rs01 agenda

1. Intro
2. Logistics
3. Motivation
4. Lab infrastructure
    a. Git/GitHub
    b. Docker
    c. Scripts and Makefiles
5. Lab 1 Overview
    a. C basics
    b. gdb
6. Q&A

git != GitHub

git: version control software

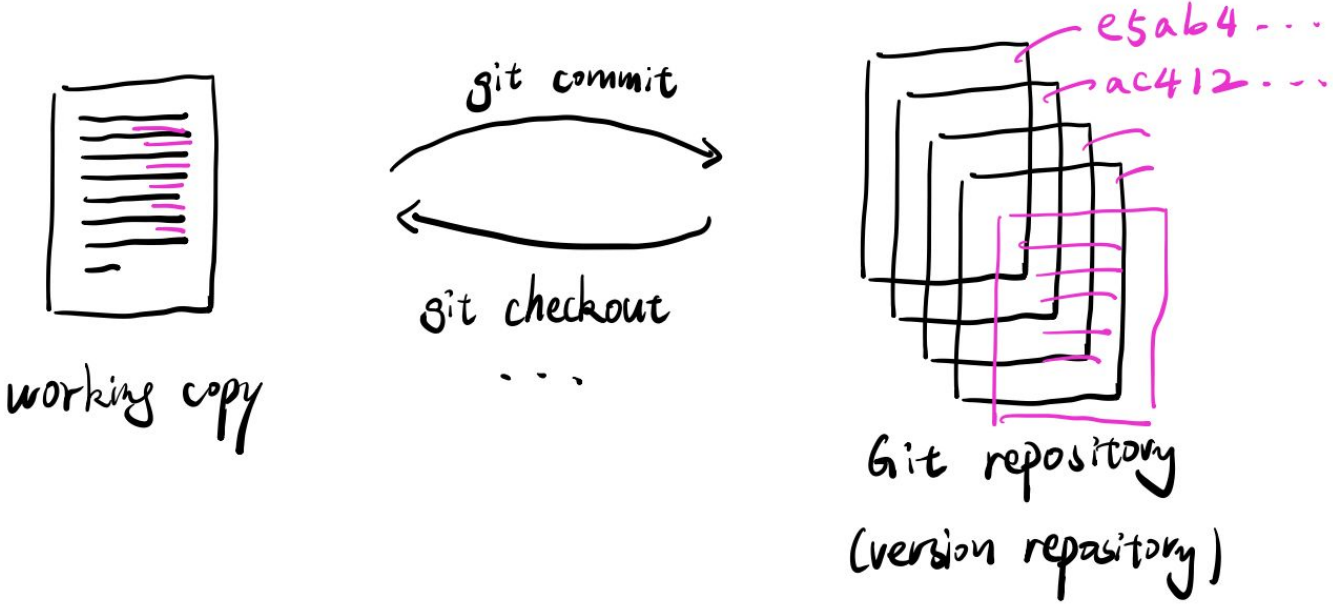
repository: “container” which holds files, tracks history of changes

working copy: current state of files in your repository

remote: repositories stored elsewhere (e.g. GitHub)

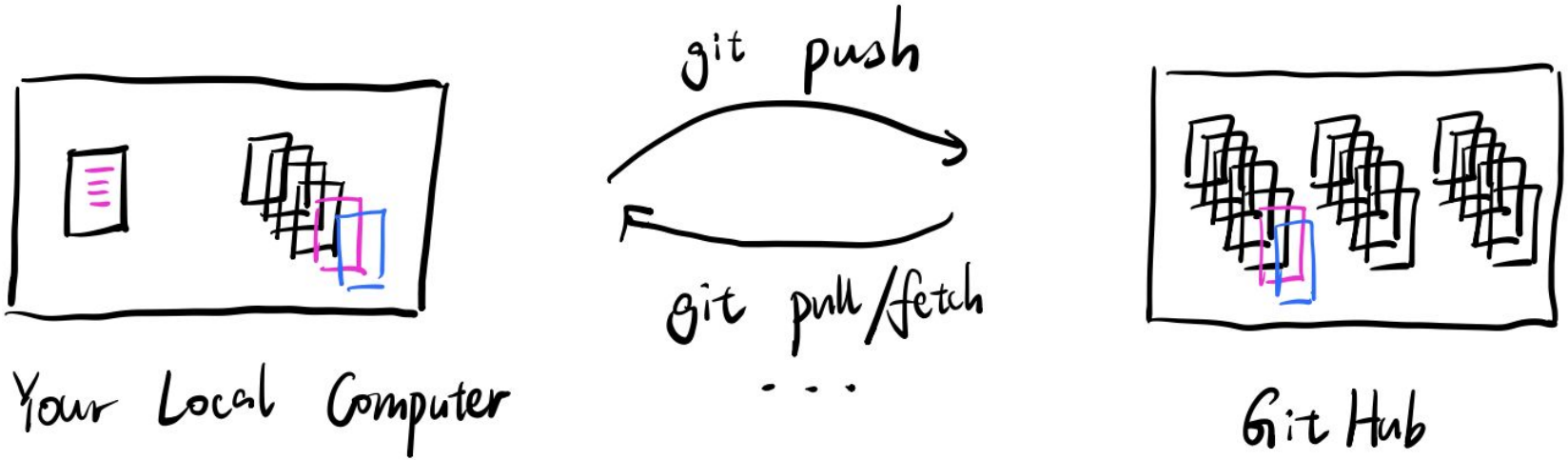
clone: copy a repository (usually from remote)

GitHub: website that stores git repositories (see alternatives: GitLab, Bitbucket, etc.)



commit: save your changes to the git version history

checkout: switch working copy to a current version

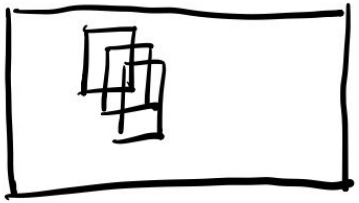


push: update remote repo with local commits

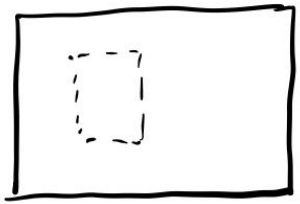
fetch: copy commits from remote into local repo

pull: fetch AND update your working copy to reflect new commits

Lab Git Workflow



nyu-cs/labs
(upstream)

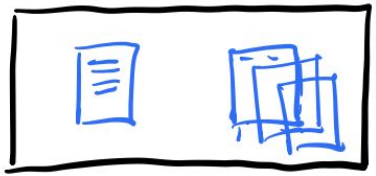


nyu-cs/labs-23sp-<username>
(origin)

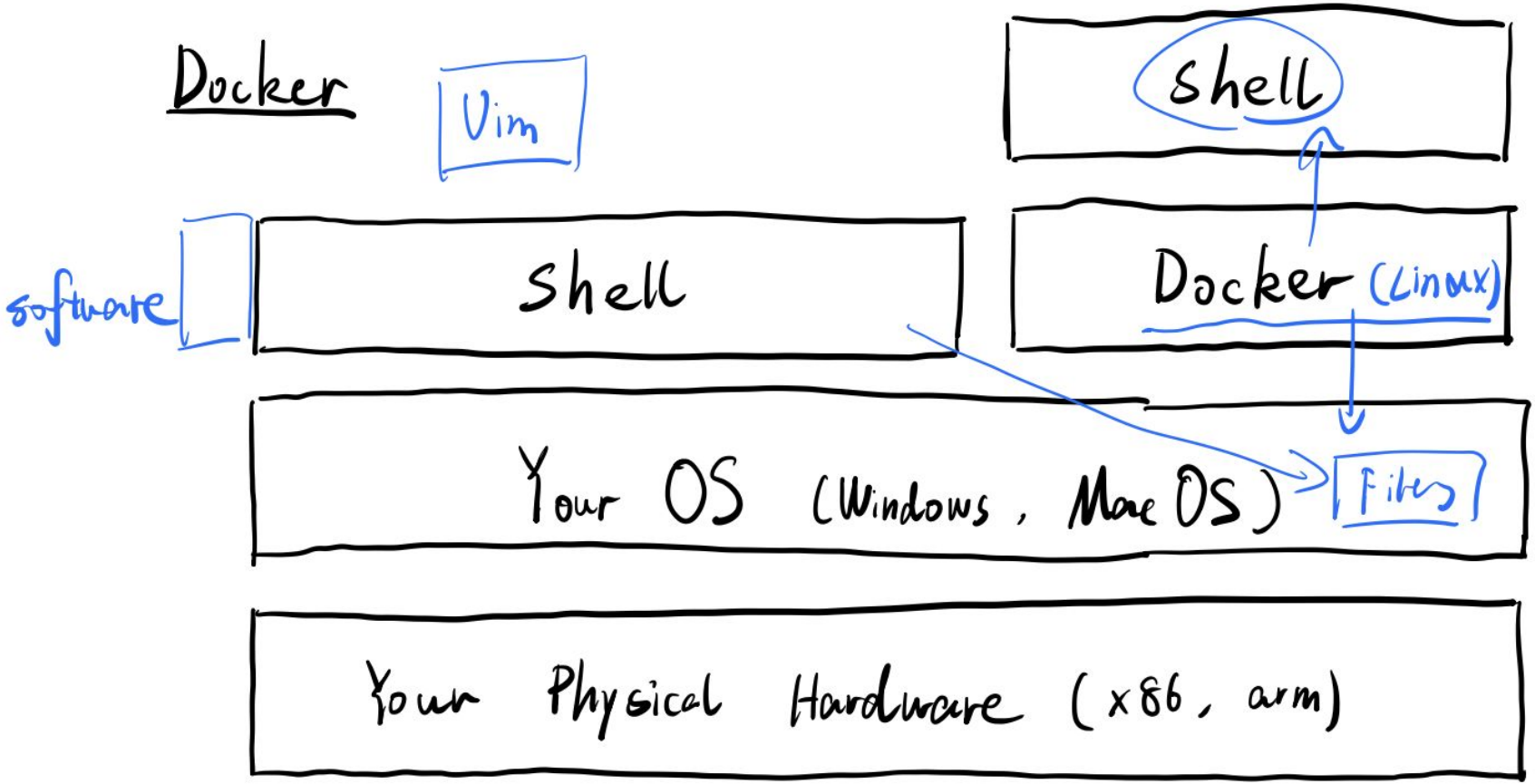
① git clone

② git fetch

git push



Your Local Computer


Docker
Vim
shell
software
Shell
Docker (Linux)
Your OS (Windows, Mac OS)
Files
Your Physical Hardware (x86, arm)

Makefile

Syntax:

target: dependencies

    recipe

Ex:

$ make hello

Makefile:

```
hello: hello.c
    gcc -Wall -g hello.c -o hello
```

C Basics

Declare a variable:

int x;

boolean b;

char * c;

Don’t forget to initialize!

x = 6;

int y = 3;

C Basics

Pointers:

“Point” to an address in memory

Syntax: type * name

Dereference with *

Get address of a variable with &

Examples:

```
int * int_p;

* int_p = 6;

int x = *int_p; // x =6

x = 3;

int_p = &x; // *int_p = 3. Note that the address of int_p has also changed!
```

C Basics

Pointers mental model:

```
int a = 1;

int* b = &a;

int** c = &b;
```

```
var     data        addr
 a   [     1     ]   0x100
 b   [ 0x100 ]   0x108
 c   [ 0x108 ]   0x116
```

a = 1

*b = 1

*c = 0x100

**c = 1

C Basics

Strings: no built-in strings in C

Instead: use array of chars

“Null-terminated”: strings end with null character (‘\0’)

E.g.

```
// wrong but will compile sometimes
char name[5] = “Alice”;

// better
char name[6] = “Alice”;

char[ ] name = “Bob”; // mutable
char * name = “Bob”; // immutable
char * name = malloc(4 * sizeof(char));
*name = “Bob\0”; //mutable
```

GDB!! :)

```c
#include <stdio.h>
#include <stdlib.h>

/*
 * Source: Example by Xiangyu Gao
 */

void swap (int x, int y) {
    int tmp = x;
    x = y;
    y = tmp;
}

void swap2 (int* x, int* y) {
    int tmp = *x;
    *x = *y;
    *y = tmp;
}

int main () {
    int a = 0;
    int b = 10;

    swap(a, b);
    printf("result after first swap: a=%d b=%d\n", a, b);


    swap2(&a, &b);
    printf("result after second swap: a=%d b=%d\n", a, b);

    return 0;
}
```