Git

Amir Dirin

Spring 2024

About Git

- Created by Linus Torvalds, creator of Linux, in 2005
  - Came out of Linux development community
  - Designed to do version control on Linux kernel

- Goals of Git:
  - Speed
  - Support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects efficiently

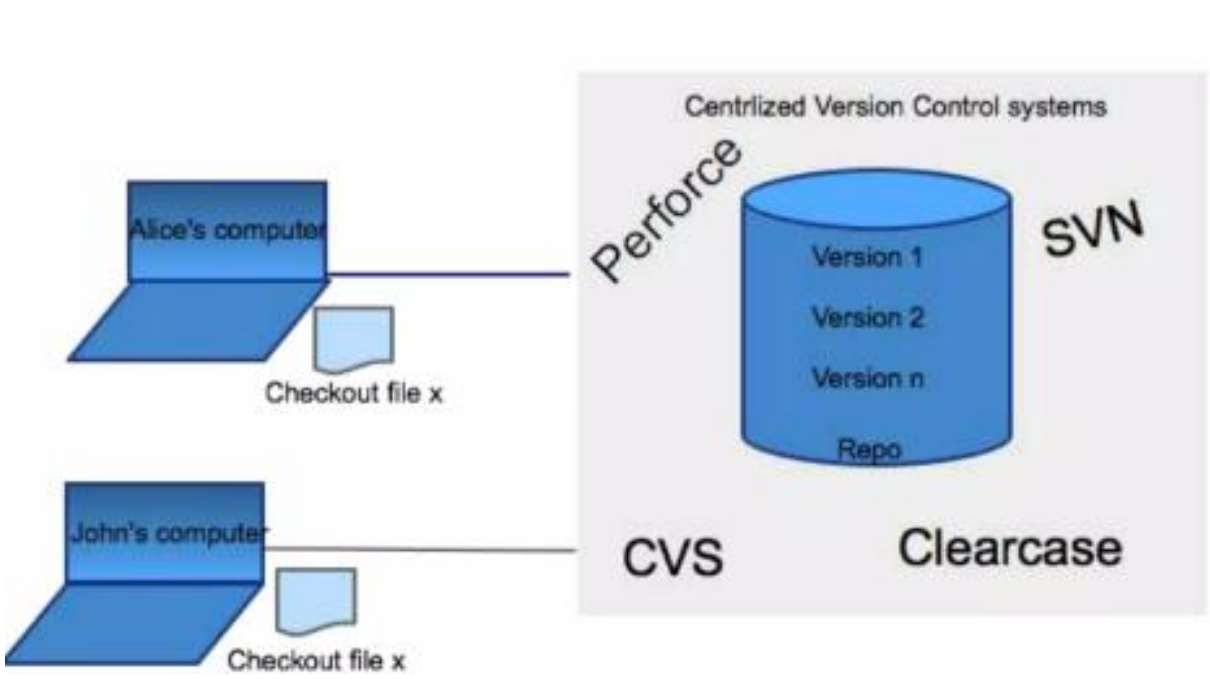
  - *(A "git" is a cranky old man. Linus meant himself.)*

Installing / learning Git

- Git website : http://git-scm.com/
- Free on-line book: http://git-scm.com/book
- Reference page for Git: http://gitref.org/index.html
- Git tutorial: http://schacon.github.com/git/gittutorial.html
- Git for Computer Scientists:
    - http://eagain.net/articles/git-for-computer-scientists/
- At the command line: (where verb = config, add, commit, etc.)
    - *git help verb*

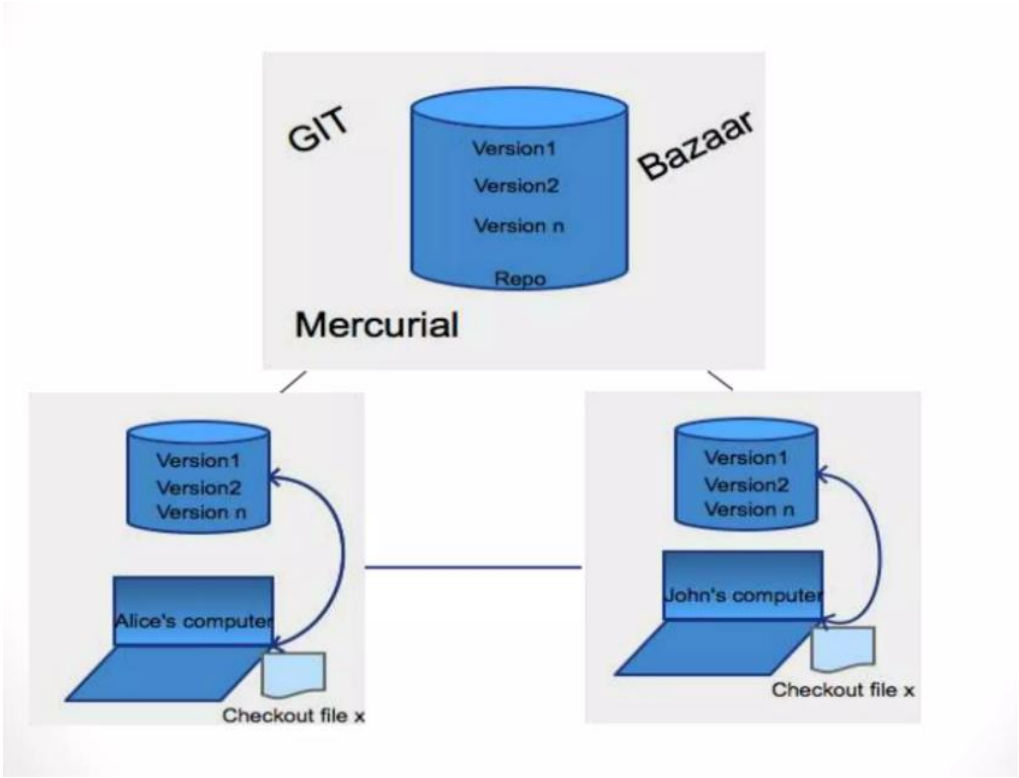
Centralized VCS

- Is a version control system where a central repository serves as the single point of truth for the entire project source code.
  - Developer retrieves code
  - Make changes locally
  - And then commit back to central repository



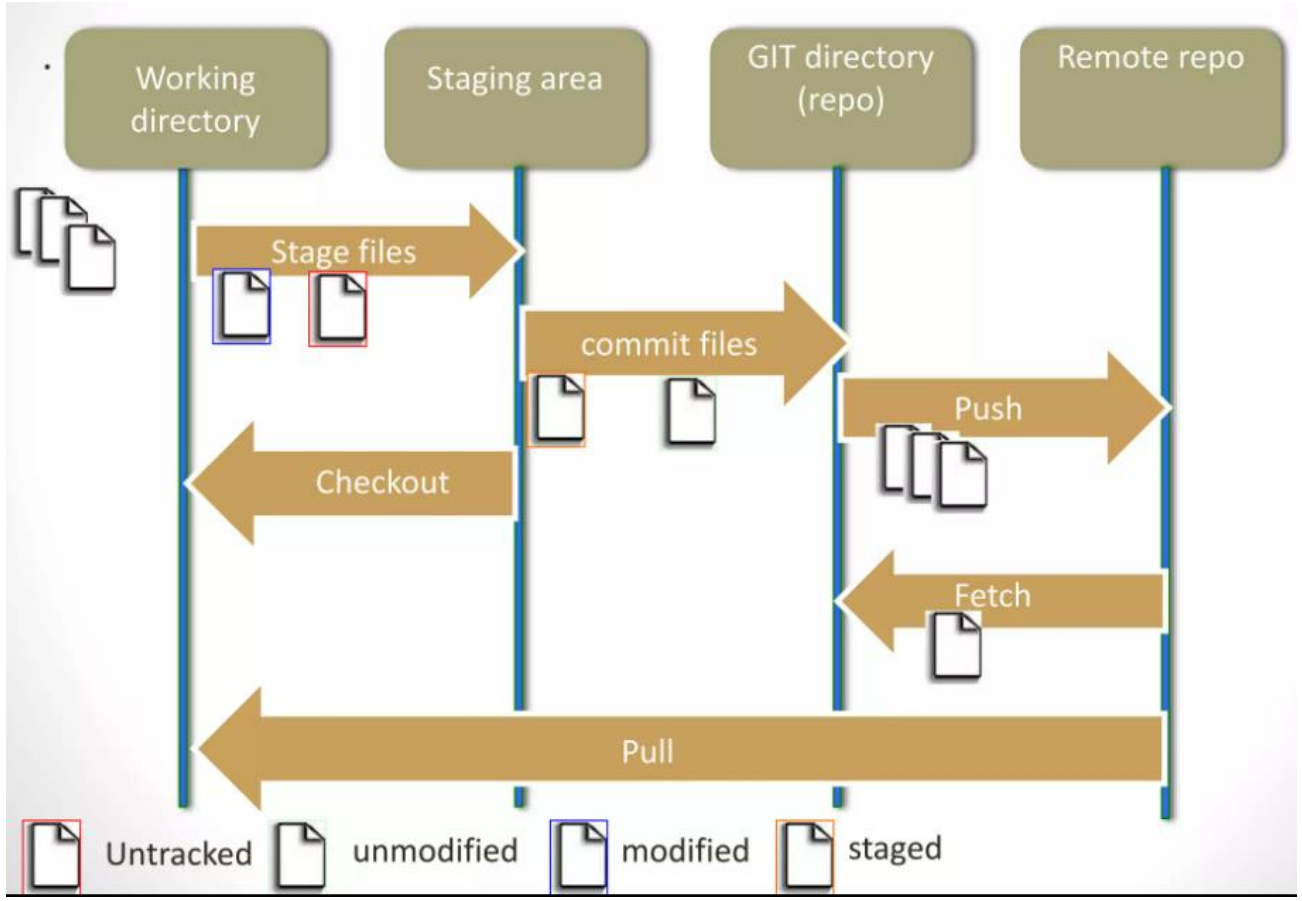
Distributed VCS

- In DVCS each user has a complete copy of the entire repository, including its full history.
  - Developer works independently and offline
  - Changes can be synchronized between repositories in a decentralized manner

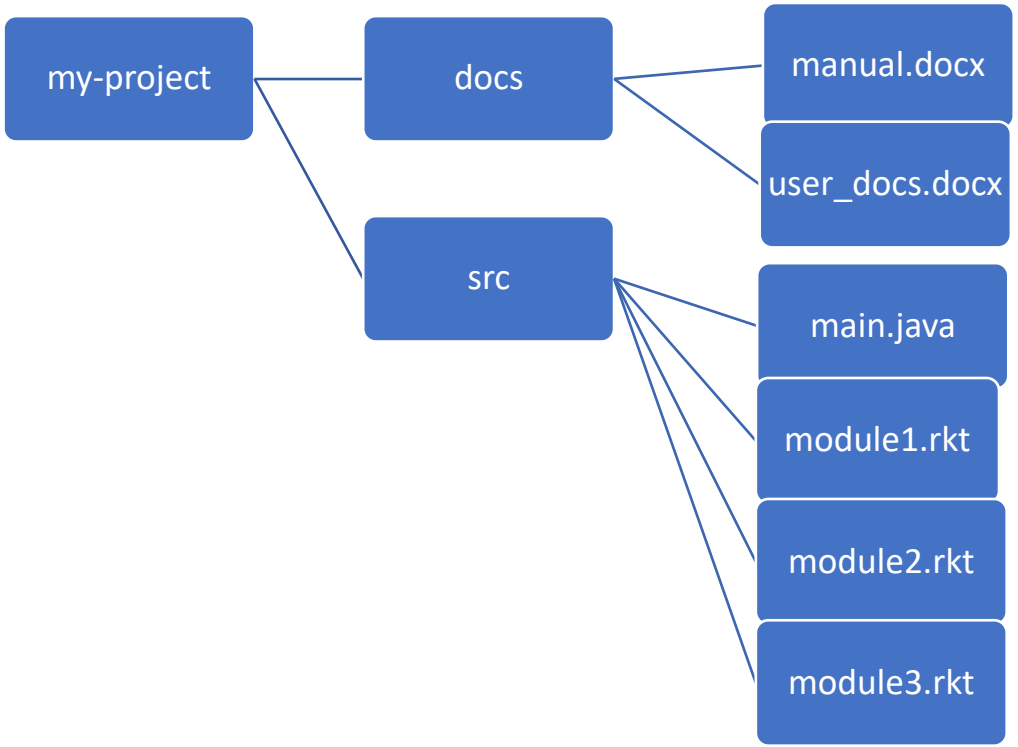


GIT state operation

- Git branching
  - Lightweight movable pointers for different commits
  - By default the branch name is called “master”
  - You can create as many branches as you need merge them or delete
  - Margining of branches is easy

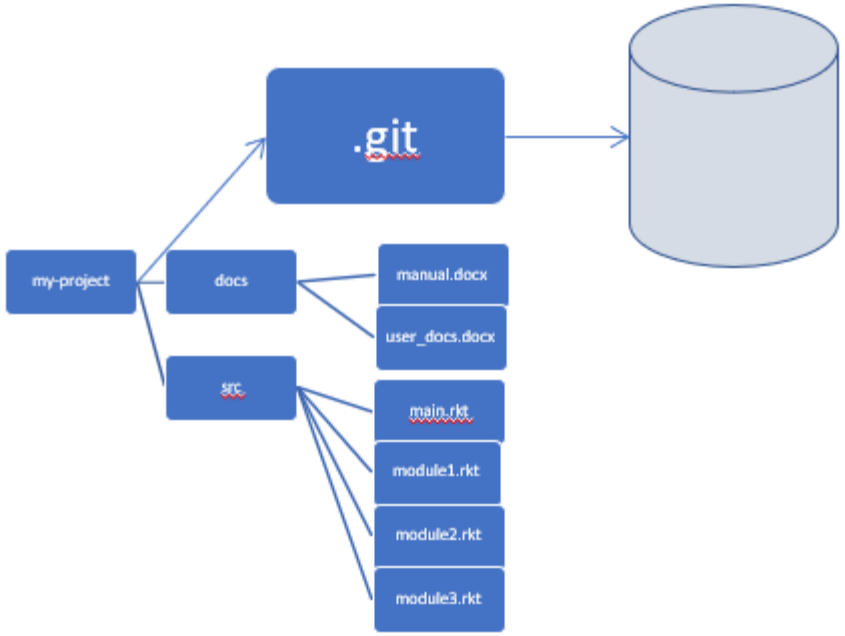


Your files



Here are your files, sitting in a directory called my-project

Your files in your git repository



When you have a git repository, you have an additional directory called .git, which points to a mini-filesystem.

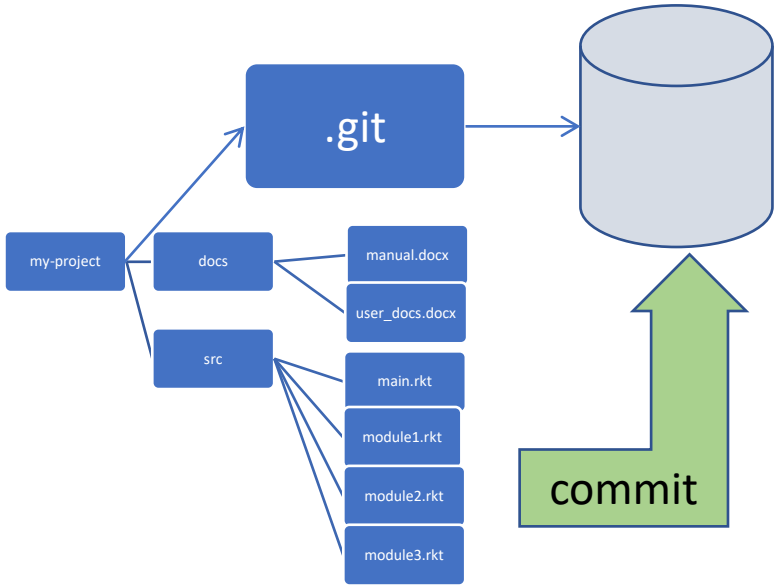
This file system keeps all your data, plus the bells and whistles that git needs to do its job.

All this sits on your local machine.

This mini-filesystem is highly optimized and very complicated. Don't try to read it directly.

The job of the git client (either Github for Windows, Github for Mac, or a suite of command-line utilities) is to manage this for you.

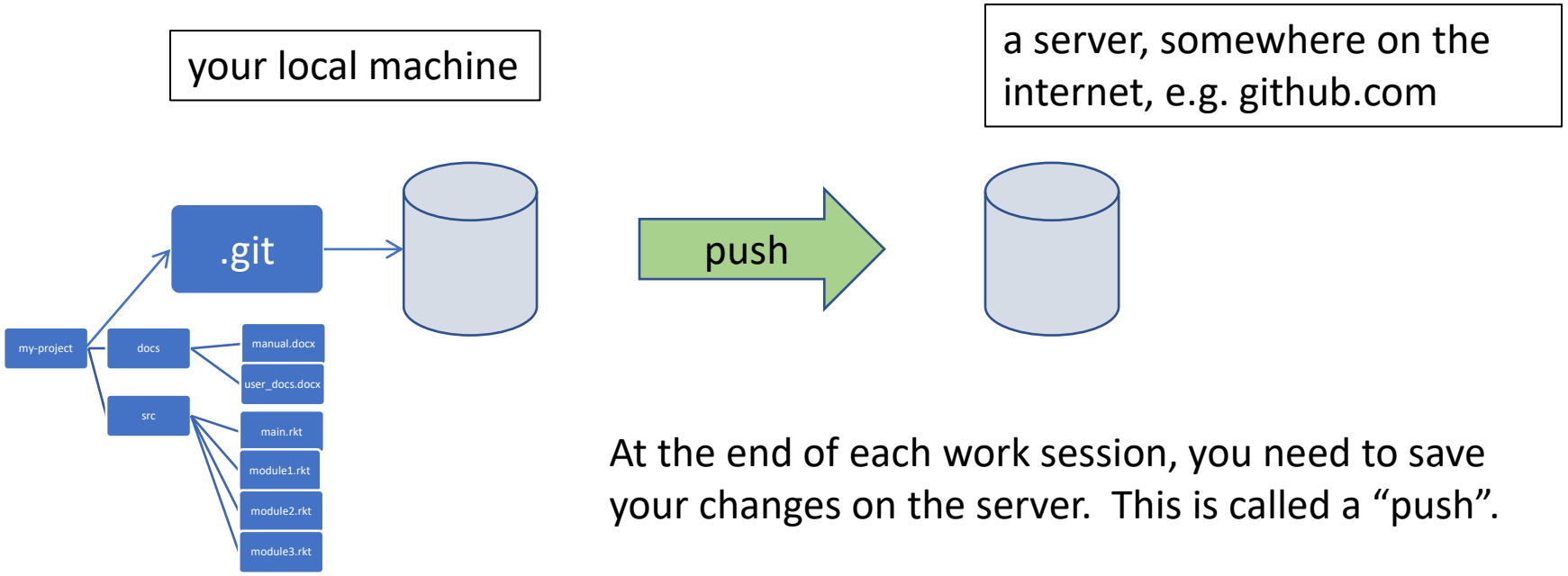
A Commit



When you do a “commit”, you record all your local changes into the mini-fs.

The mini-fs is “append-only”. Nothing is ever over-written there, so everything you commit can be recovered.

Synchronizing with the server (1)

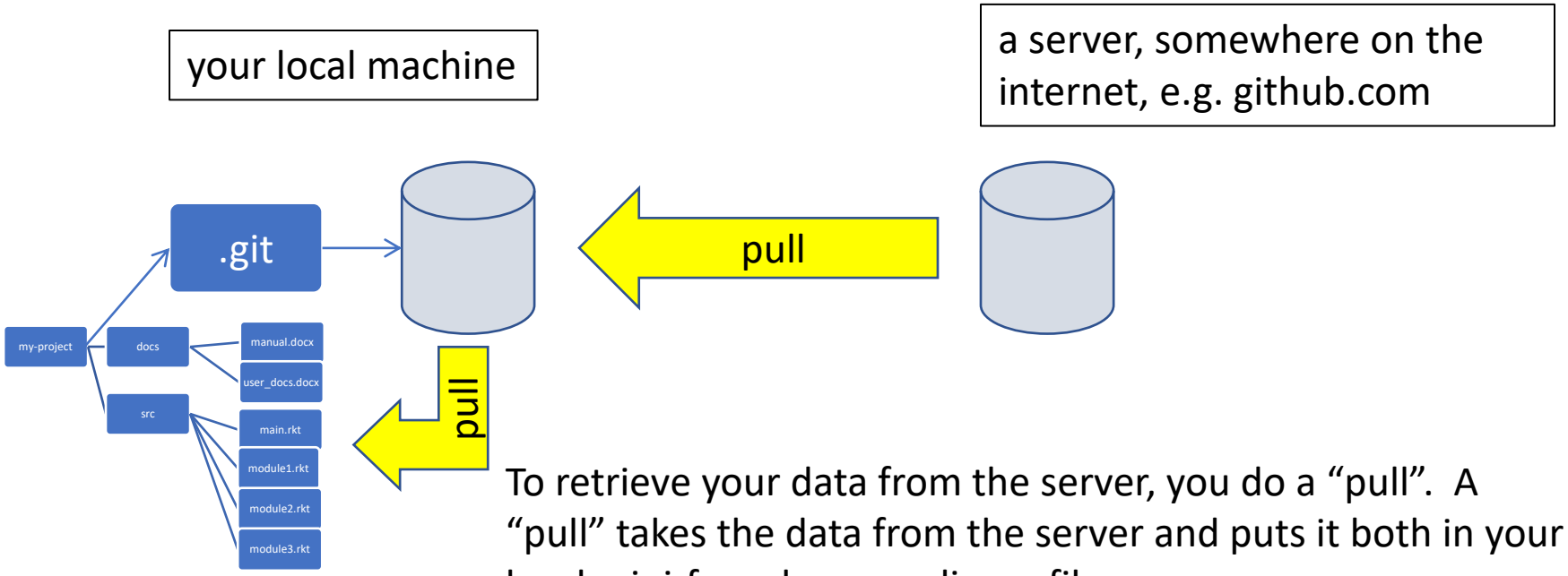


At the end of each work session, you need to save your changes on the server. This is called a “push”.

Now all your data is backed up.

- You can retrieve it, on your machine or some other machine.
- We can retrieve it (that’s how we collect homework)

Synchronizing with the server (2)

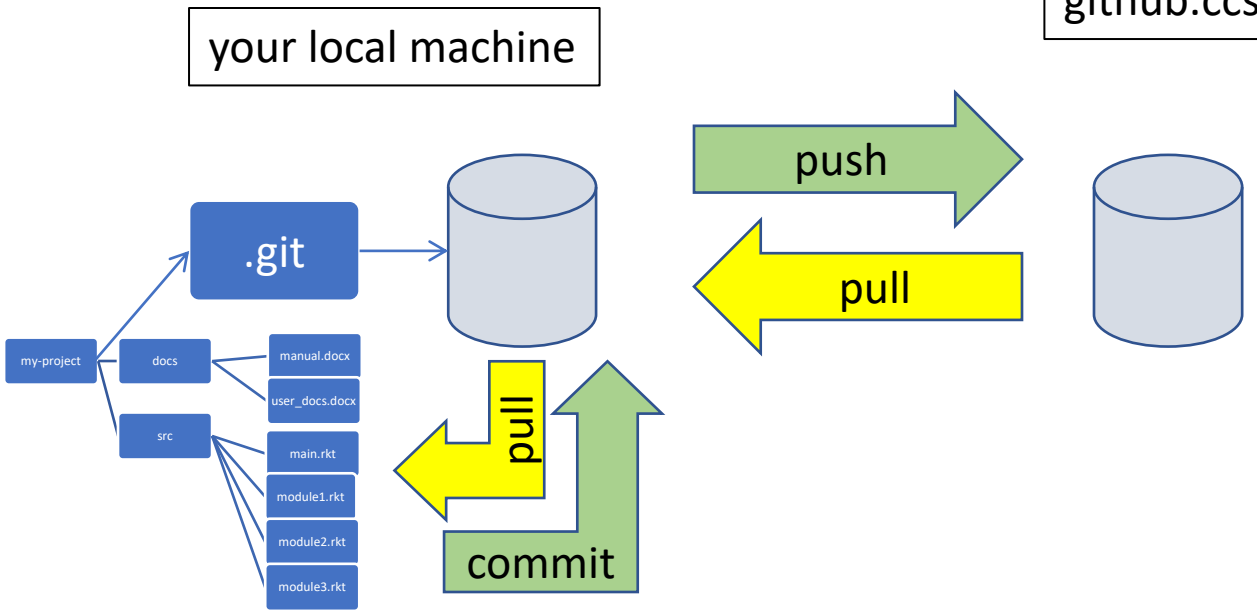


To retrieve your data from the server, you do a “pull”. A “pull” takes the data from the server and puts it both in your local mini-fs and your ordinary files.

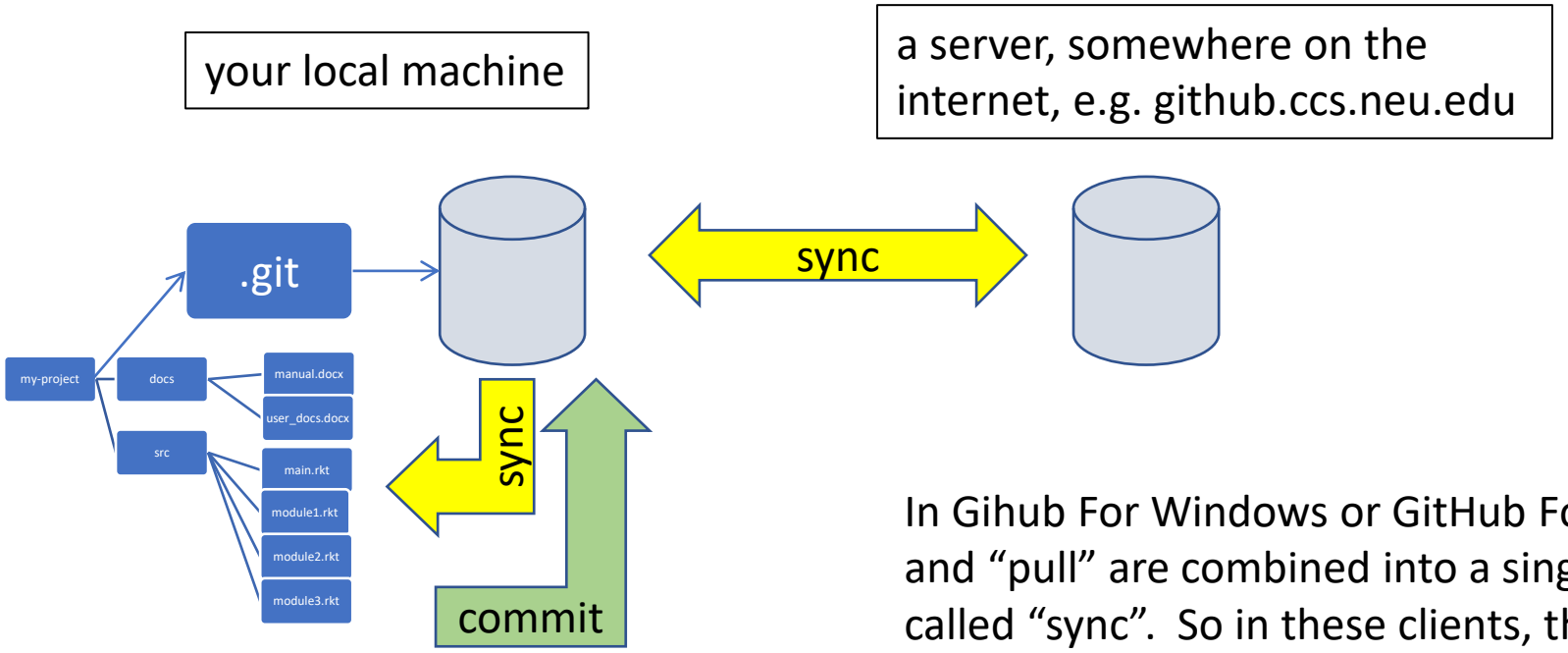
If your local file has changed, git will merge the changes if possible. If it can’t figure out how to merge, you will get an error message. We'll learn how to deal with these in the next lesson.

The whole picture

a server, somewhere on the internet, e.g. github.ccs.neu.edu

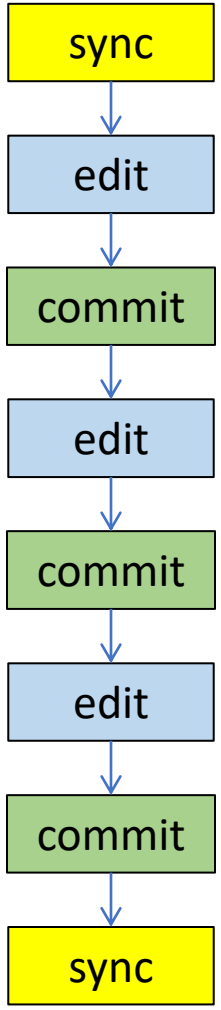


The whole picture using GHFW



In Gihub For Windows or GitHub For Mac, “push” and “pull” are combined into a single operation called “sync”. So in these clients, there are only two steps (“commit” and “sync”) to worry about, not three.

Your workflow (2)



Best practice: commit your work whenever you've gotten one part of your problem working, or before trying something that might fail.

If your new stuff is screwed up, you can always "revert" to your last good commit. (Remember: always "revert", never "roll back")

Your workflow with a partner