Computer System Organization
Recitation
[Spring 2018]
CSCI-UA 0201-001

R1: Introduction

Know your staffs

- Instructor: Prof. Shuai Mu
  - Recitation Instructor: Conrad Christensen
  - cc5608@nyu.edu
  - Office Hour: Mon 12:30-1:30pm (60 5th Ave Office room 406)
- Grading Assistants:
  - Conrad Christensen
  - Lamont Nelson: Th 4-5pm (60 5th Ave Office room 402)

What is this recitation for?

- Exercises that will help you understand CSO more.
- Tutorial of labs.
- Review of midterm I and II.

How are we going to proceed?

- Problems driven.
  - Except for today.

- If you are confident that you can solve all the exercises and labs by YOURSELF and don’t need extra guides through the course, you can skip the recitation.

- Exercises will be posted every Sunday (possibly earlier) night and the deadline will be the next Wednesday (day of recitation) midnight.

Academic Integrity

- Do not use/look code from any other places, e.g., Internet, your friends.
- We will check.
- We will use a tool to check your code.
- We will use a tool to check your code without further notification (except for today).
- You may fail the class because of plagiarism. Someone did…
- Just don’t do it!

Let’s do some statistics

- How many of you have never used Unix-like systems?
  - How many of you have never used command line?
- How many of you have never programmed in C or C++?
  - How many of you have never programmed in Python?
  - Other languages?
- How many of you have never used version control softwares?
  - How many of you have never used Git?

Let’s begin

- How to setup your repo and submit your code?
- Unix/command line
- Program development
  - Editor (vim)
  - Compile
  - Debug
  - Version control (Git)
- **Goal:**
  - **Setup your recitation-USERNAME repo.**
  - **Submit modified Makefile, fixed foo.c and hello.c.**

Starting up VM

- Open a terminal
- sudo apt-get update
- sudo apt-get install gcc gdb make git

How to initialize recitation and lab repositories

- **[YOU SHOULD DO THIS ONLY ONCE.]**

- labs
    1. Click the link (https://classroom.github.com/a/uORLsCac).
    2. You will be asked to login to your GitHub account if you have not login.
    3. You will be asked to link your NYU ID to your GitHub account. Do not skip this step. CHOOSE YOUR NYU ID. (If you make a mistake, contact us immediately through cso-staff email. If you can't find your NYU ID, also contact us immediately.)
    4. Accept the invitation.

- recitation
    1. Click the link (https://classroom.github.com/a/cghx9W-s)

How to initialize recitation and lab repositories

- **cd /home/lab/** [change the current folder to lab which is under home folder]
  - If you want to share the folder with your Mac or Windows, check the information on class website and change the path accordingly.
- **git config --global user.email “your_emal@example.com“**
- **git config --global user.name “Your name"**
- **git clone https://github.com/nyu-cso18spring/recitation-<YourGitHubUserName>** [clone your recitation folder from GitHub]
- **cd recitation-<YourGitHubUserName>** [change the current folder to your recitation folder]
- **git remote add upstream https://github.com/nyu-cso18spring/recitation** [add the class repository as the upstream]
- **git pull upstream** [get the latest assignments from the class repository]
- **ls** [check what are in the folder]
- [change recitation to labs above to initialize labs repository]

How to get the latest assignment

- **cd /home/lab/recitation (or labs)**
- **git pull upstream**
- **ls**

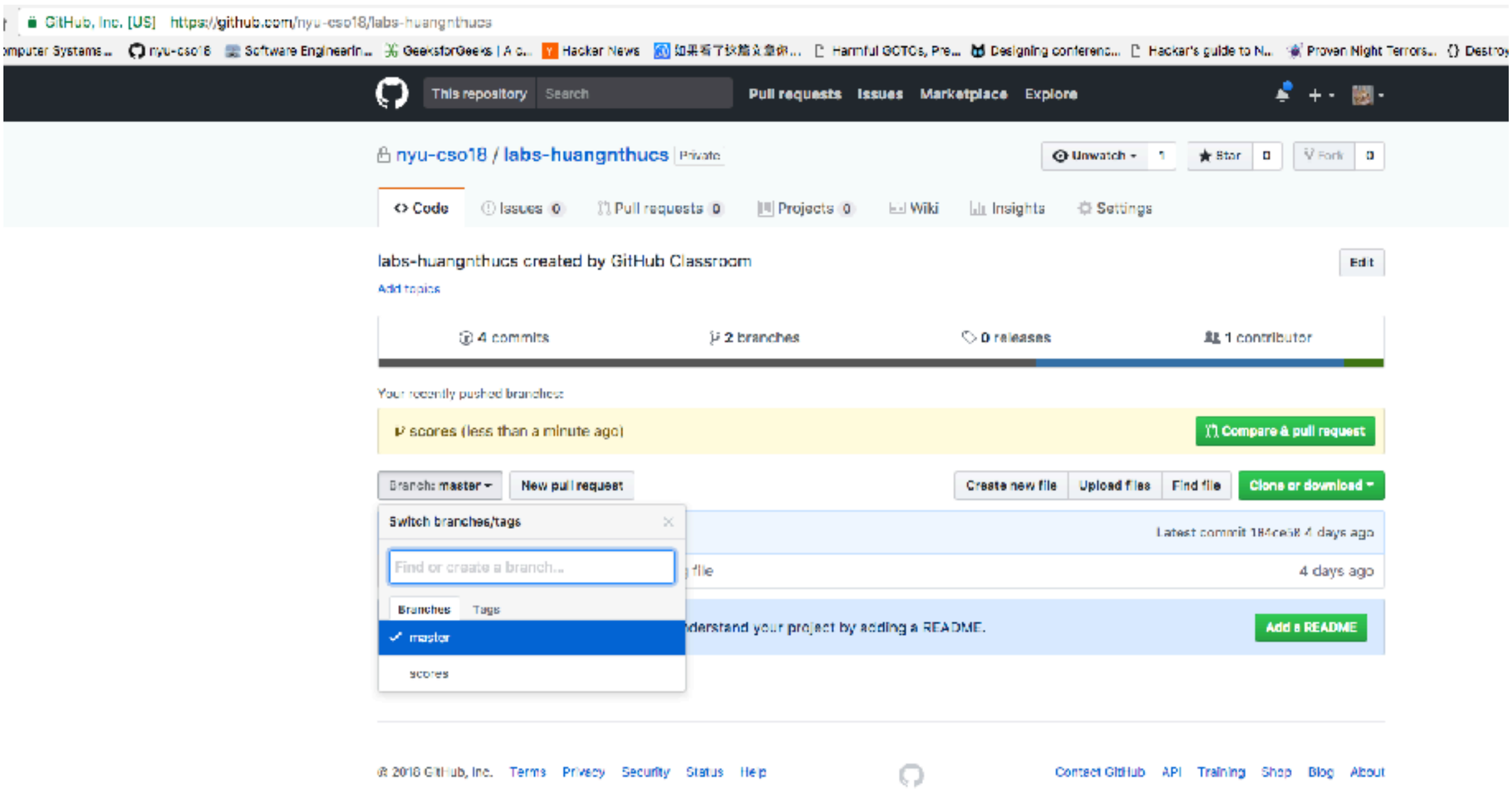
How to submit assignments

- **[Change to the assignment folder first (cd XXX).]**
- **git add <FILES>** [add <FILES> to list to be committed]
- **git commit -m “lab1 final commit”** [commit <FILES>]
  - This step and the previous step can be done more than once.
- **git push origin** [submit your commit! don’t forget to do this!!!!!]

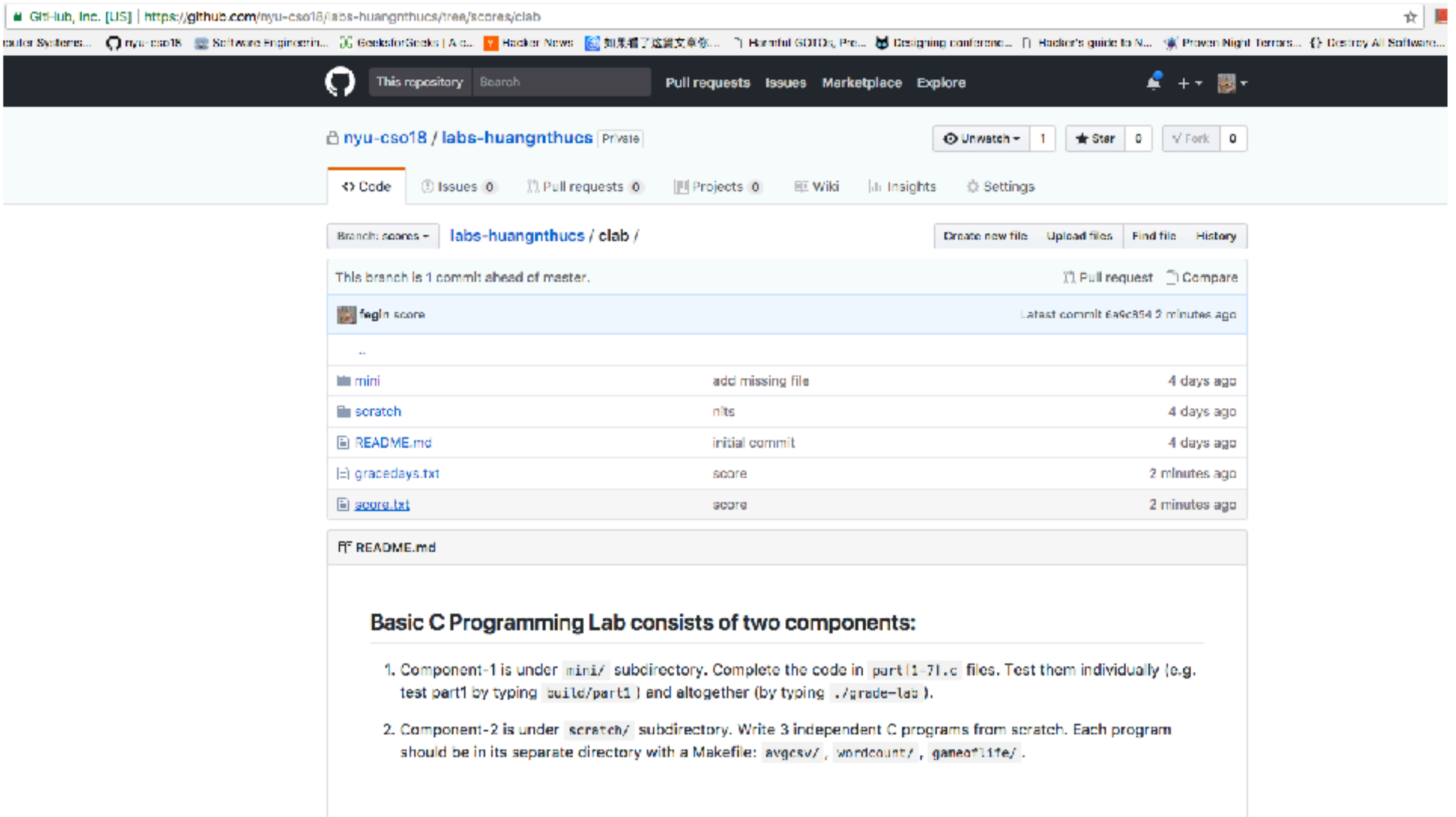
How to use grace days (only for labs)

- **[DO THIS RIGHT AFTER YOUR LAST SUBMIT.]**
- **[Change to the assignment folder first (cd XXX).]**
- **echo “0.5” > gracedays.md** [0.5 means 12 hours. The number can only be 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]
- **git add gracedays.md**
- **git commit -m “lab1 final commit”**
- **git push origin** [submit your commit! don’t forget to do this!!!!!]

How to get the scores



How to get the scores



How to recover the repo when something is wrong

- **[Google first!!!]**
- **[Change to the labs/recitation folder first (cd XXX).]**
- **cd ..** [change to the parent folder of current folder]
- **mkdir backup** [create a backup folder]
- **cp labs/<AssignmentFolder>/<ModifiedFile> backup**
  - e.g., cp labs/lab1/a.c labs/lab1/b.c backup
- **reinitialize labs(or recitation)**
- **cp backup/* labs/<AssignmentFolder>/**
  - e.g., cp backup/* labs/lab1/

Command line

- A **command-line interface** or **command language interpreter** (**CLI**), also known as **command-line user interface**, **console user interface**[1] and **character user interface** (**CUI**), is a means of interacting with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines).
—Wikipedia

Recitation README

- https://github.com/nyu-cso18spring/recitation/tree/master/r01

- Always check the latest readme (change r01 to rXX accordingly).

Editor

- A **text editor** is a type of program used for editing plain text files. Such programs are sometimes known as "**notepad**" software, following the Microsoft Notepad.
  —Wikipedia

- An **integrated development environment** (**IDE**) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools, and a debugger.
  — Wikipedia

- **A text editor is not an IDE but an IDE must contain a text editor.**

Compilation

- Compiler
  - A software that transforms computer code written in one programming language (the source language) into another computer language (the target language).
  - gcc / cc
- Interpreter
  - A software that directly executes instructions written in a programming/ scripting language without previously compiling them into a machine language program.
- Linker
  - A linker is a software that takes one or more object files and combines them into a single executable file, library file, or another object file.
  - gcc / ld

Compilation

```
int foo() {
   int i = 0;
   i += 1;
   return i;
}
```

foo.c

```
foo:
...
movl %eax 0
movl %ebx 1
addl %eax, %ebx
...
```

foo.o

a.out

```
int main() {
   foo()
}
```

main.c

```
main:
...
move %eax foo
call %eax
...
```

main.o

Make

- What is Make?
  - A build automation tool that automatically builds executable programs and libraries from source code.
- Why?
  - A project can contains a lot of source files.
  - Each source file may needs different compiler option.
  - Dependencies exists among source files.
- How?
  - Describe everything in a file, Makefile, and Make will do everything for you.

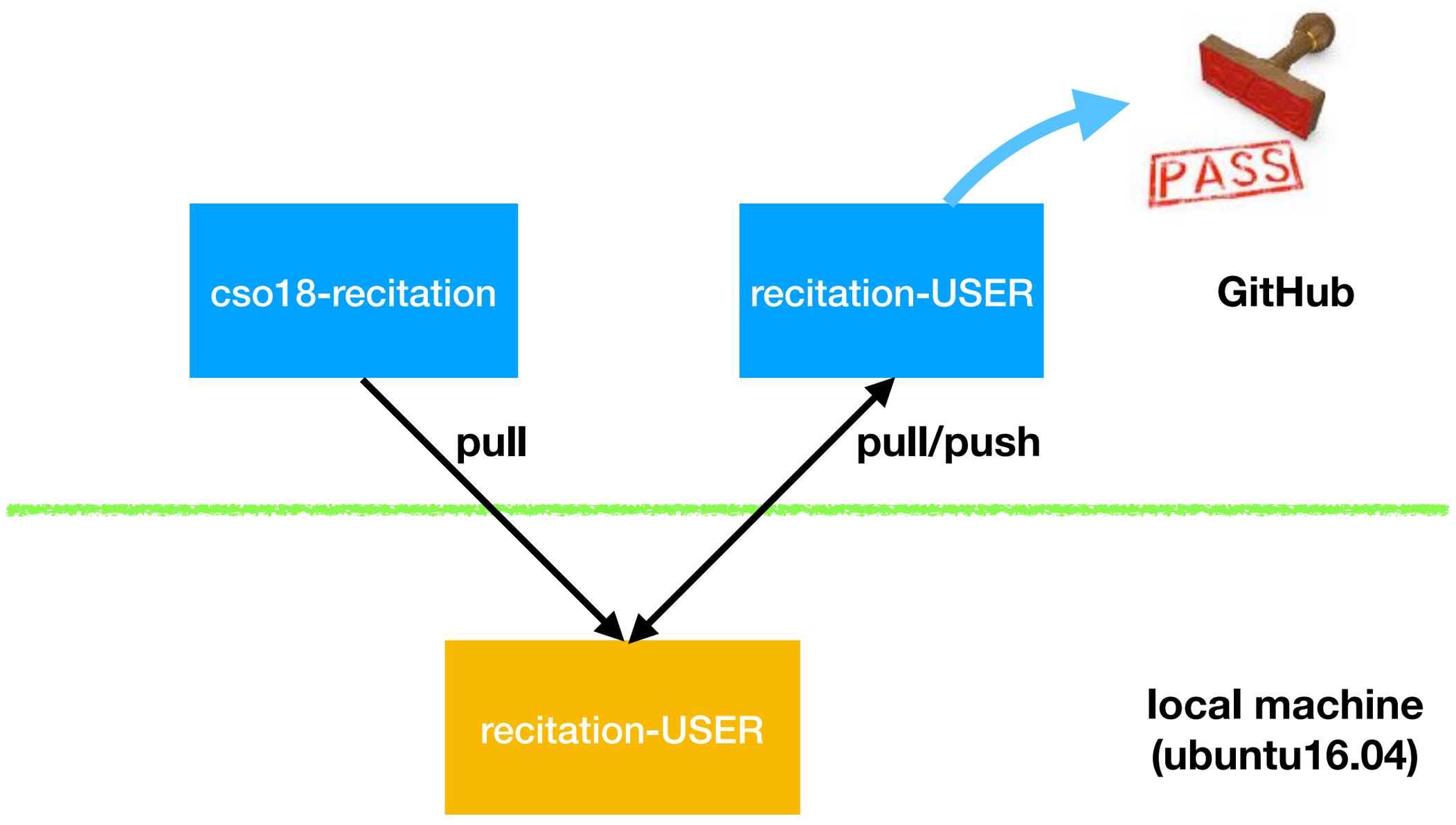
Make with automatic variables

- Why automatic variables?
  - Specifying how to compiler everything does not remove our burdens.
  - Automatic variables can help us unify the same type of files.
- How?
  - $@ (target name)
  - $^ (name of all pre-requisites)
  - pattern-matching using % and *.

Debug

- How to debug?
  - Print logs, observe and then debug.
  - Use a debugger to help you.
    - gdb
- How to use gdb?
  - First you need to ask gcc to add debug information when compiling the source files.
  - Debug!

Git status for our recitation and labs



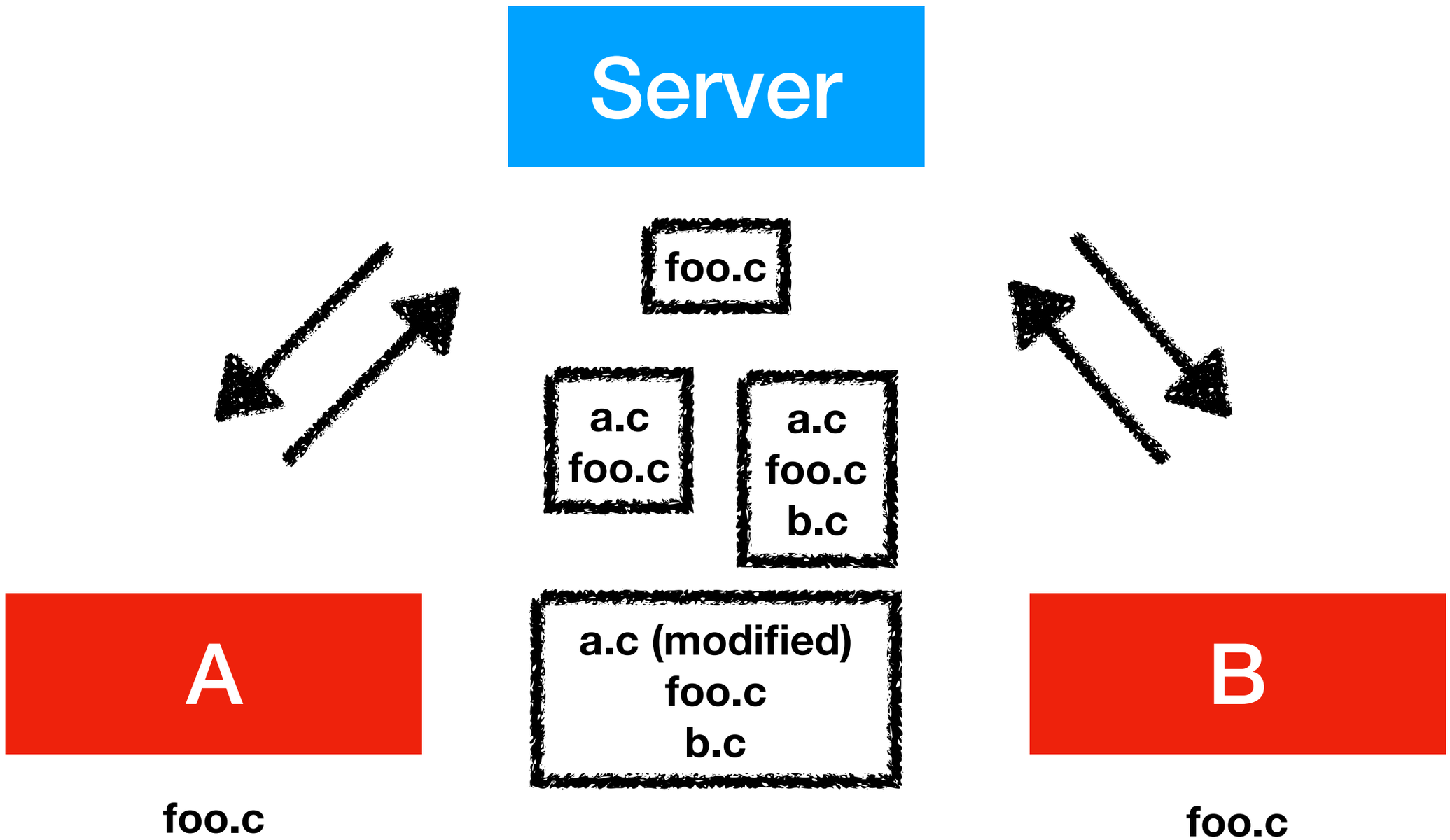
Version control system

- What?
  - Manages changes to documents, source files and other collections of information.
- Why?
  - Do you remember which source file you added/modified last week? Probably not.
  - Have you ever developed a project with other people? Coordinating programmers is hard.
- How?
  - CVS, SVN and GIT

Server/client version control system

- What?
  - A kind of version control system that puts all tracking metadata on a server. Clients can fetch/upload source files and information from the server.
- Why?
  - Strait-forward and easy to maintain.
  - Save space.

Version control system



Branching

- What?
  - The duplication of an object under version control so that modifications can happen in parallel along both branches.
- Why?
  - Developing different features in a same project.

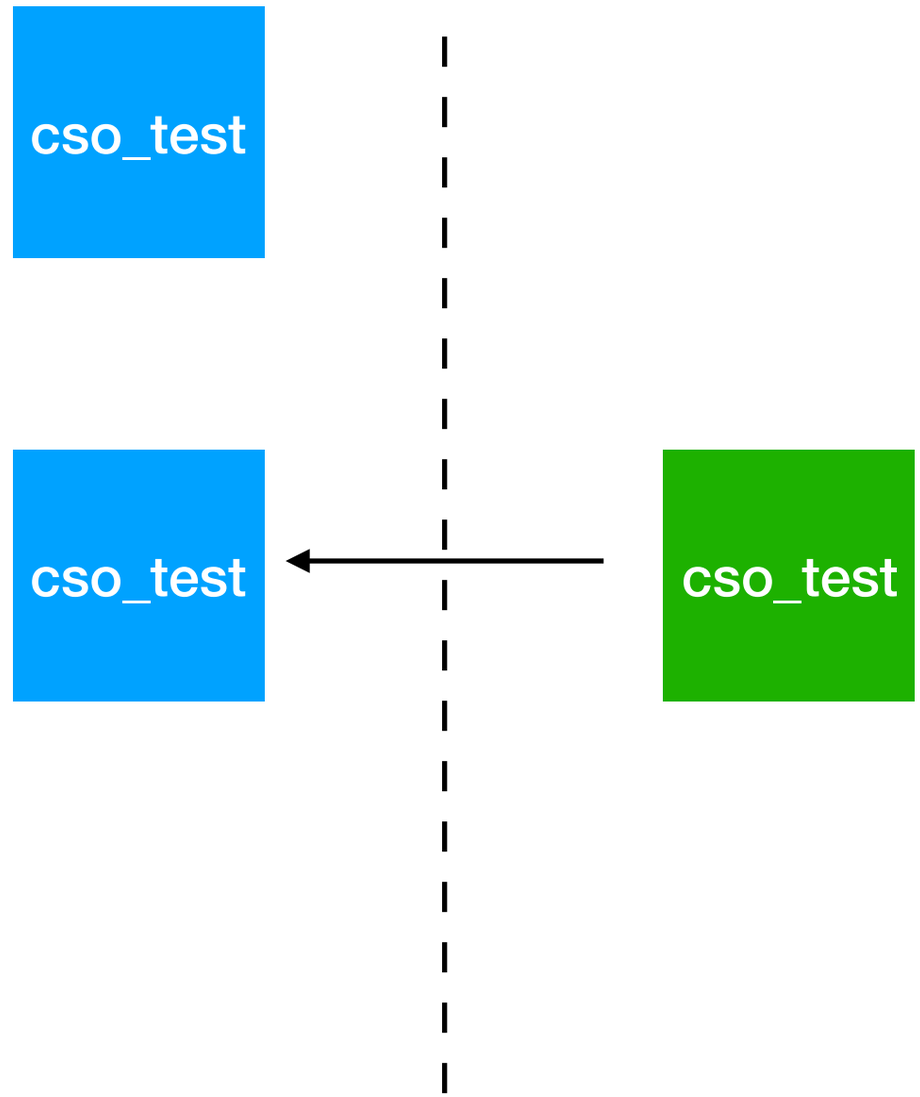
Distributed version control system

- What?
  - There is no “server”. Every client owns a complete repository locally (local repository) and can sync(push/pull) with any other remote repositories.
- Why?
  - There are hundreds or more projects and thousands or more developers in Linux community.
    - Coordinating the development using one single server is difficult.
  - Can work without network.

Git — initialization

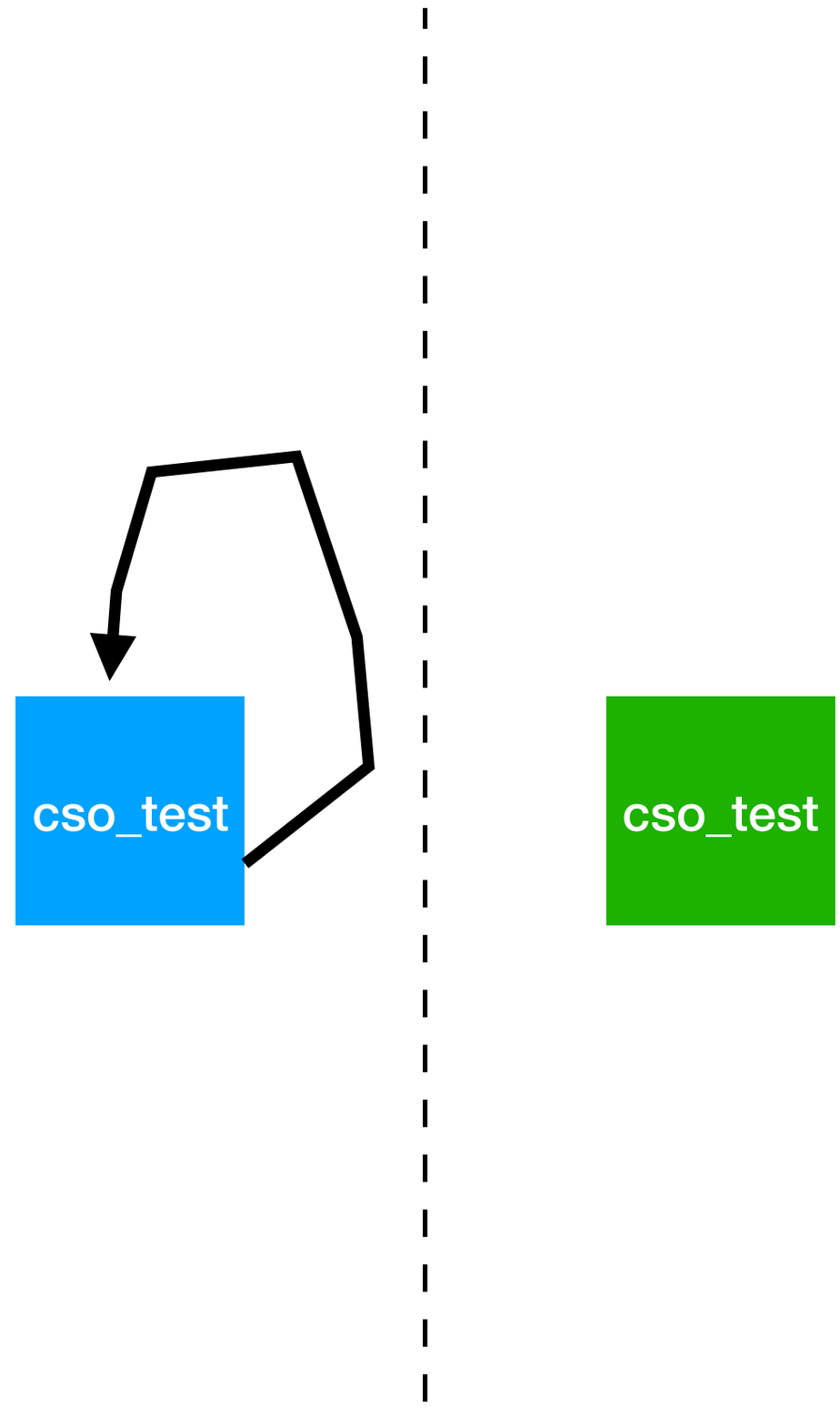
- git init

- git clone
  - git remote -v



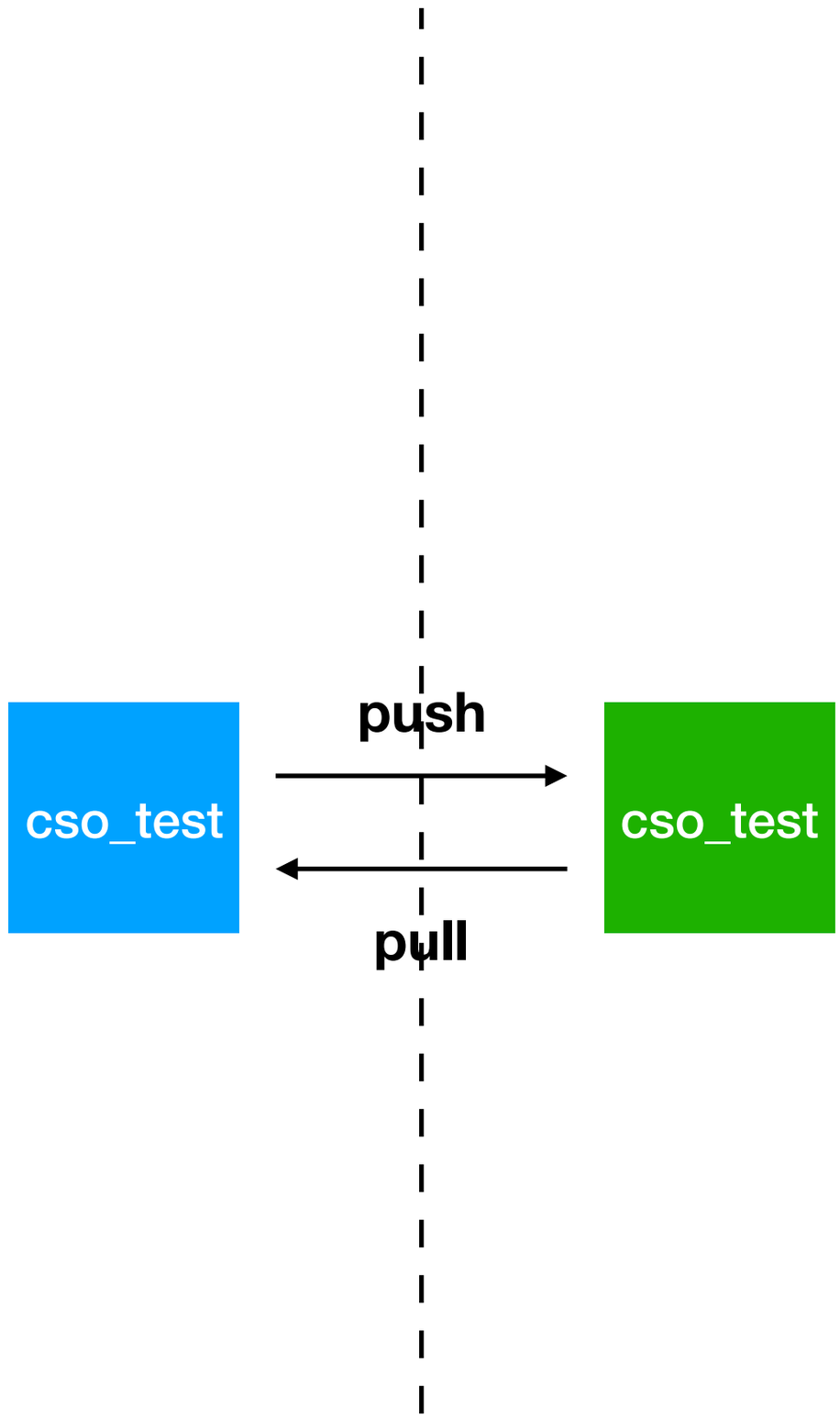
Git — commit

- git commit -m “comment”
- git add FILES
- git rm FILES
- git log



Git — remote

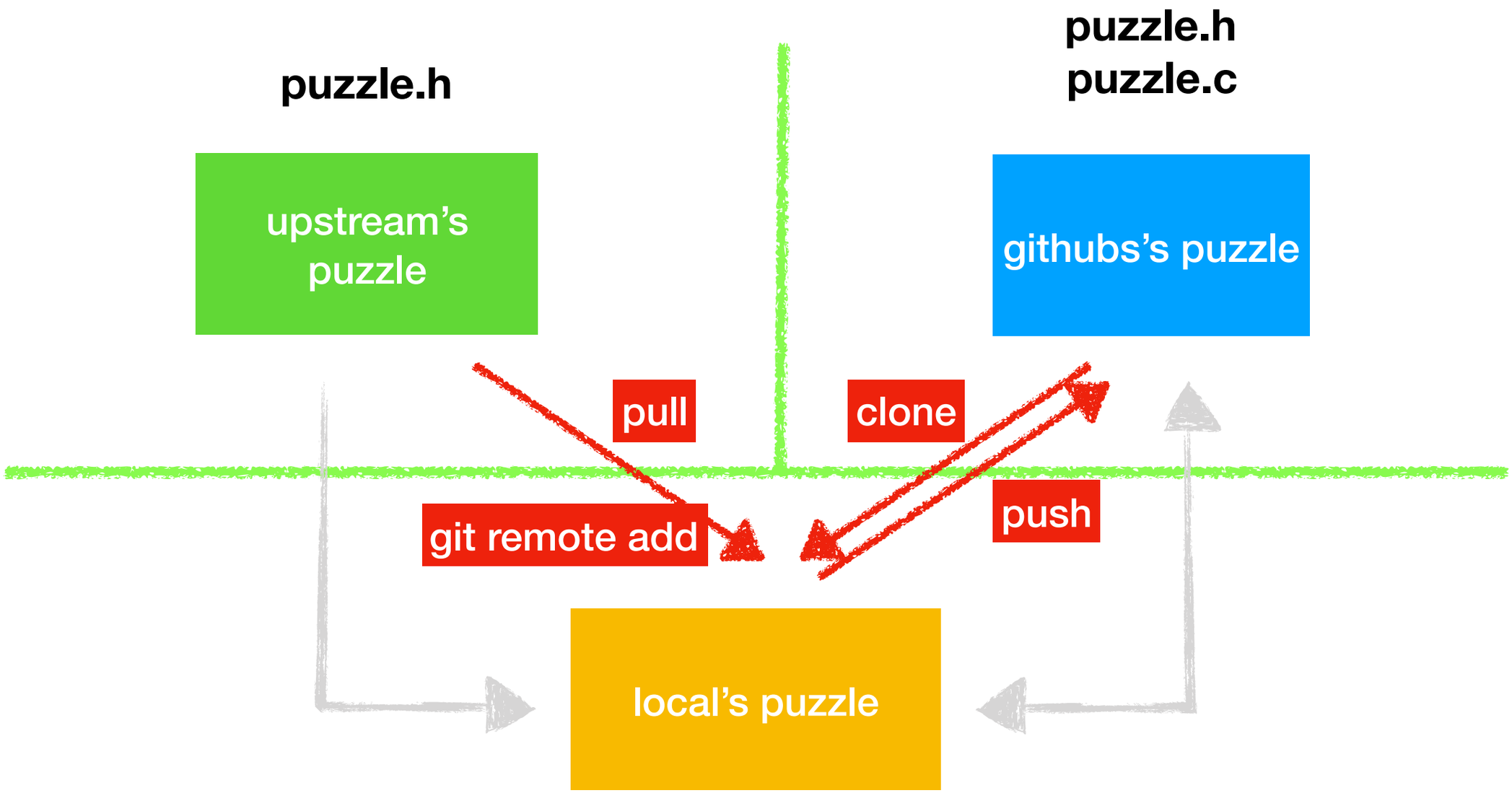
- git remote -v
- git push
  - git push origin master
- git pull
  - git pull origin master



Git — checkout/branch

- git checkout -b BRANCH_NAME
- git checkout HASH
  - git log
- git branch -a

Git status for our recitation and labs



puzzle.h

puzzle.c