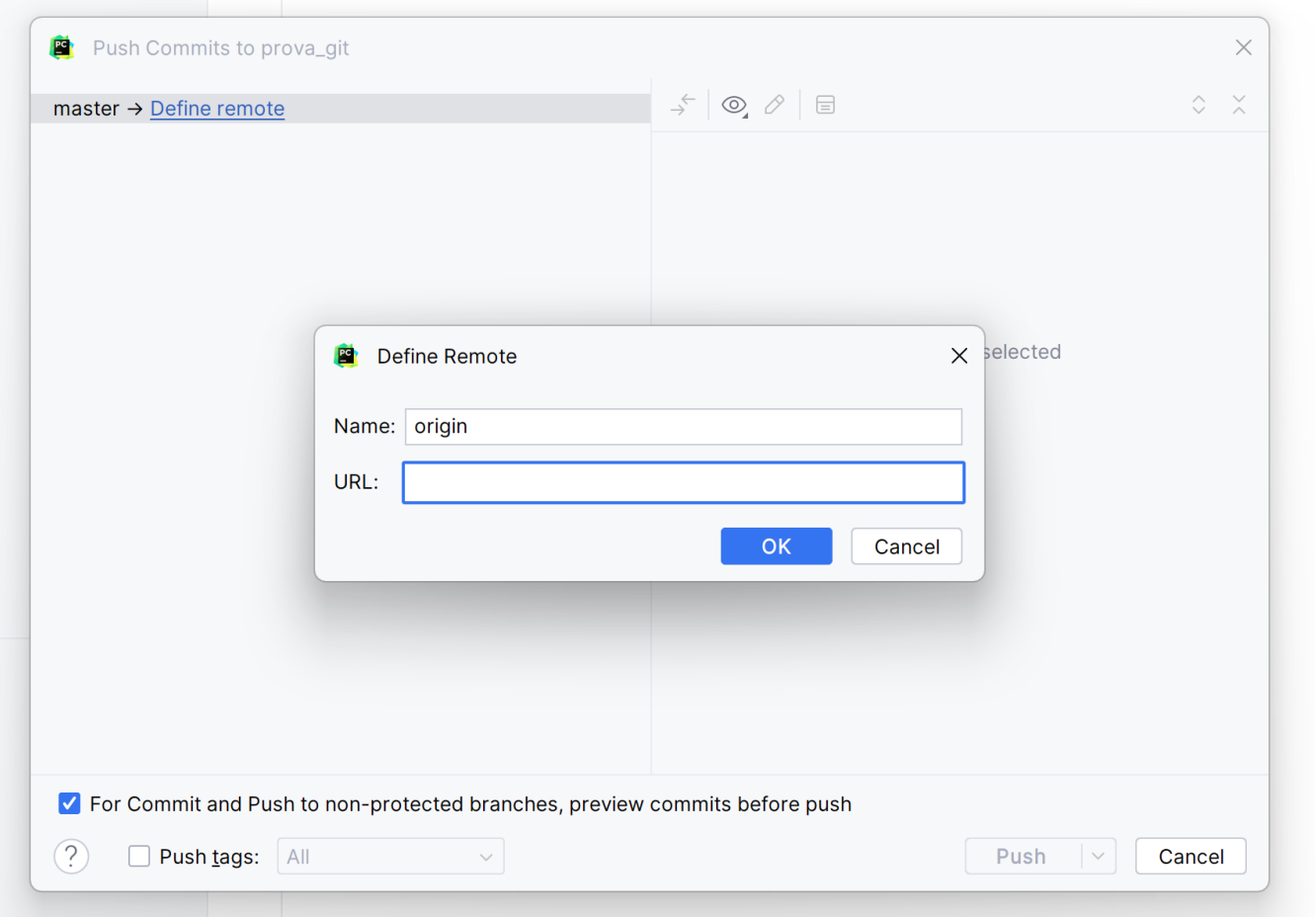




Copy the Repository URL



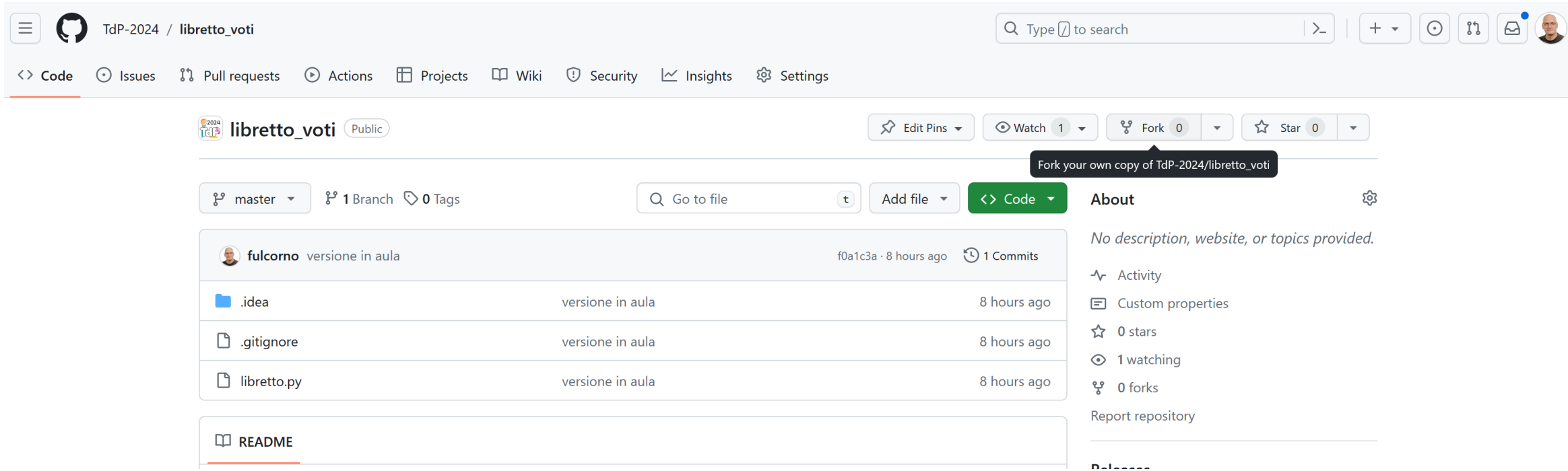
Defining remote to push



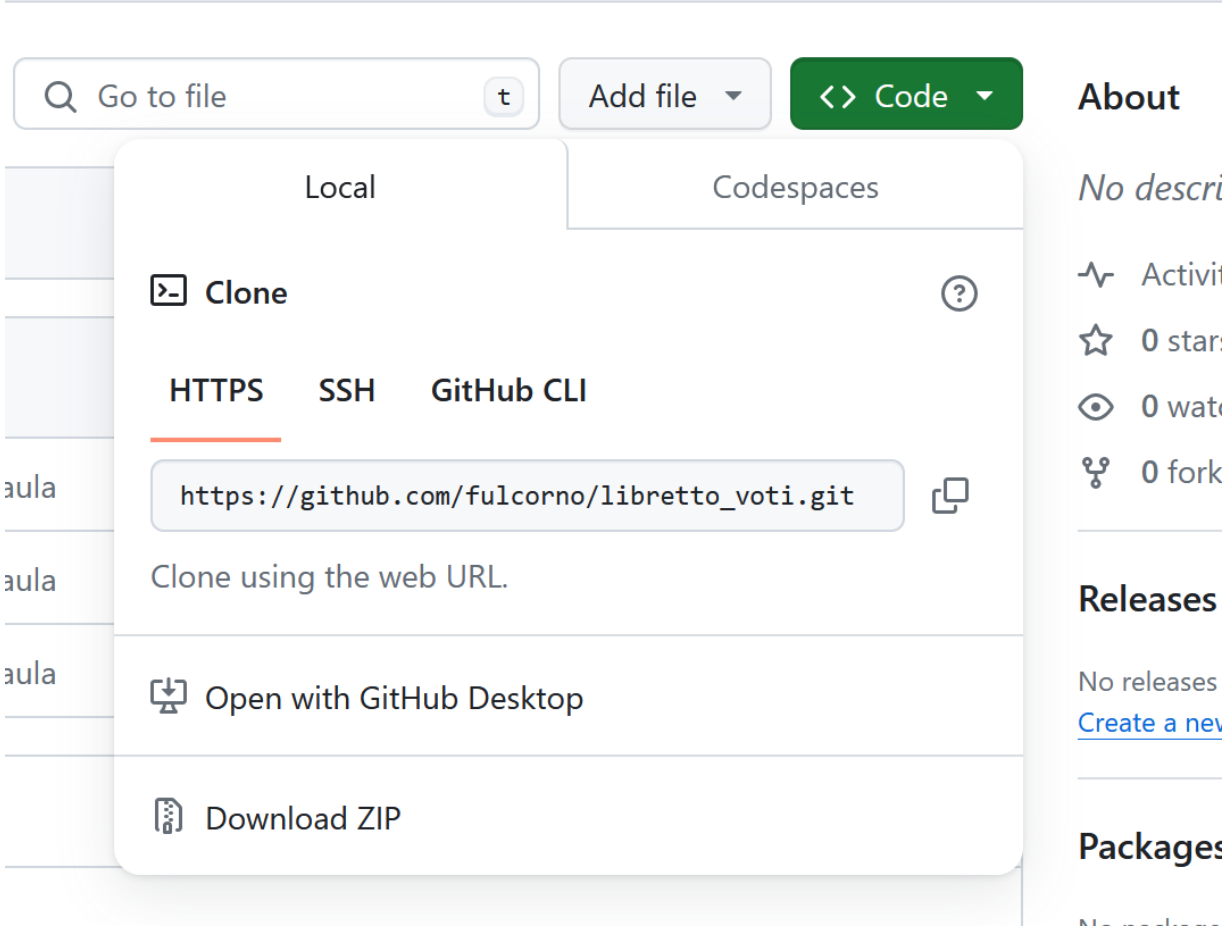
Workflow 2: “Work on a project”

1. “**Fork**” the project in GitHub (you make a copy in your repository)
2. **Clone** your project in PyCharm
3. Work on the project
4. **Commit** and **Push** the changes

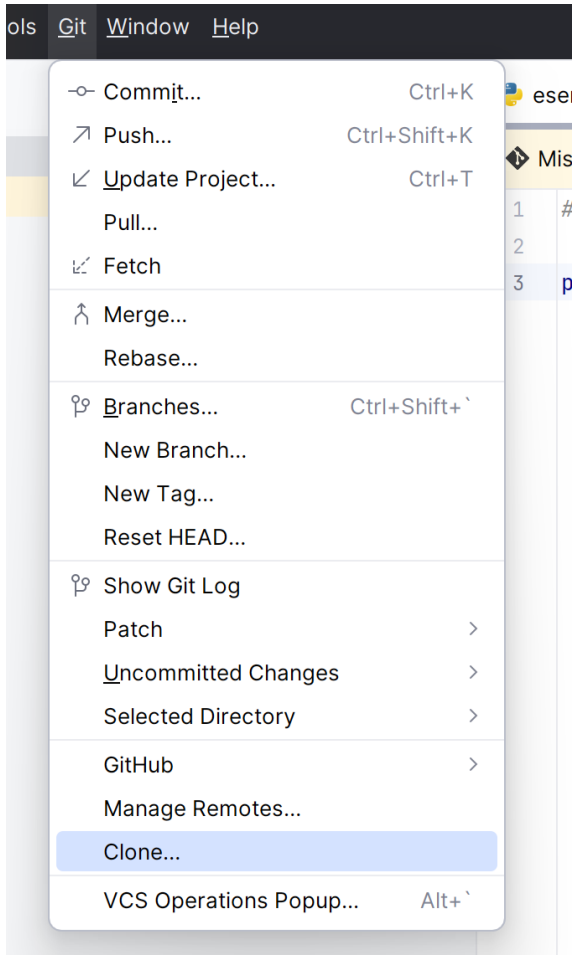
Forking the project

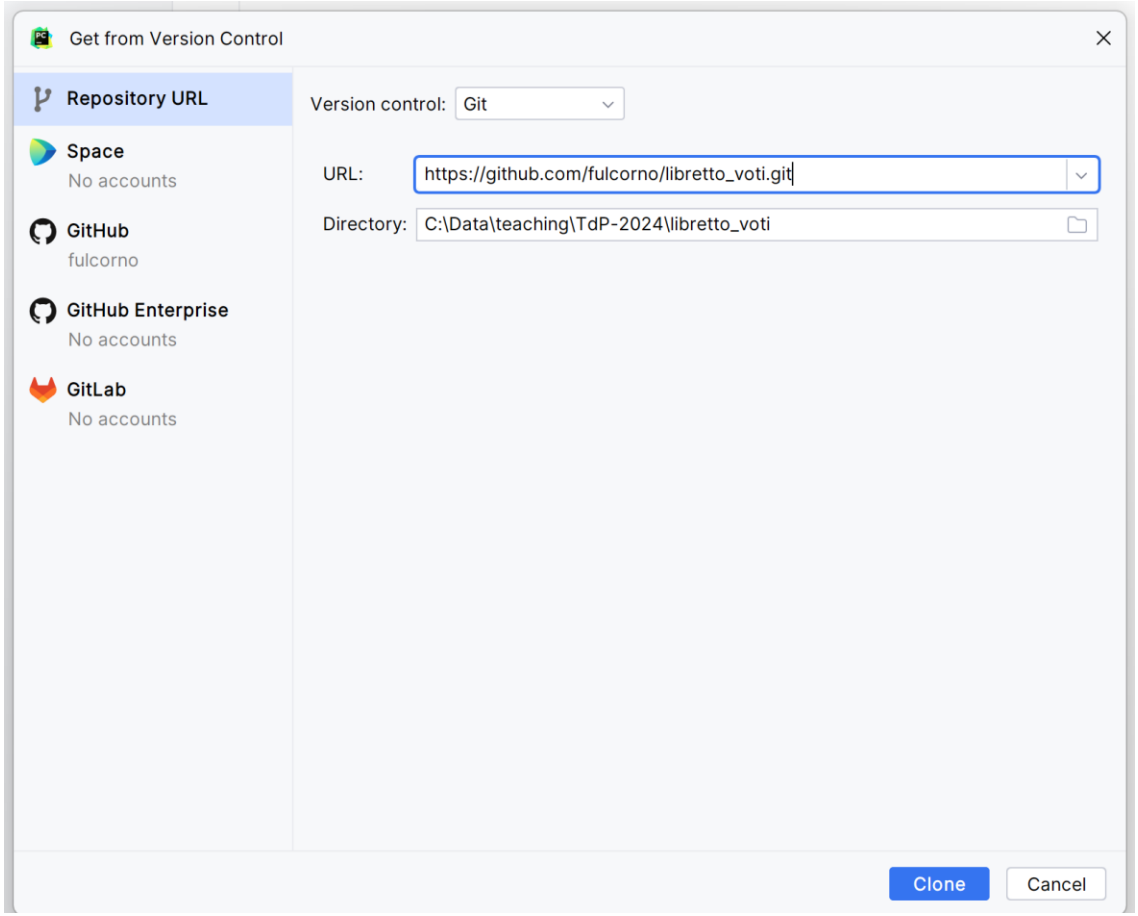


Copy the new project URL



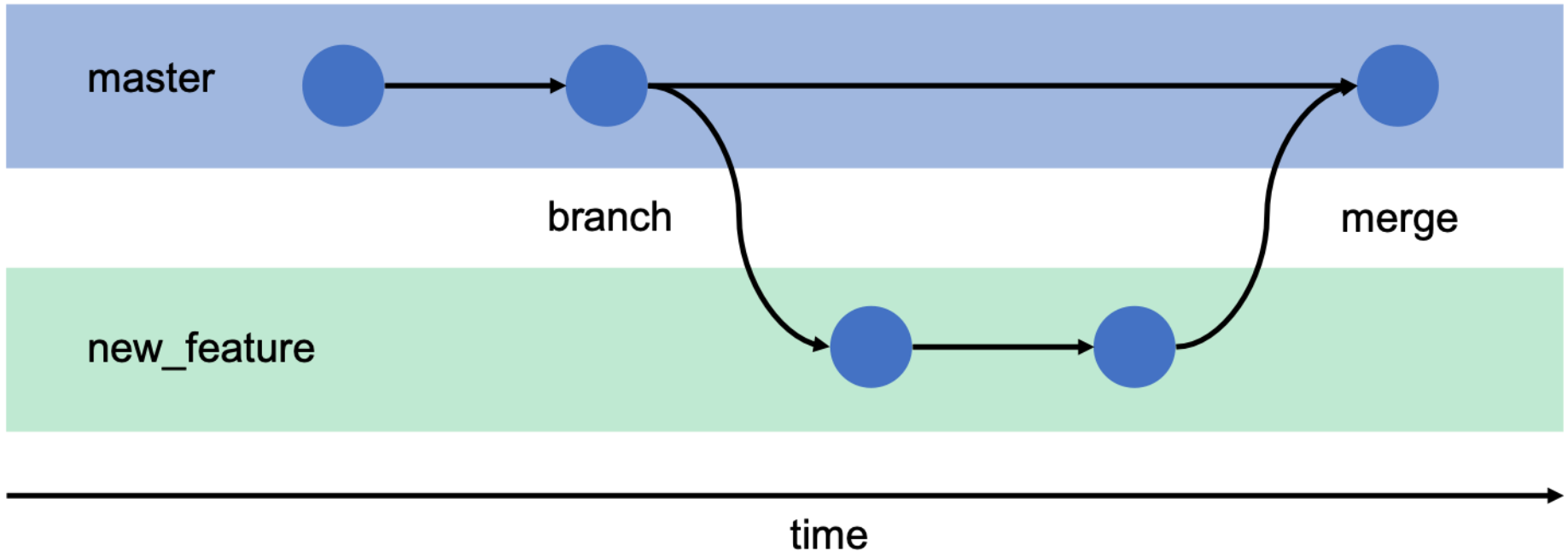
Clone the project





MORE ADVANCED GIT

Branches



Branches… in brief

- used to develop features isolated from each other
- the *main (or master)* branch is the “default” branch when you create a repository
  - you should use other branches for development and merge them back to the master branch upon completion
- Branches can be local (your local repo) or may be pushed to GitHub

LINKS AND REFERENCES

References

- Git Reference
  - http://gitref.org/
- Git - the simple guide
  - http://rogerdudler.github.io/git-guide/
- Git Documentation
  - http://git-scm.com/docs
- Pro Git (online book)
  - http://git-scm.com/book
- Version Control by Example (online book)
  - http://www.ericsink.com/vcbe/

References

- Try Git!
  - http://try.github.io/
- Various Git resources
  - https://help.github.com/articles/what-are-other-good-resources-for-learning-git-and-github
- A successful Git branching model
  - http://nvie.com/posts/a-successful-git-branching-model/
- Some Git (graphical) clients
  - http://git-scm.com/downloads/guis

License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/