Computer Systems Organization Recitation

CSCI-UA 0201-007

R01: Introduction & Environment Setup

Many slides are based on John Westhoff's Fall 2019 CSO recitation

Logistics

Important things you should know

What is this recitation for?

- Help you better understand the course contents, including but not limited to:
    - Reinforce this week’s lecture content
    - Review previous week’s assessment
    - Some exercises meant to help with the labs/assessments
- Make us all suffer by making us get up early

Where we release course materials

- Course website
  - https://nyu-cso.github.io/
  - Recitation slides also on the course schedule page
- NYU Brightspace
  - Zoom recitation links/recording
- CampusWire
  - It’s your responsibility to read Instructor’s Note on Campuswire
  - You are encouraged to ask questions on Campuswire
- GitHub
  - All labs are released on GitHub
  - You will submit all labs on both GitHub and Gradescope
- Gradescope
  - Weekly mini-quiz on Gradescope

How to contact us

- Don't be afraid to ask questions!
- If you have general questions about course contents or labs
  - Ask on Campuswire
  - Come to office hours
  - Register the in-person recitation

  $\left.\begin{array}{l}\text{Come to office hours}\\\text{Register the in-person recitation}\end{array}\right\}$ If you want more personal question answering
- If you want to send us a private message
  - Email cso-staff mailing list at cso-staff@cs.nyu.edu

How are we going to proceed?

- For the first two weeks, we will focus on lab setups, usage of basic tools, etc.
  - Today we will cover lab related setups
  - Next recitation will cover programming tools
- From next week
  - We will also review weekly assessment, reinforce some course contents, exercises to prepare for your labs
- Weekly assessments will be due Friday 9pm EST
  - Done on Gradescope, do it early
  - No late submission

Academic Integrity

- All work must be your own – do not copy or even look at assignments done by others
  - Don’t ask StackOverflow or Chegg for help - if you need it, ask us!
  - Don’t hire someone to do your work for you
- We reserve the right to use software plagiarism detection tools such as Moss
- It’s not worth the risk, just don’t cheat and make us sad
- More details: https://nyu-cso.github.io/policy.html

Getting Started

Important things you must do

Today’s Topics

- Log into snappy1
- Setting up your git repositories
- Basic Unix commands
- Program development
  - Editor (Nano)
  - Version control (Git)

Today’s Goal

- By the end of today’s recitation, you should
  - Be able to log into snappy1
  - Have GitHub ready for you to submit work
    - An account
    - Lab-1 repository
    - Know how to submit assignments

Log into snappy1

- Follow https://nyu-cso.github.io/labs/ instructions
  - Separate instructions for Mac and Windows
- Forget password?
  - https://cims.nyu.edu/webapps/password
- Demo

Log into snappy1

Attention: You MUST test your code on snappy1

- We recommend you to do your labs on snappy1 and test it before submission
- More tools are available for debugging (gdb etc.)
- Gradescope runs the same test script
    - In general, there should be no surprises
- If you choose to do your labs outside of snappy1, we will not provide any technical support should you encounter any OS-related issues in doing the labs

I’m in snappy1, now what?

Basic Commands

- Some useful commands to know:
  - man
  - ls, cd, pwd, mkdir
  - cp, mv, rm
  - echo, cat
  - wc
  - grep
  - ctrl-c,ctrl-d, ctrl-z, fg, bg
  - |, >, <, >>
  - apt install/search
  - history, ctrl-r

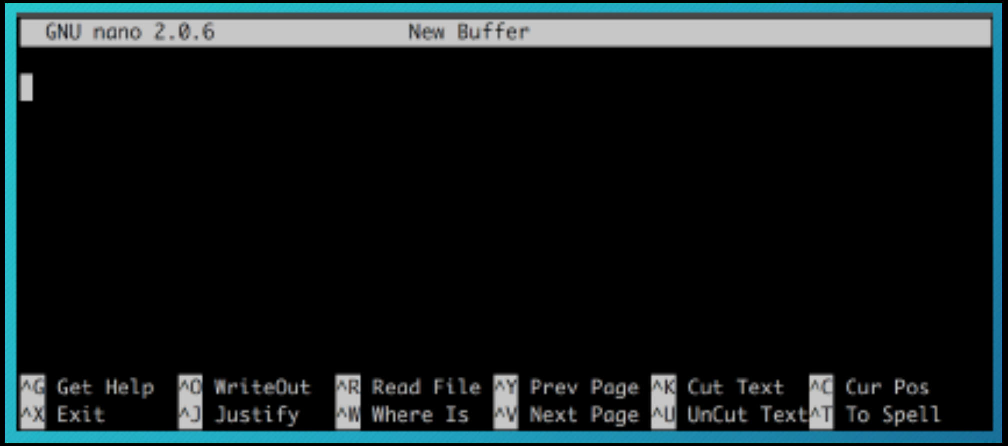
Basic Commands

- Whenever you want to find out how to do something using command line, ask Google first
- Here are some useful links:
    - A one-pager: https://nyu-cso.github.io/labs/linux_cheat_sheet.pdf
    - A more descriptive source: https://github.com/jlevy/the-art-of-command-line

Editor

- You need a good editor to code with for productivity
- Popular editors used by programmers:
  - vim
  - emacs
  - vscode
  - sublime
  - nano
- We recommend you use nano
  - Unfortunately, vscode and sublime are not installed on snappy1
  - No self-respecting programmers use nano, but you can get by with nano in CSO

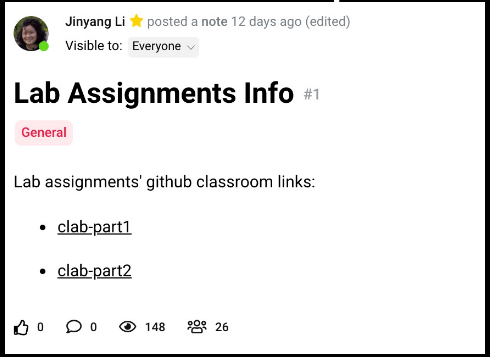
Brief Intro of Nano Editor


```
  GNU nano 2.0.6                  New Buffer





^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

- To create/edit a file
  - nano <Path to (Save) the File>
  - Edit
  - Hit Ctrl+O to save your changes (^ means Ctrl)
  - Hit Ctrl+X to exit

Setup GitHub/lab1 repo

- Create a GitHub account if you don't have one
- We have created for you a corresponding private lab repository on Github.com
- Enroll yourself in the GitHub classroom
    - Create your lab-1 repository by following the link posted on Campuswire
        - Select your NYU NetID
            - Very important!
        - Don't select someone else's NetID!
- If you cannot find your NetID, let me know!
- Setup GitHub **ssh-based logins:** https://nyu-cso.github.io/labs/#repo


Jinyang Li posted a note 12 days ago (edited)
Visible to: Everyone

Lab Assignments Info #1

General

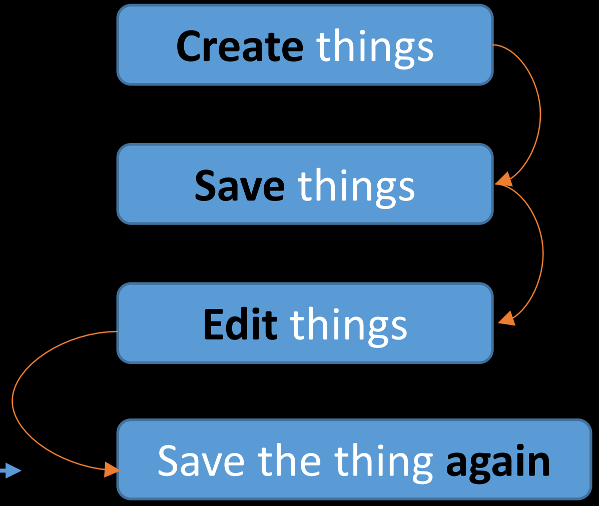
Lab assignments' github classroom links:

- clab-part1
- clab-part2

0 0 148 26

Setup GitHub/lab1 repo

Git Overview



- Distributed version control system
- What is version control?
  - Manages *changes* to documents, source files and other collections of information
- Why is version control indispensable?
  - History tracking: track code changes
  - Roll back to older version
  - Collaborate with others (*collaborative history tracking*)
- We are going to use the popular "Git" as our version control system

You need to config git first!

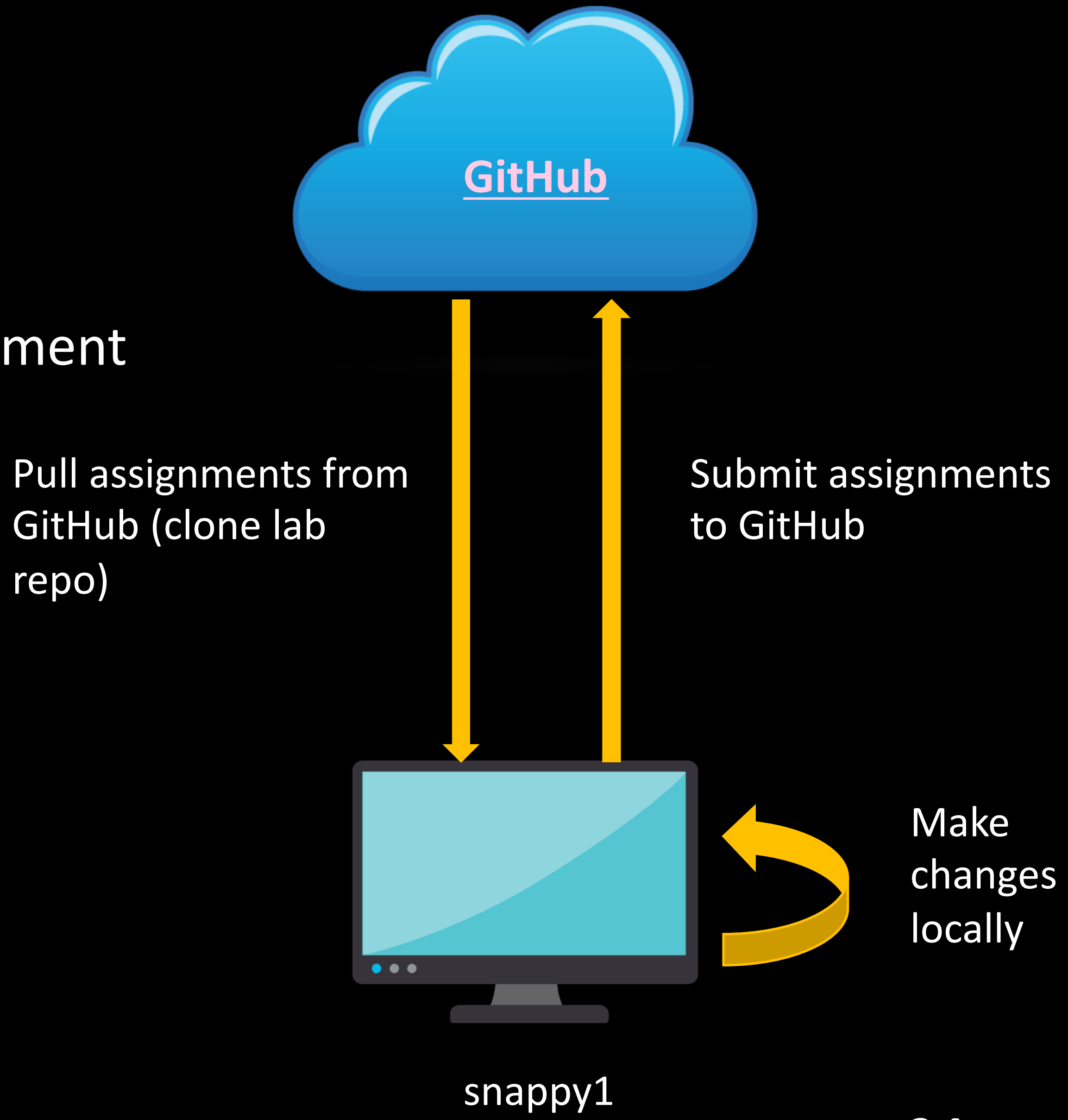
- git config --global user.email “<Your Email>”
- git config --global user.name “<Your Name>”
- You can issue “git config --list” to check your configuration
- Here, the <Your Email> should be the one associated with your GitHub account

A list of git commands you need

- git clone
- git status
- git remote
- git add <file name>
- git commit -m <commit messages>
- git push origin master
- git pull upstream master

Git Overview

- GitHub:
  - provides hosting for software development and version control using Git.

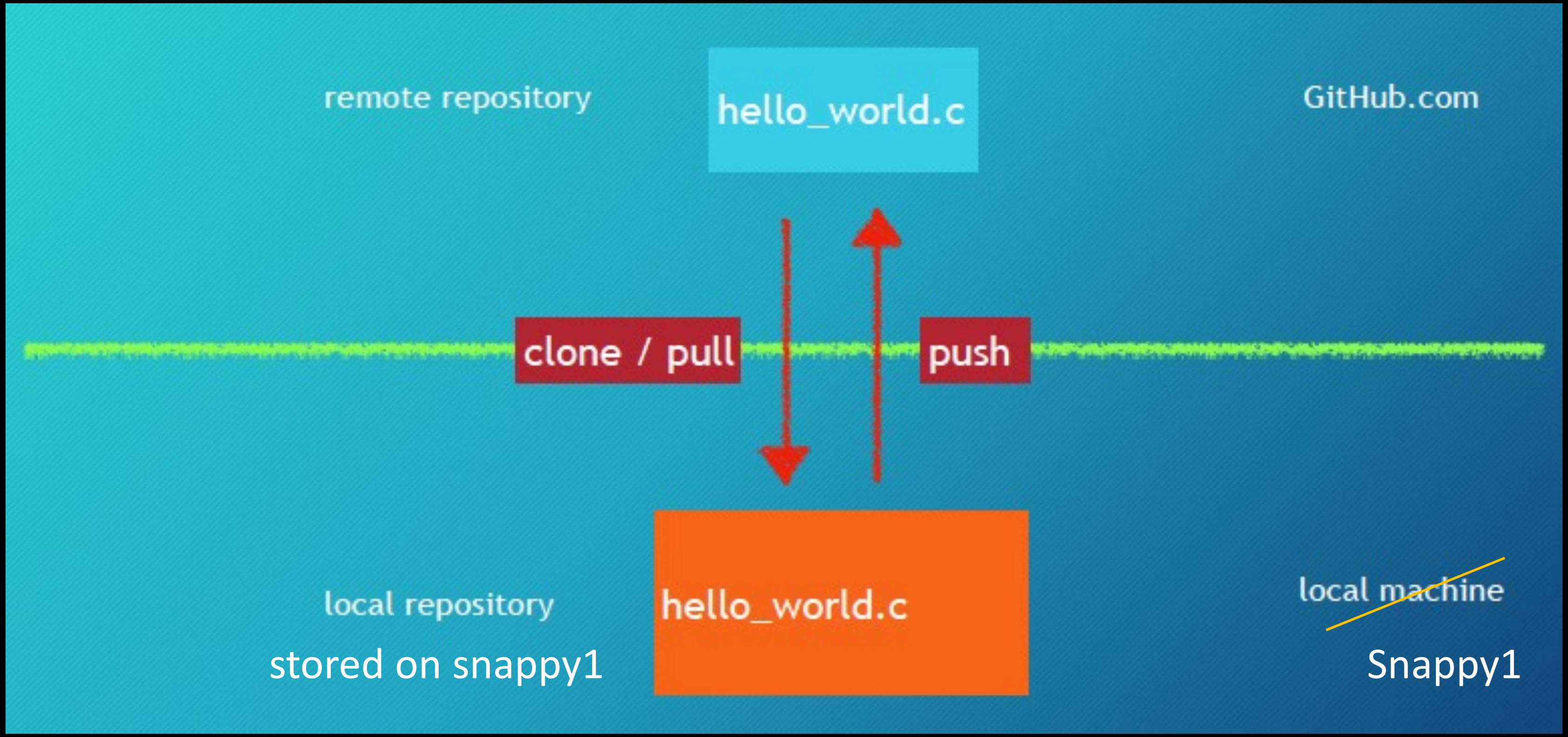

GitHub
Pull assignments from GitHub (clone lab repo)
Submit assignments to GitHub
Make changes locally
snappy1

Clone your lab repo on Snappy

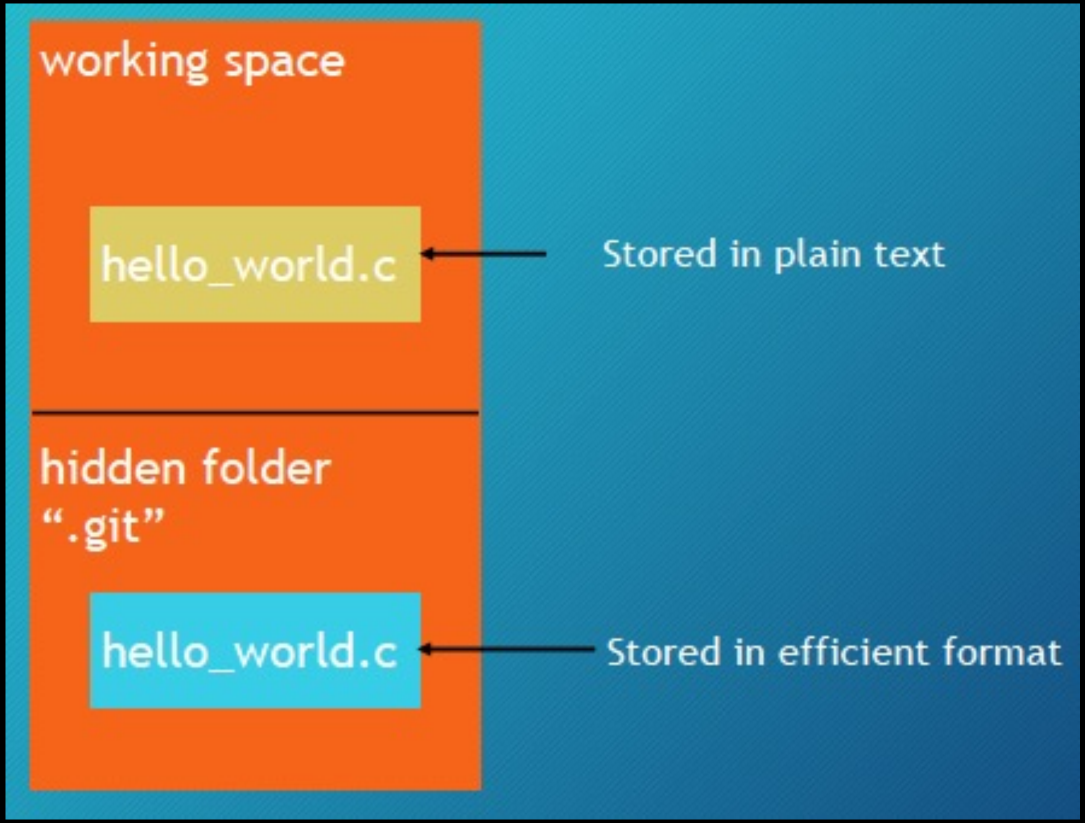
- In command line, type:
  - mkdir cso-labs
  - cd cso-labs
  - git clone git@github.com:nyu-cso-fa21/clab-part1-<Your Github Username>.git clab-part1
    - If you copy the above command to command line, don't let the line break
    - Replace <Your GitHub Username> (including the angle brackets) with your GitHub username.
  - cd clab-part1

Git Setup

The remote copy is stored in some efficient format



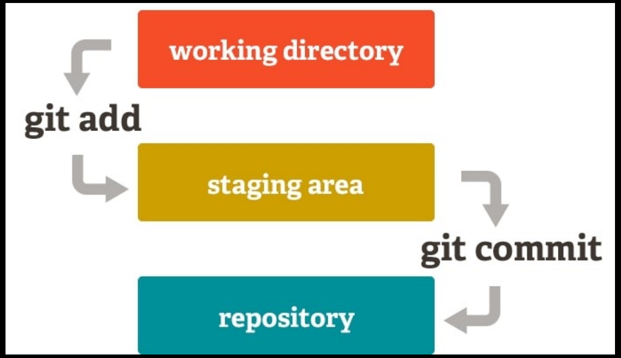
A closer look at your repository stored on snappy1

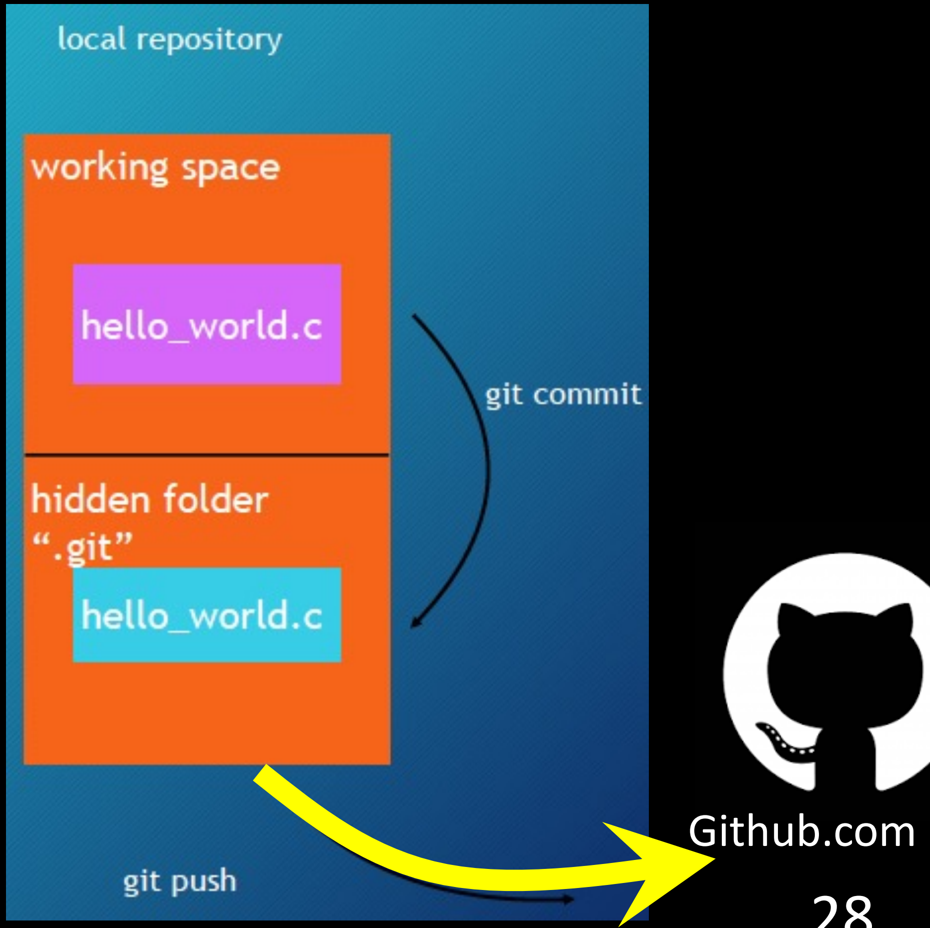


Local repository stored on snappy1

How to interact with Git

- git add hello_world.c
  - Tell git to track changes to hello_world.c
- git commit
  - Store tracked file to .git
- git push origin master
  - Submit commits to your remote repository





Git commit

- When you issue “git commit”, you are required to provide a message which is a short description of the changes you made
- Use "-m"/"--message" option to specify that
- E.g.: git commit -m “my first commit”
- Warning: please remember to add "-m"/"--message"
- If you unfortunately forget:
  - a command line editor will pop up for you to edit the commit message
  - By default, vim, which is not really friendly to beginners (see the short guide in the end)

Double check with “git status”

- Sometimes, you might forget to do some (or all) of
  - git add, git commit, git push
- It’s always good to check the status of your repository
- git status tells you
  - What files are going to commit
  - What files are not tracked
  - Whether you forget to push commits to remote

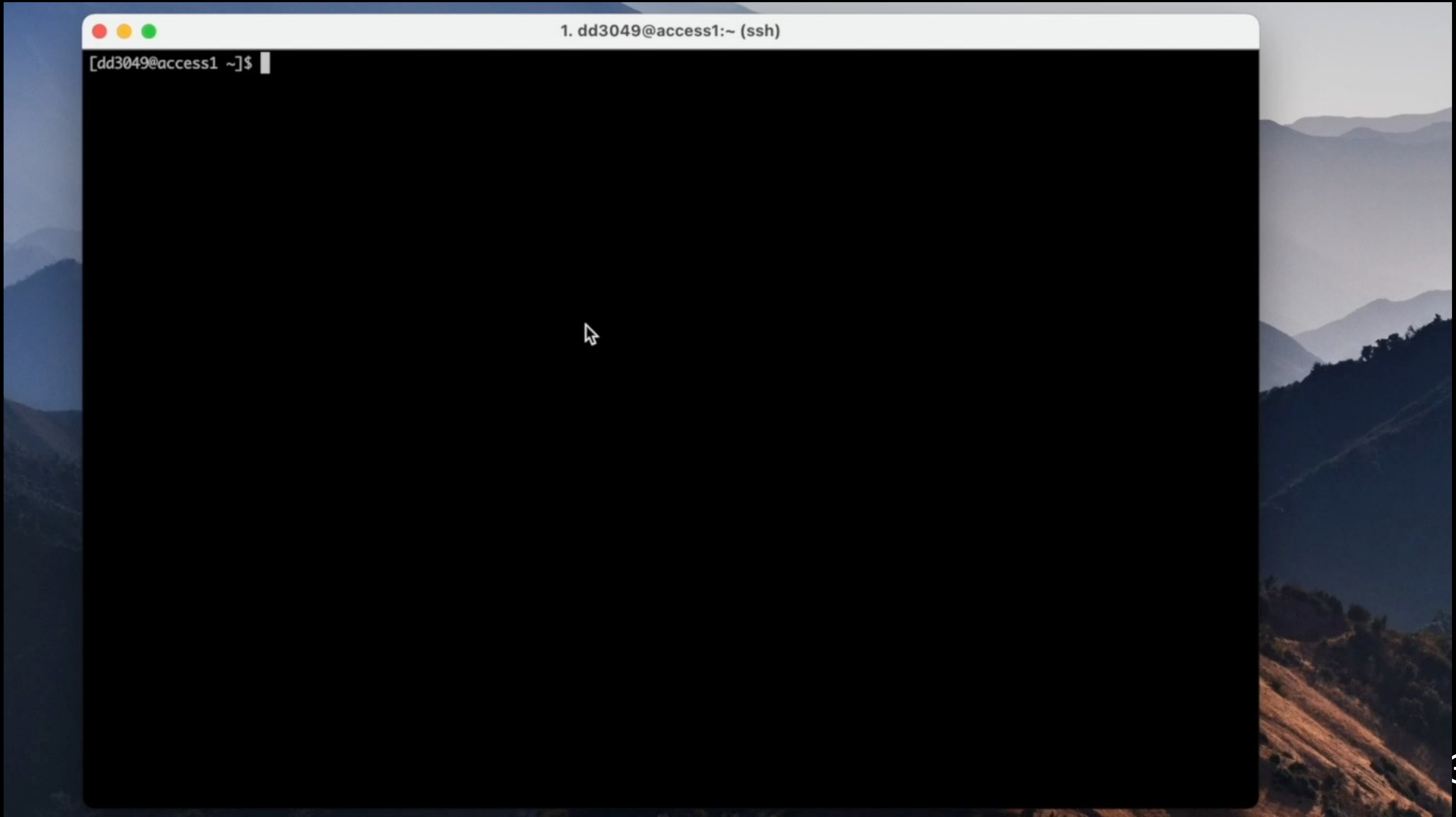
Triple check with GitHub

- Still not sure/confident about whether commits was submitted properly?
- Go to github.com, then go to your repo
- Manually check if every file contains the up-to-date information

For each new assignment

- Create lab repo on GitHub (click link, select yourself) (covered)
- Clone your lab repo to snappy1
  - cd cso-labs
  - git clone git@github.com:nyu-cso-fa21/clab-part1-<YourGithubUsername>.git clab-part1
- Make changes accordingly
  - Tell git to track changes: git add <Changed/Newly-Created Filenames>
- Commit your changes
  - git commit -m “commit messages”
- Push to your remote repository (on GitHub)
  - git push origin master

Git push commit demo



```
[dd3049@access1 ~]$
```

All the git commands you need for CSO

- For beginners, it's super easy to mess up Git
  - Notably conflicts, typically seen for collaboration, but not for individual usage
- To avoid conflicts, only clone the repo once for each lab
- After setting lab repository, you ONLY need to use the following git commands:
  - git add filenames
  - git commit -m "commit message"
  - git push origin master
  - git clone your-lab-repo lab
  - git status

Warning: unless you know what you are doing, do not use any other git commands or git command flags

Ask the staff for help

- If you really cannot fix conflicts or other git problems, you should ask course staff for help
  - You need to email the staff or attend office hours
  - You should start your lab earlier
- Don’t randomly issue commands to further mess things up

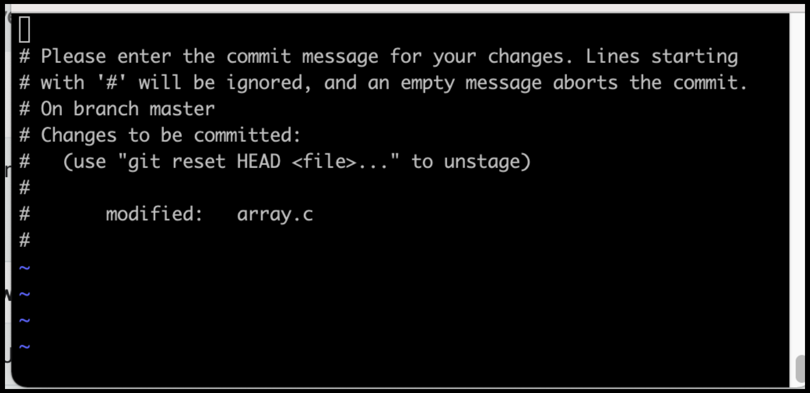
Things you should NEVER do

- Don’t use git add *, git add .
  - Instead, you should always specify the file names you want to commit
  - Please don’t add complied programs to git
- Don’t modify any file using GitHub website
  - Instead, you should always make changes on snappy1 and then push commits to GitHub
  - Otherwise, there will be conflicts, which will lead to sadness

Git is much powerful than that

- Our git introduction only covers a small part of Git
- Git tutorial:
  - https://www.atlassian.com/git/tutorials/what-is-versioncontrol
  - https://try.github.io/levels/1/challenges/1

How to commit with Vim Editor (Optional)



```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   array.c
#
~
~
~
~
```

- The default editor is called Vim
- To add a commit message from vim
  - First hit Esc (make sure you're in Normal mode)
  - Then hit "i" (entering Insert mode)
  - Then type in some commit message
  - Hit Esc (exiting Insert mode, going back to Normal mode)
  - Then type in ":wq" and press Enter (exit vim)
- If you are lost, hit Esc and start from the beginning