

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 stay at school late

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**
- If you want to send us a private message
 - Email cso-staff mailing list at cso-staff@cs.nyu.edu

} If you want more personal question answering

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 setup related setups
 - Next recitation will cover programming tools
- From next week
 - We will 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

Getting Started

Important things you must do

Today's Topics

- Log into snappy1
- Setting up your git repositories
- Basic Unix commands
- Program development
 - Editor (Sublime)
 - 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

Basic virtual machine setup

- Follow <https://nyu-cso.github.io/labs/> instructions
 - Separate instructions for Mac and Windows
- Demo

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
 - A more descriptive source: <https://github.com/jlevy/the-art-of-command-line>

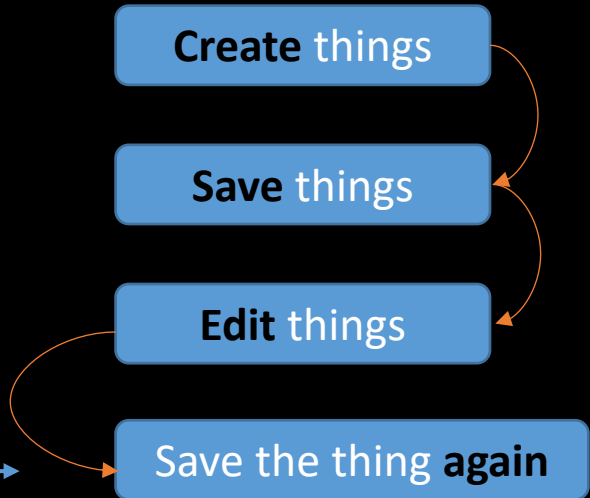

Editor

- You need a good editor to code with for productivity
- Popular editors used by programmers:
 - vim
 - emacs
 - vscode
 - sublime
- 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

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!

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!

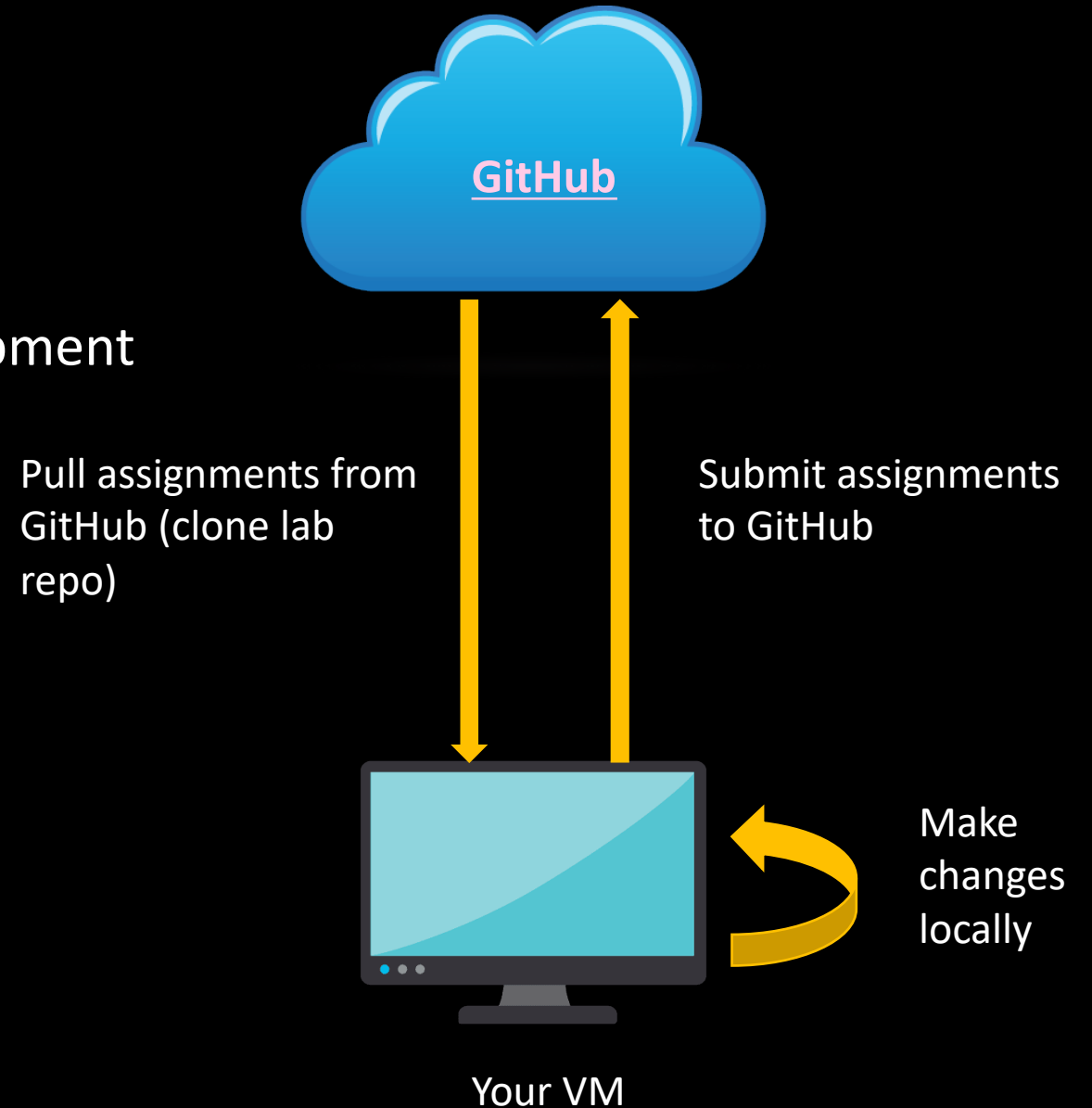
- `git config --global user.email "<Your Email>"`
- `git config --global user.name "<Your Name>"`
- You can issue `"git config --list"` to check your configuration
- Here, the `<Your Email>` should be the one associated with your GitHub account

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.



Clone your lab repo locally

- In command line, type:

- `mkdir cso-labs`

- `cd cso-labs`

- `git clone https://github.com/nyu-cso-fa21/clab-part1-
<YourGithubUsername>.git lab1`

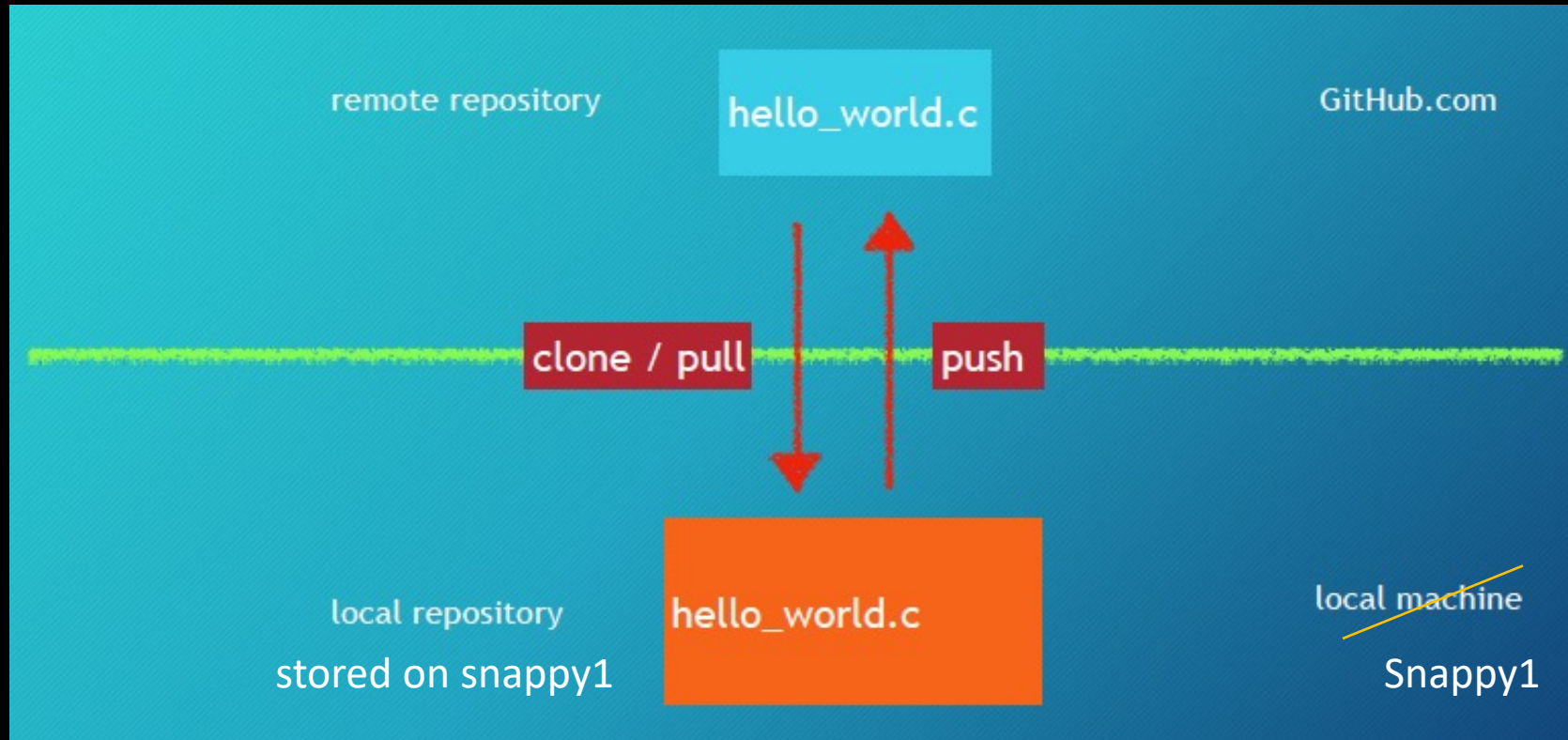
- If you copy the above command to command line, don't let the line break

- Replace <Your GitHub Username> (**including the angle brackets**) with your GitHub username.

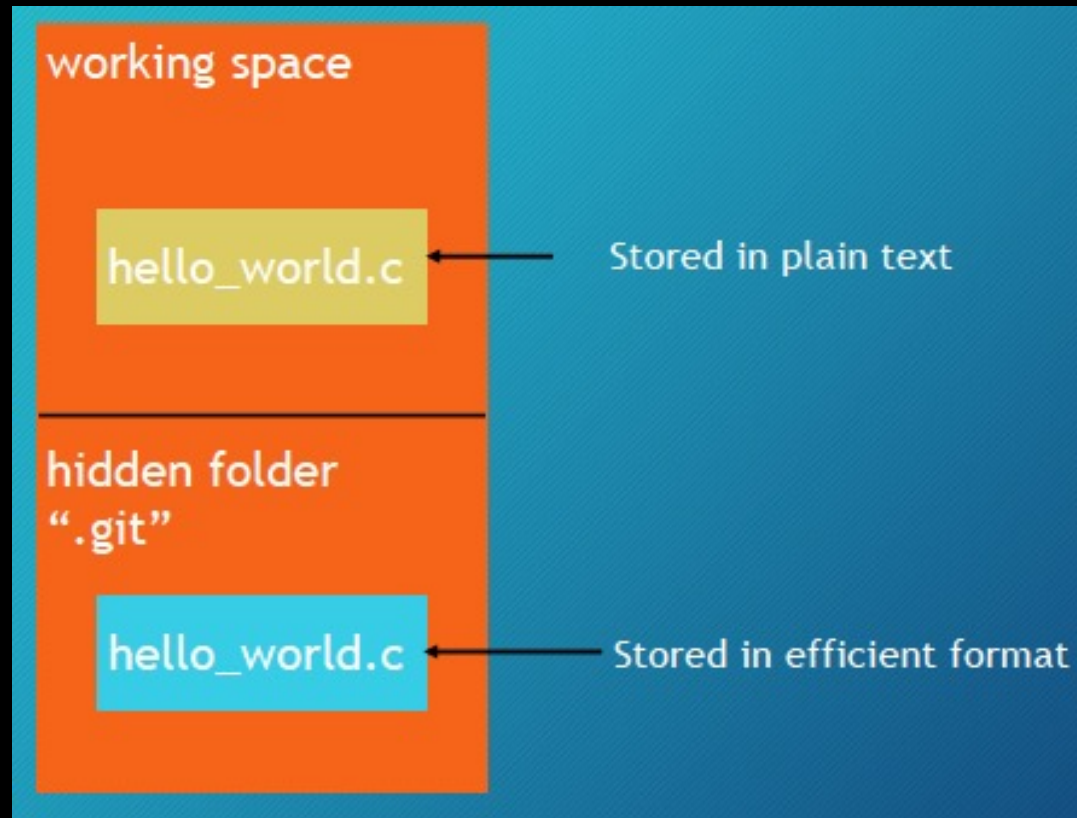
- `cd lab1`

Git Setup

The remote copy is stored in
some efficient format



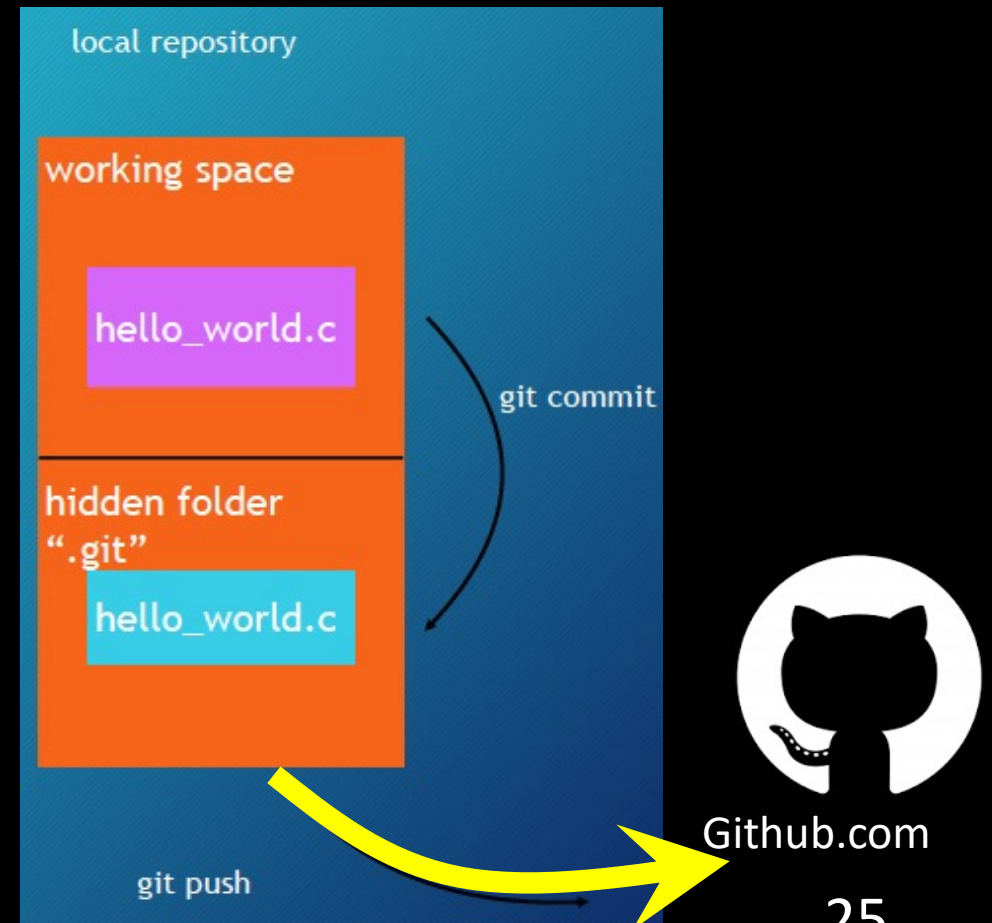
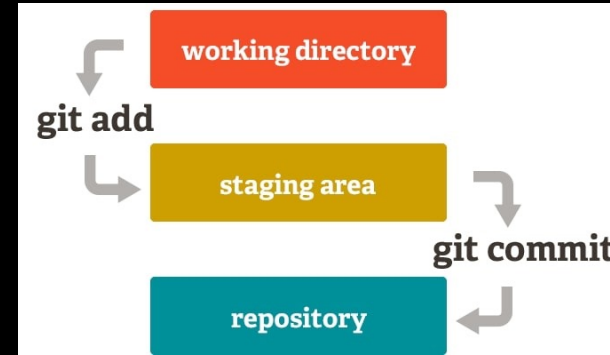
A closer look at your repository stored on snappy1



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



For each new assignment

- Create lab repo on GitHub (click link, select yourself)
- Clone your lab repo to snappy1
 - `cd cso-labs`
 - `git clone https://github.com/nyu-cso-fa21/clab-part1-<YourGithubUsername>.git lab1`
- Tell git to track changes
 - `git add filenames`
- Commit your changes
 - `git commit -m "commit messages"`
- Push to your remote repository (on GitHub)
 - `git push origin master`

Git commit

- When you issue “git commit”, you need to provide a message which is a short description of the changes you made
- You can use “-m” option to provide the commit message
 - E.g.: git commit -m “my first commit”
- If you don’t use “-m” option, a command line editor will pop up for you to edit the commit message
 - By default, nano

How to get out of Nano Editor

- The default editor is called Nano



- To add a commit message from nano
 - First type in some commit message
 - Hit Ctrl+O to save your commit message (^ means Ctrl)
 - Hit Ctrl+X to exit

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 assignment was submitted properly?
- Go to github.com, then go to your repo
- Manually check if every file contains the up-to-date information

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>

All the git commands you need for CSO

- For beginners, it's super easy to mess up Git
- 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
 - Online makes things harder..
 - 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 locally on your laptop and then push commits to GitHub
 - Otherwise, there will be conflicts, which will lead to sadness