A Simple Introduction to Git: a distributed version-control system

CS 5010 Program Design Paradigms
“Bootcamp”

Lesson 0.5

Learning Objectives

- At the end of this lesson you should be able to explain:
  - how git creates a mini-filesystem in your directory
  - what commit, push, pull, and sync do
  - the elements of the basic git workflow
  - how git allows you to work across multiple computers
  - how git allows you and a partner to work together

Git is a distributed version-control system

- You keep your files in a *repository* on your local machine.
- You synchronize your repository with a repository on a server.
- If you move from one machine to another, you can pick up the changes by synchronizing with the server.
- If your partner uploads some changes to your files, you can pick those up by synchronizing with the server.

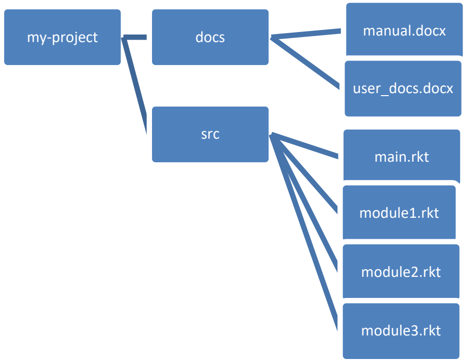
Git is a distributed version-control system

- Terminology: In git-speak, a “version” is called a “commit.”
- Git keeps track of the history of your commits, so you can go back and look at earlier versions, or just give up on the current version and go back some earlier version.

A simple model of git

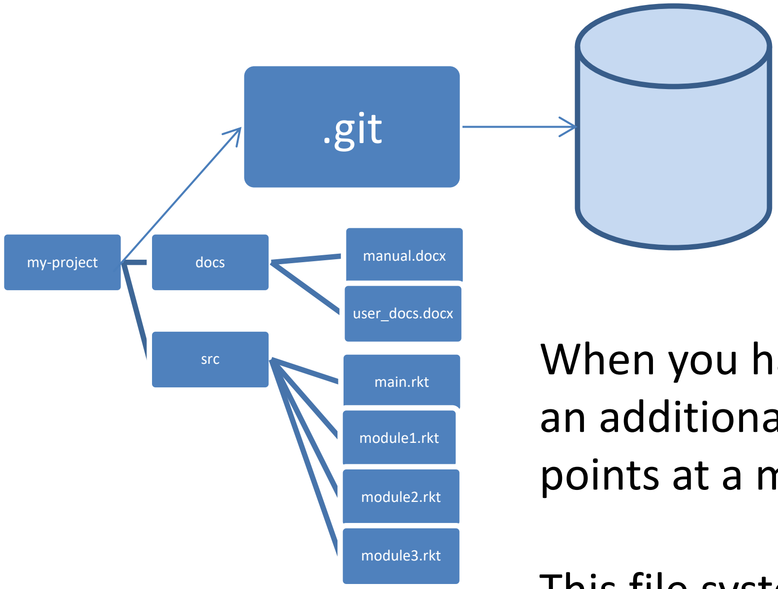
- Most git documentation gets into details very quickly.
- Here’s a very simple model of what’s going on in git.

Your files



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

Your files in your git repository

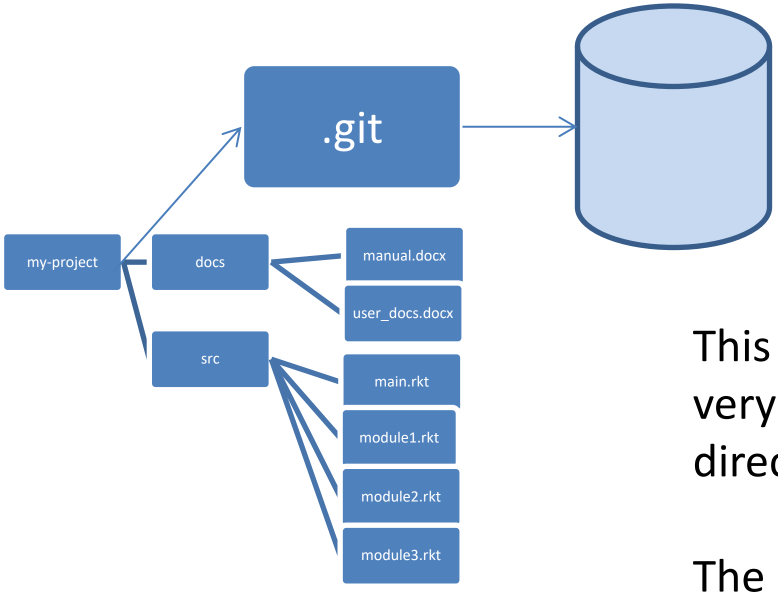


When you have a git repository, you have an additional directory called .git, which points at a mini-filesystem.

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.

The git client



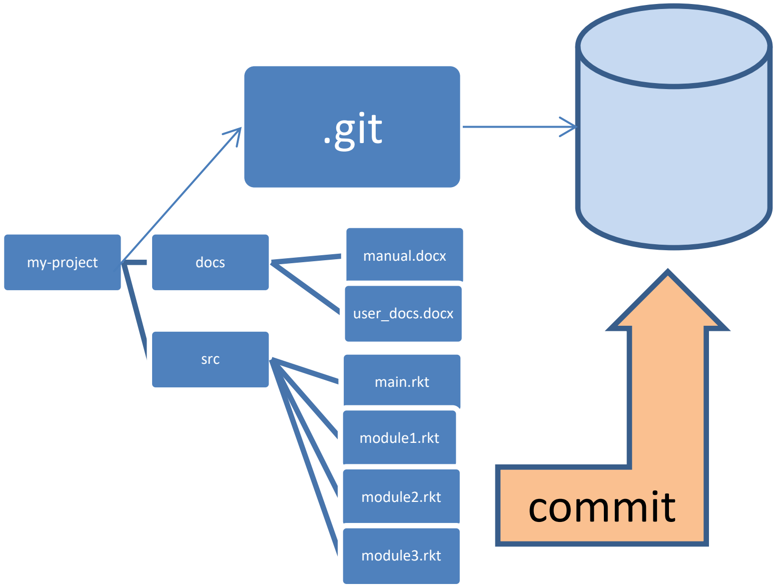
This mini-filesystem is highly optimized and very complicated. Don't try to read it directly.

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.

Your workflow (part 1)

- You edit your local files directly.
  - You can edit, add files, delete files, etc., using whatever tools you like.
  - This doesn’t change the mini-filesystem, so now your mini-fs is behind.

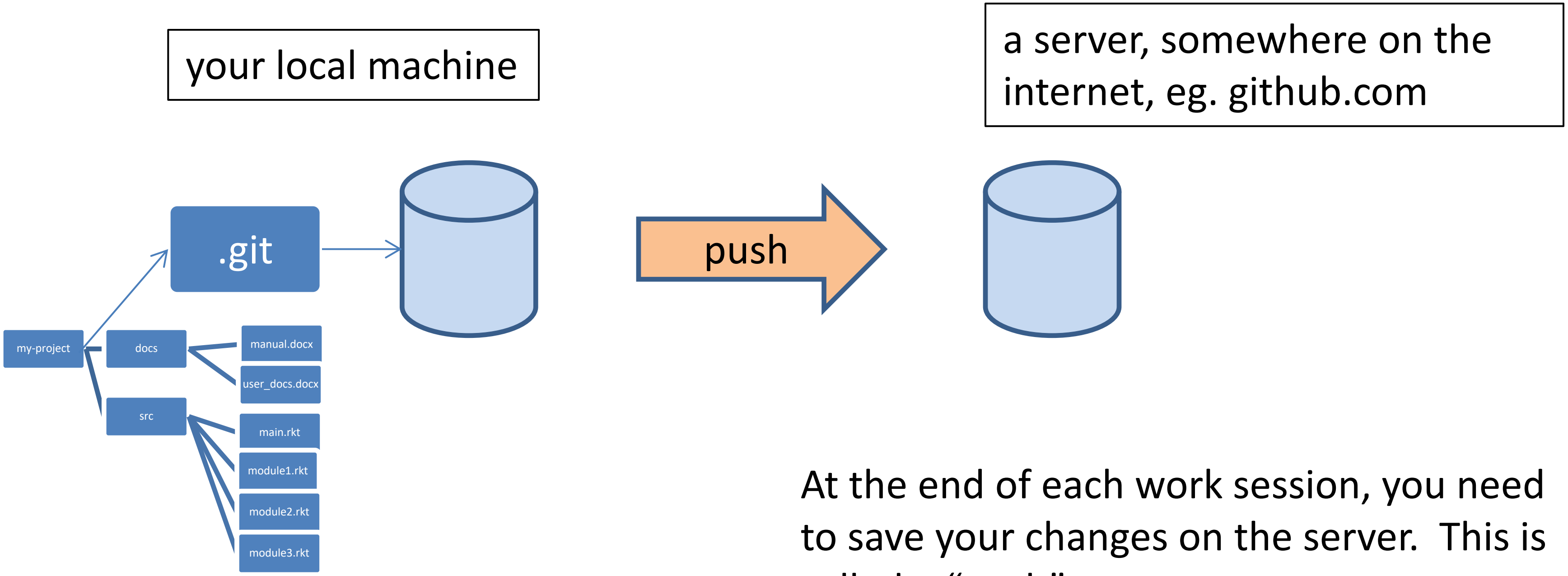
A Commit



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 ever commit can be recovered.

Synchronizing with the server (1)

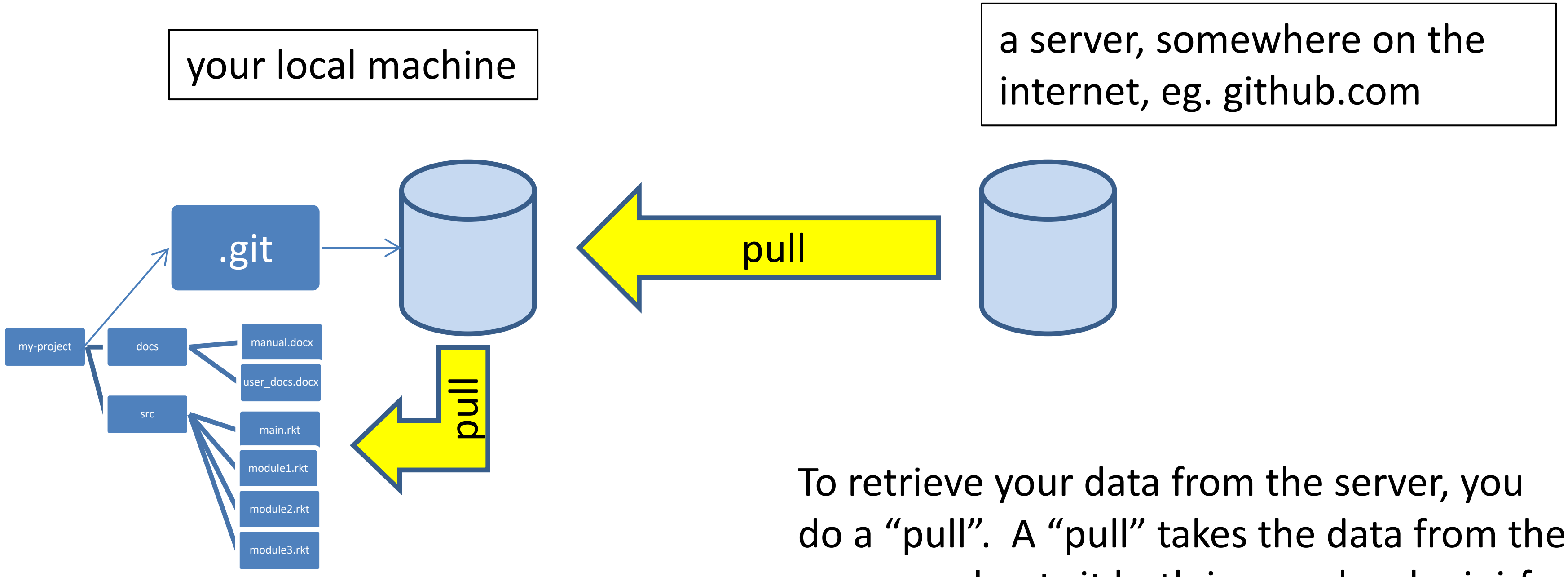


At the end of each work session, you need to save your changes on the server. This is called a “push”.

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)

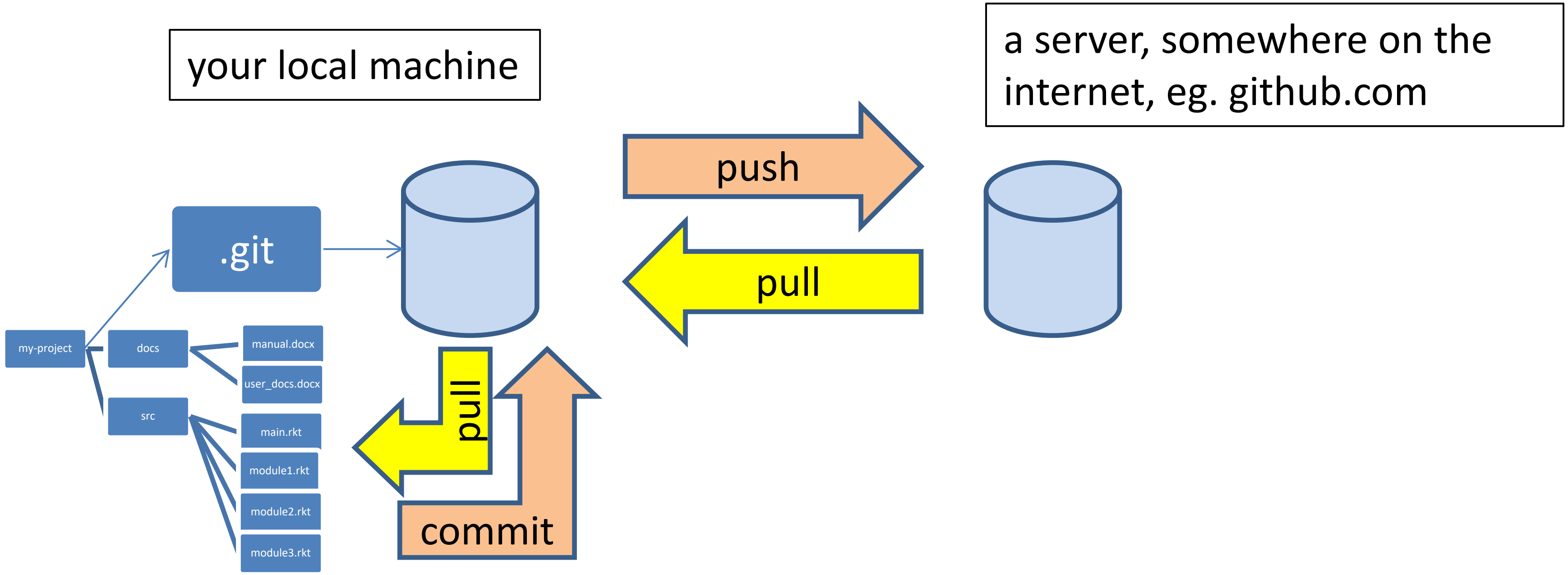
Synchronizing with the server (2)



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 in your ordinary files.

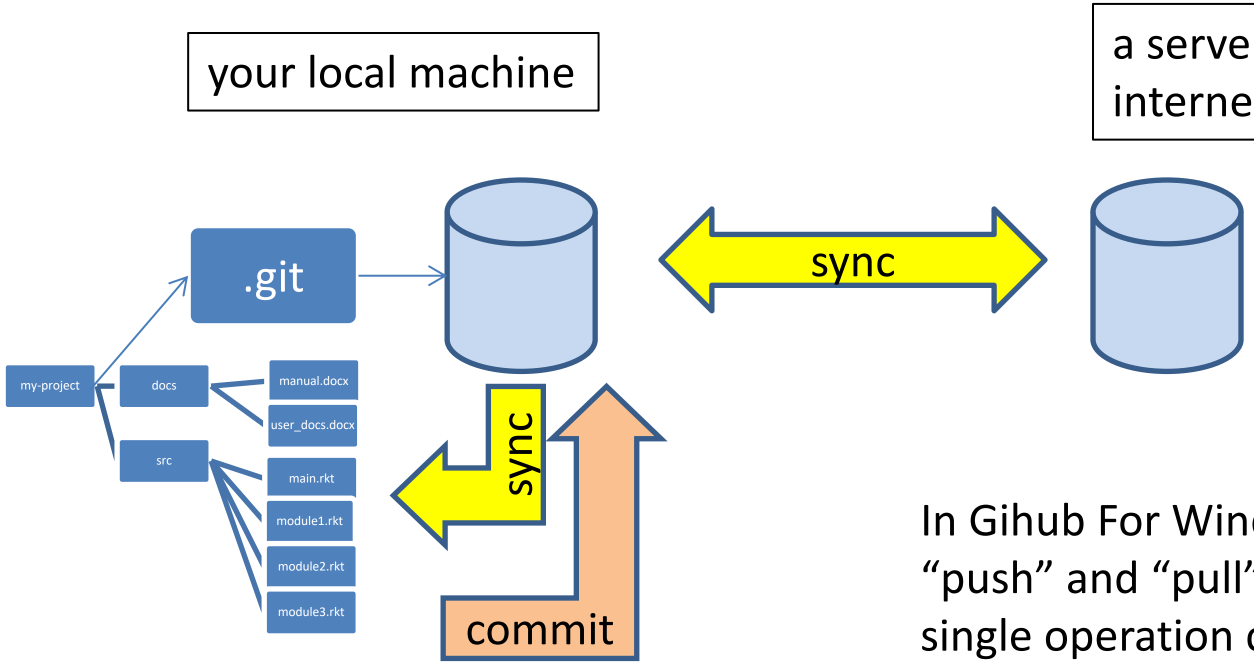
If your local file has changed, git will merge the changes if possible. If it can’t figure out how to the merge, you will get an error message. We'll learn how to deal with these in the next lesson.

The whole picture



The whole picture using GHFW

a server, somewhere on the internet, eg. github.com



your local machine

.git

my-project

docs

manual.docx

user_docs.docx

src

main.rkt

module1.rkt

module2.rkt

module3.rkt

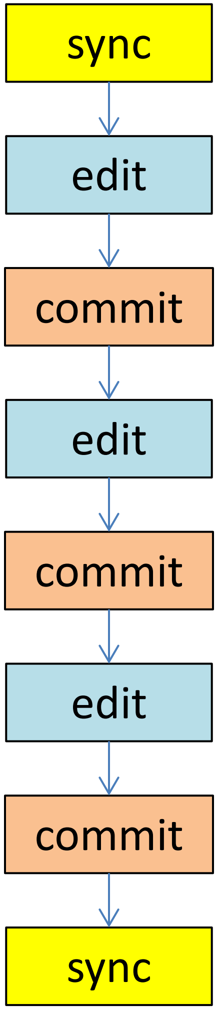
sync

sync

commit

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)



sync

edit

commit

edit

commit

edit

commit

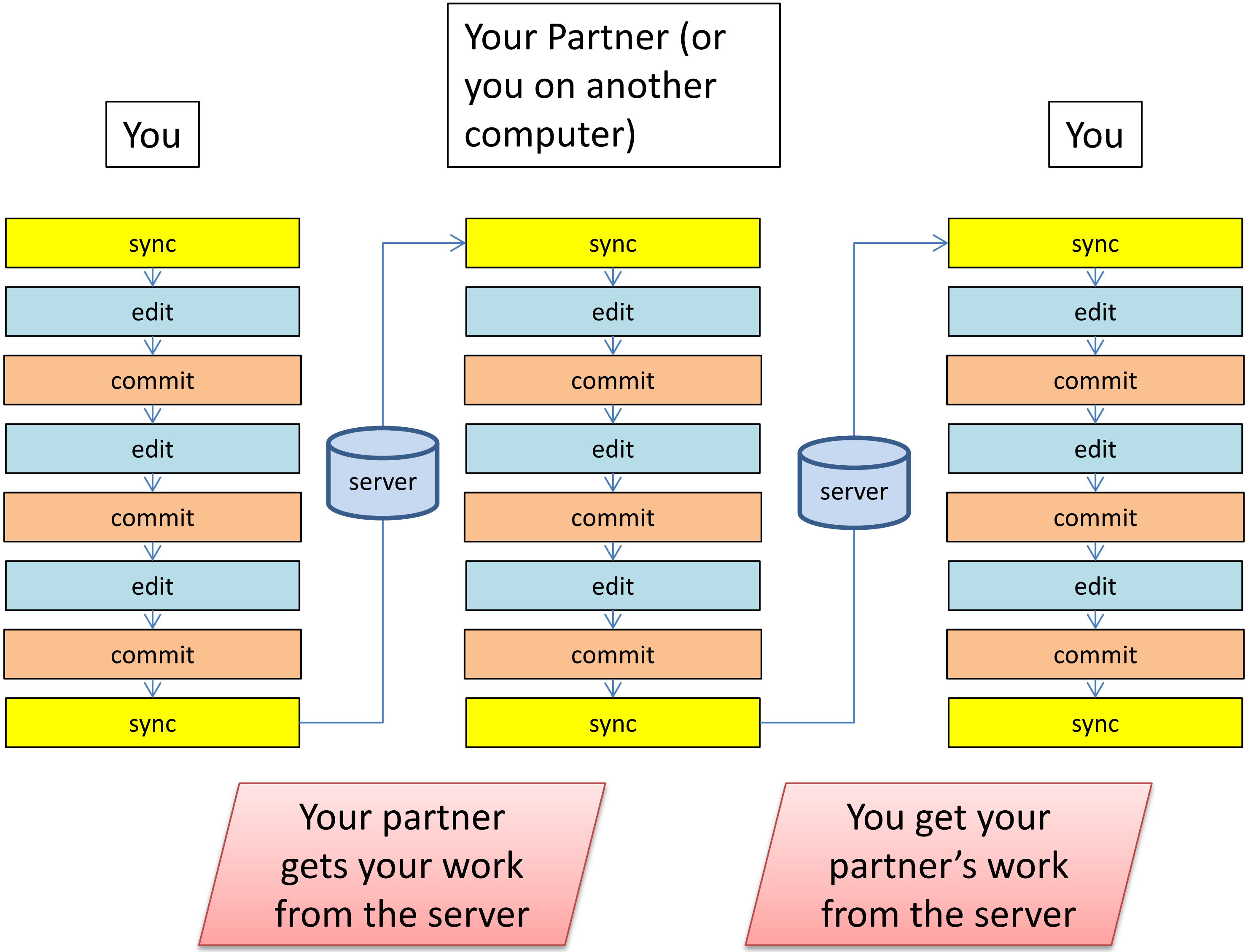
sync

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”)

Using Github for Windows/Mac

Your workflow with a partner



Summary

- In this lesson you have learned
  - that git creates a mini-filesystem in your directory
  - what commit, push, pull, and sync do
  - the elements of the basic git workflow
  - how git allows you to work across multiple computers
  - how git allows you and a partner to work together