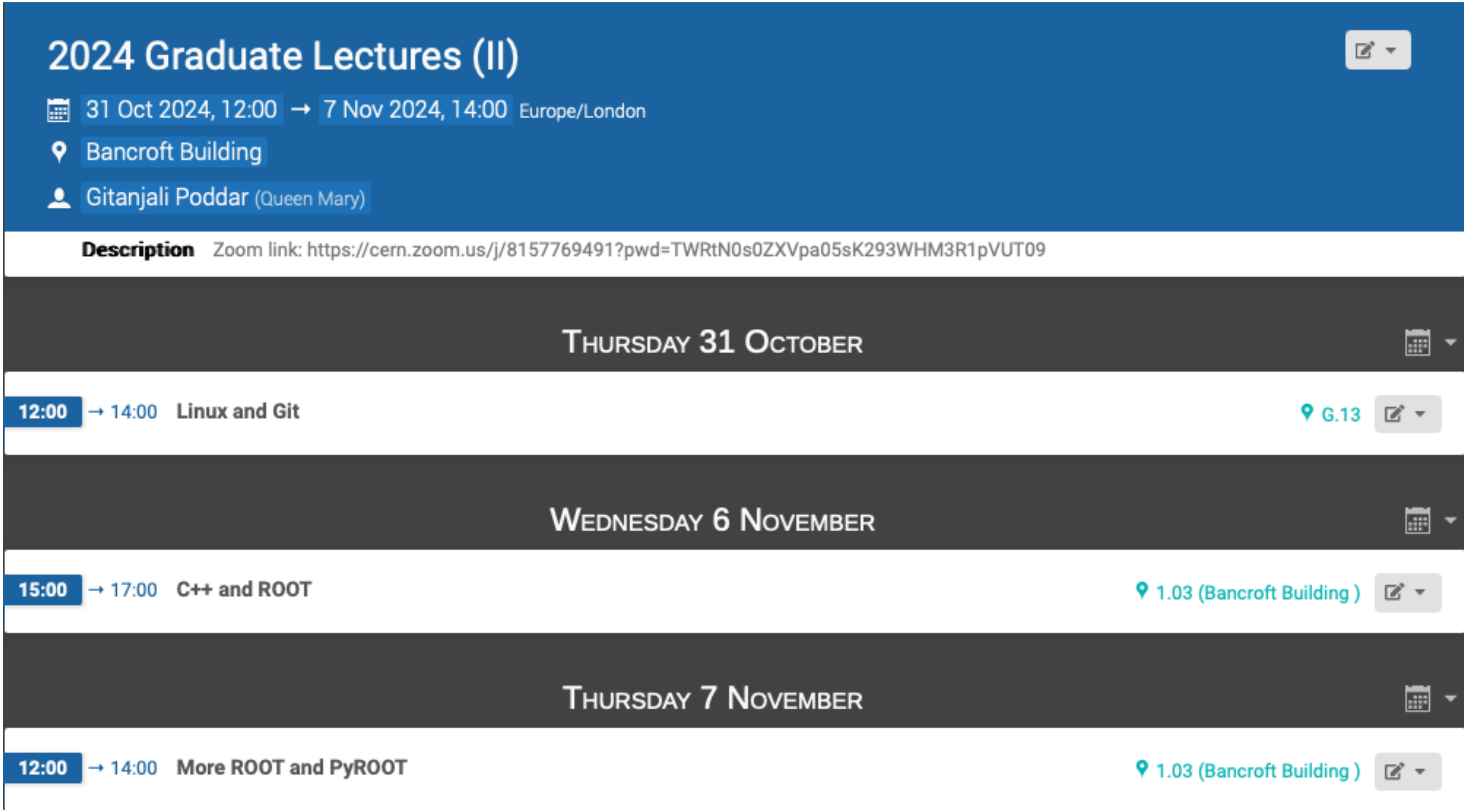
Queen Mary
University of London

Graduate Computing Course

Gitanjali Poddar
31/10/2024

Course Outline

https://indico.ph.qmul.ac.uk/event/2175/



2024 Graduate Lectures (II)

31 Oct 2024, 12:00 → 7 Nov 2024, 14:00 Europe/London

Bancroft Building

Gitanjali Poddar (Queen Mary)

Description Zoom link: https://cern.zoom.us/j/8157769491?pwd=TWRtN0s0ZXVpa05sK293WHM3R1pVUT09

Thursday 31 October

12:00 → 14:00 **Linux and Git** G.13

Wednesday 6 November

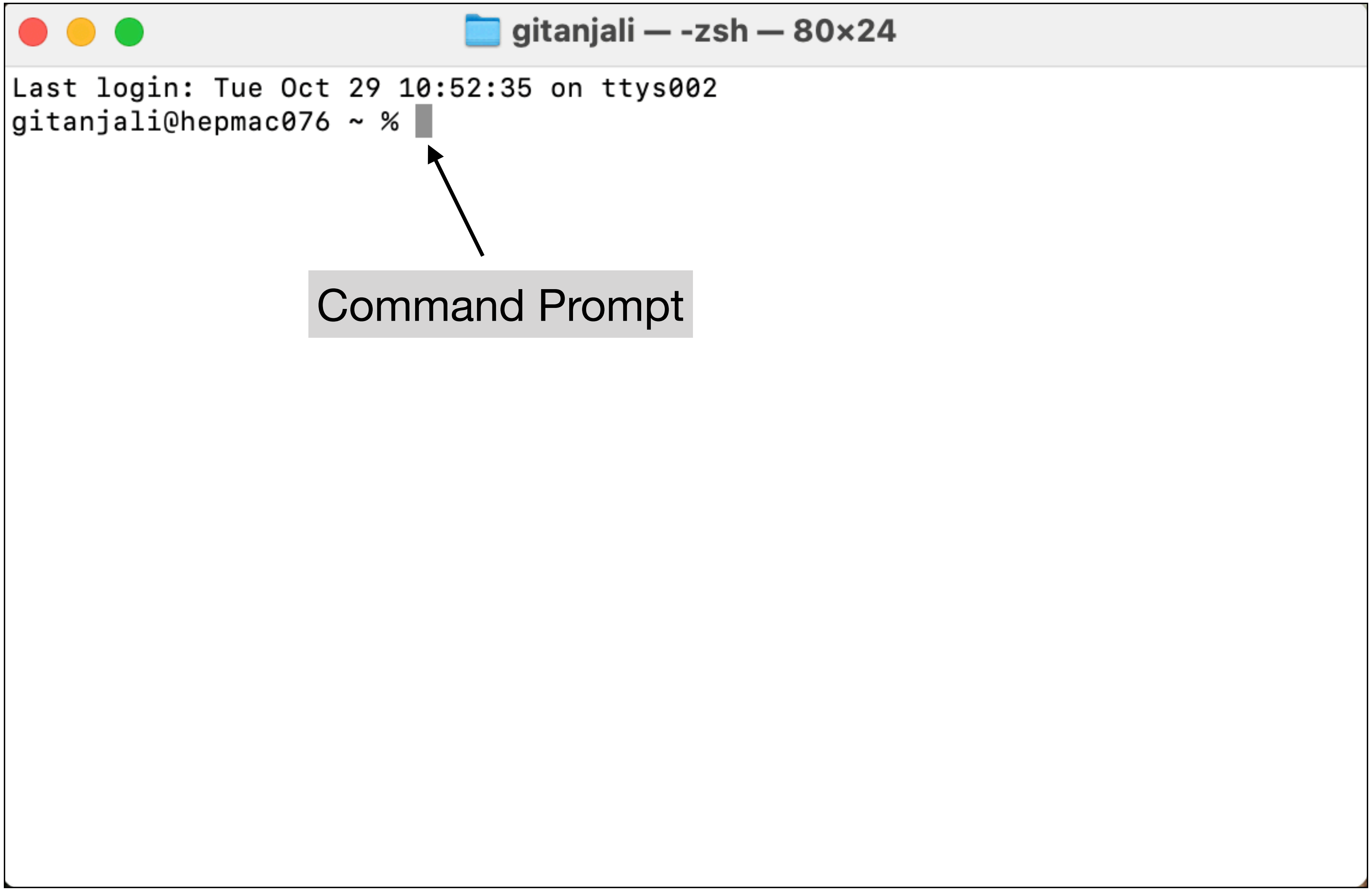
15:00 → 17:00 **C++ and ROOT** 1.03 (Bancroft Building)

Thursday 7 November

12:00 → 14:00 **More ROOT and PyROOT** 1.03 (Bancroft Building)

Part I: Intro to Unix/Linux

Command or Terminal Shell



```
Last login: Tue Oct 29 10:52:35 on ttys002
gitanjali@hepmac076 ~ % 
```

A **shell** is a user interface for access to an operating system's services

Directory Management

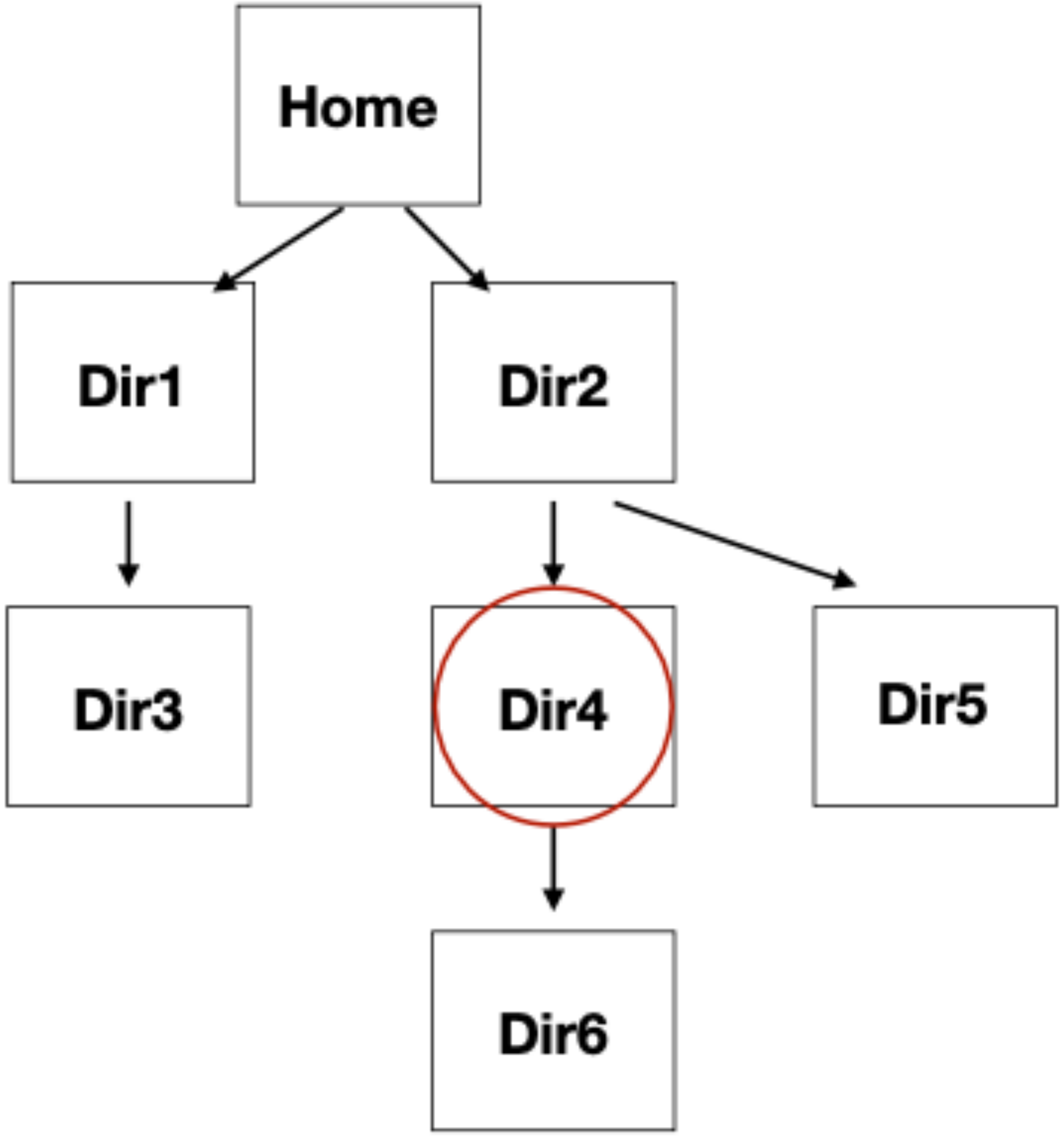
Basic idea of a **path**: home/directory1/directory2/directory3

`/` is the directory separator

- `pwd`: print current directory path
- `ls`: list all contents of current directory
- `mkdir <dir>`: create a directory `<dir>`
- `cd <dir>`: change to directory `<dir>`
- `cd ..`: change to one directory level back
  `cd ../..`: change to two directory levels back

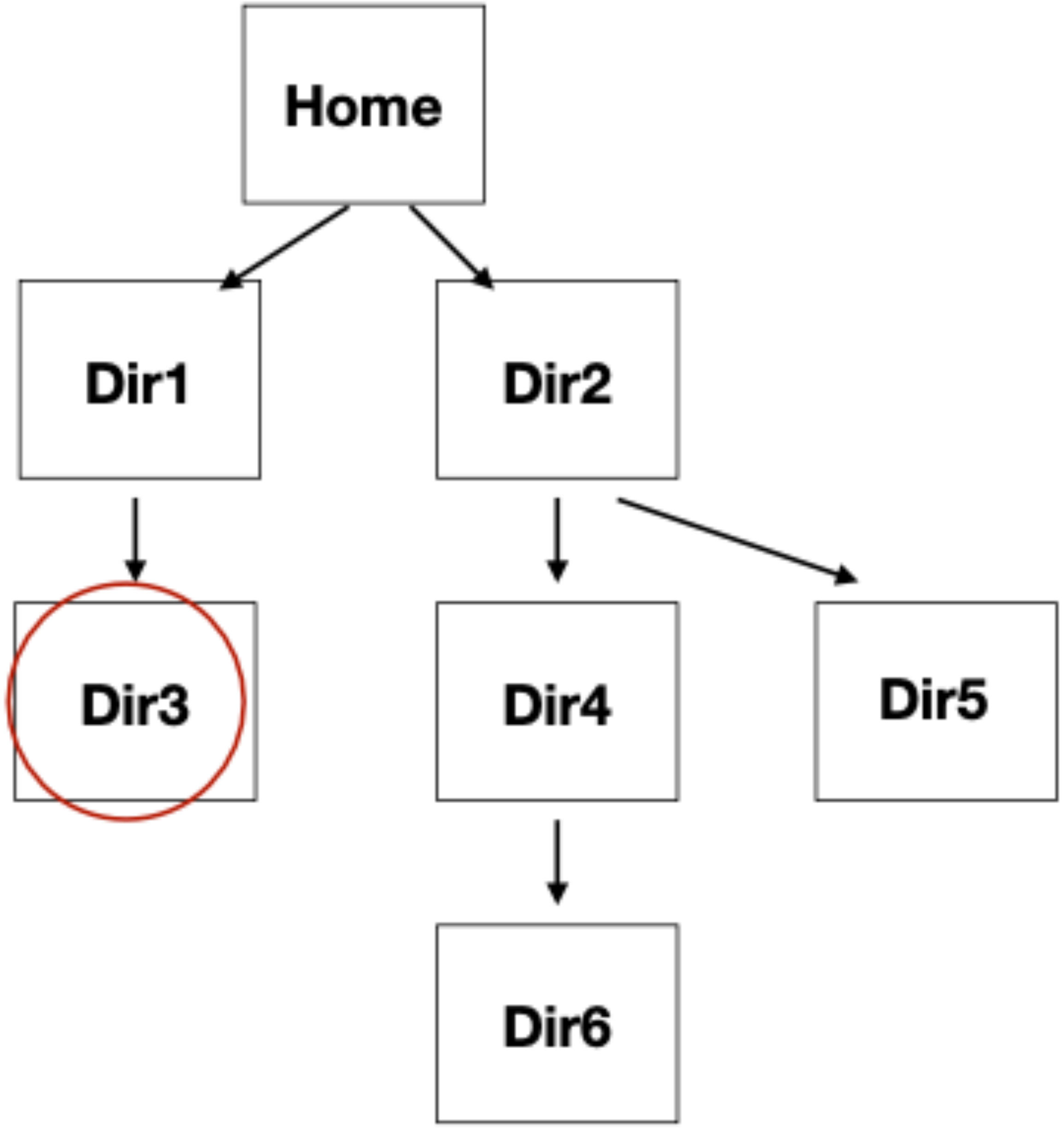
  Note: use this command iteratively to go back as many levels as desired

Exercise



Question 1: currently in Dir4, how do you go to Dir3?

Exercise

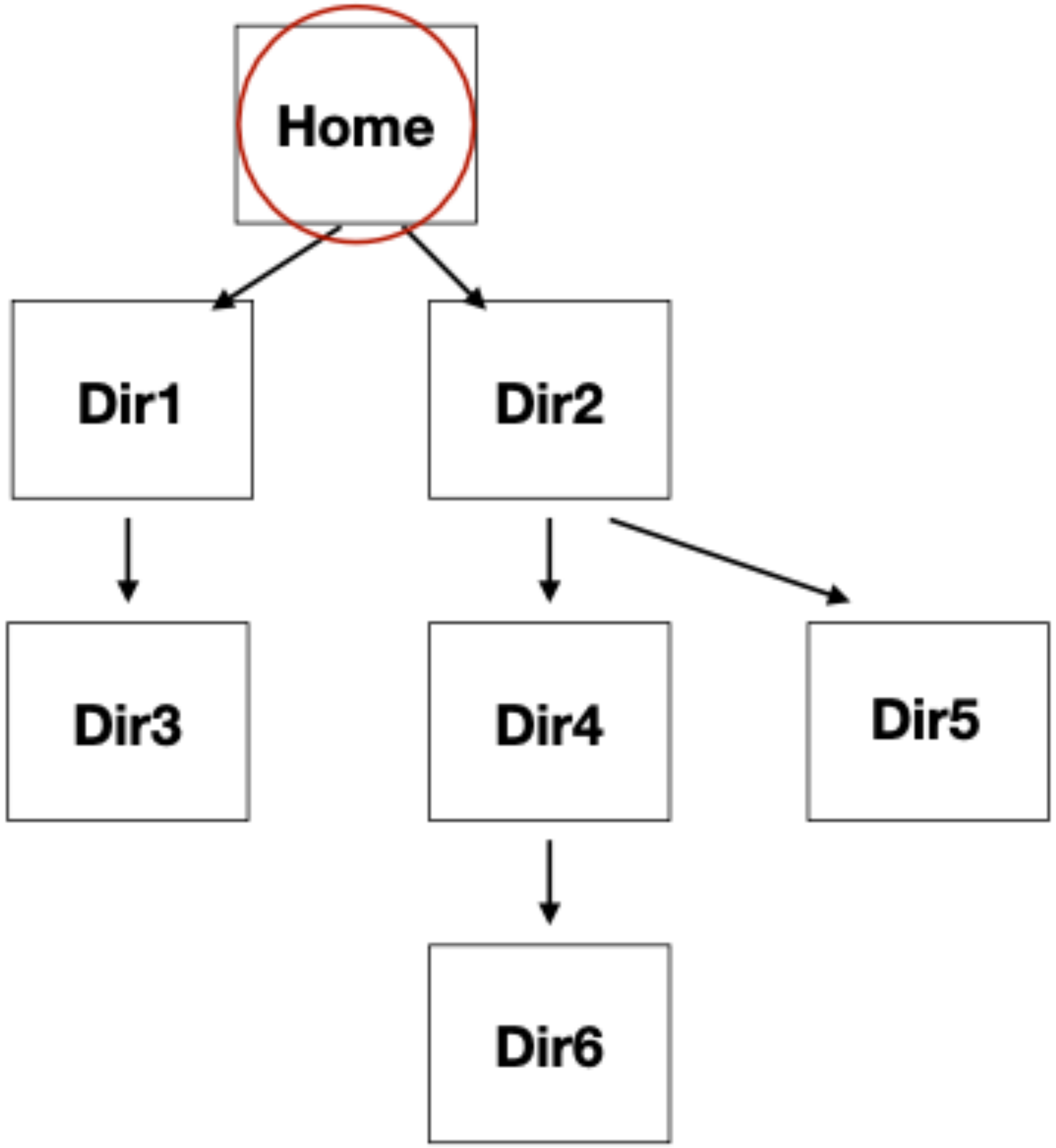


Question 1: currently in Dir4, how do you go to Dir3?

Answer: `cd ../../Dir1/Dir3`

Question 2: currently in Dir3, how do you go Home?

Exercise



Question 2: currently in Dir3, how do you go Home?

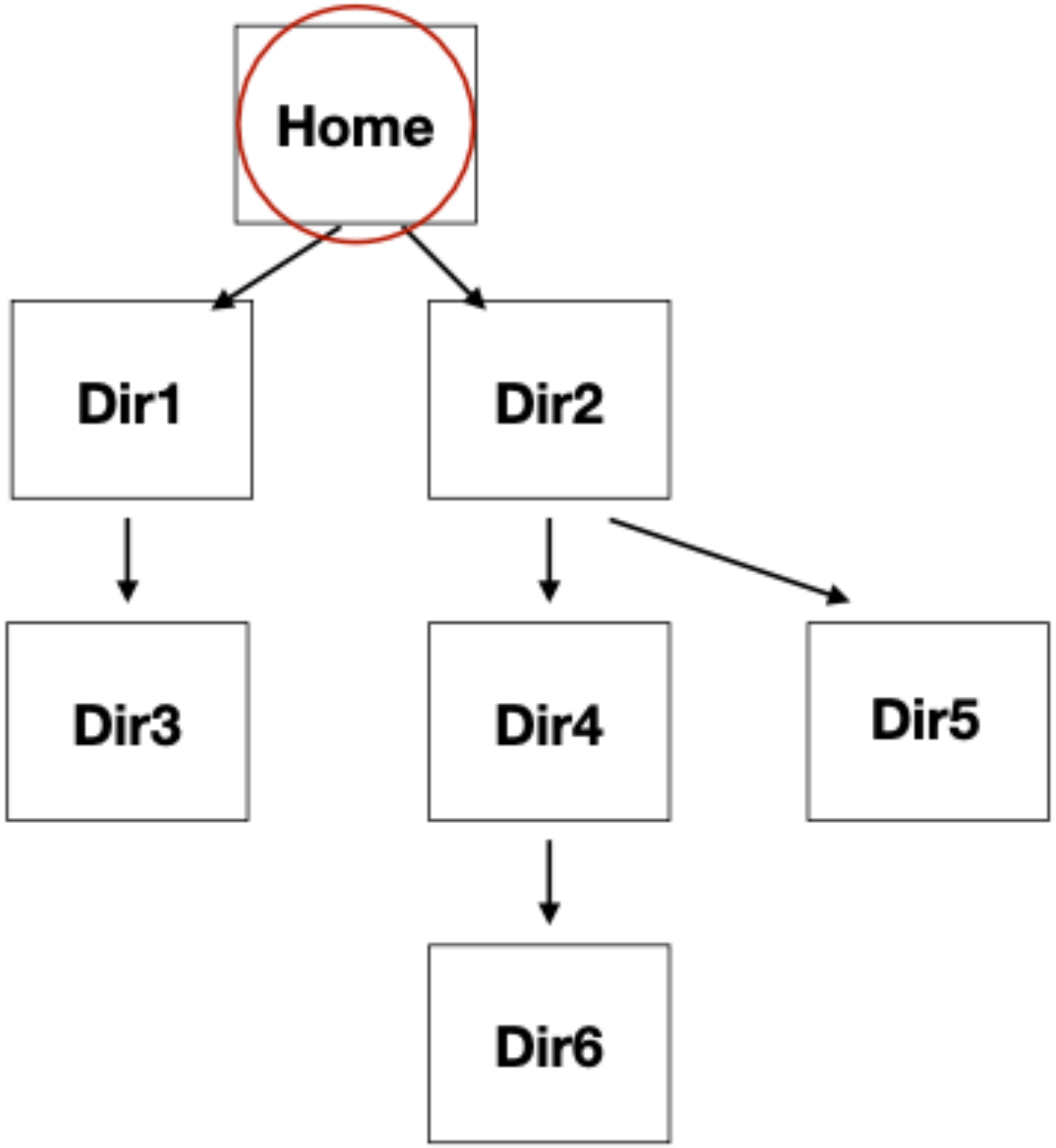
Answer: `cd ../../` or `cd`

`cd:` return to home directory

Directory Management II

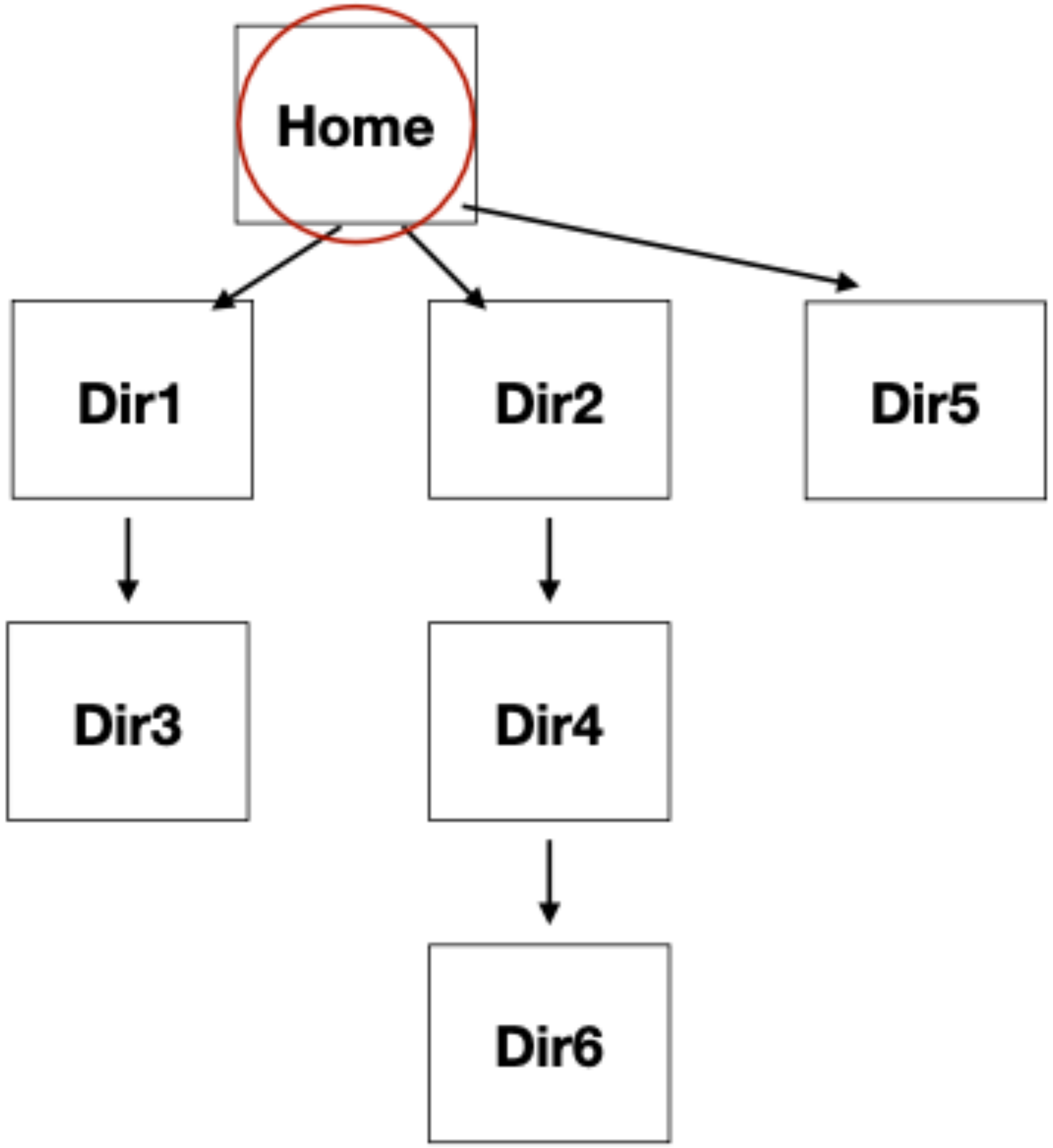
- `mv <dir> <path>`: move directory <dir> to said <path>
- `cp -r <dir> <path>`: copy directory <dir> to said <path>
- `rm -rf <dir>`: remove `<dir>` **permanently**

Exercise II



Question 1: currently in Home, how do you move Dir5 to Home?

Exercise II



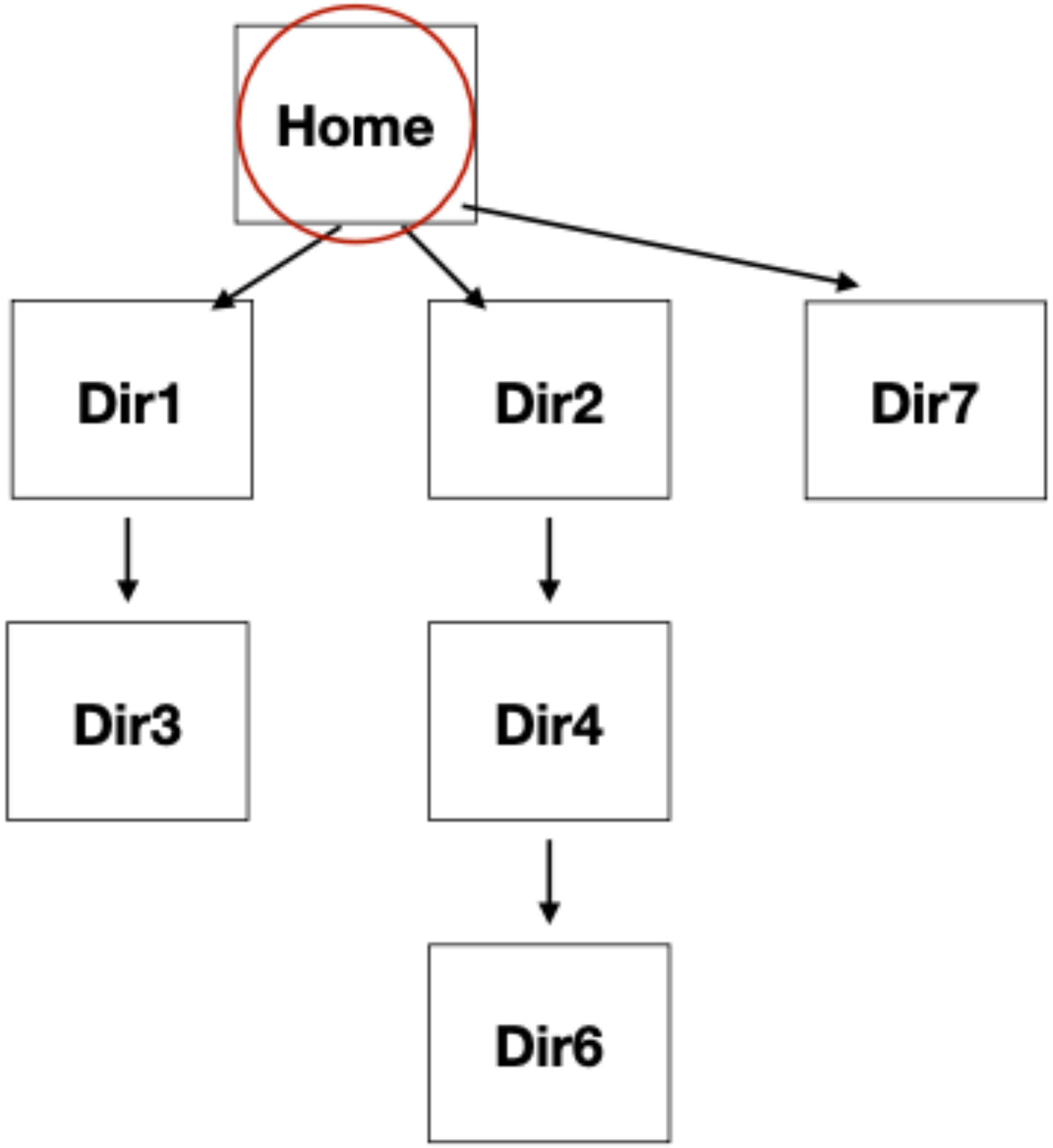
Question 1: currently in Home, how do you move Dir5 to Home?

Answer: `mv Dir2/Dir5 .`

- `.` denotes current directory

Question 2: currently in Home, how do you rename Dir5 to Dir7?

Exercise II



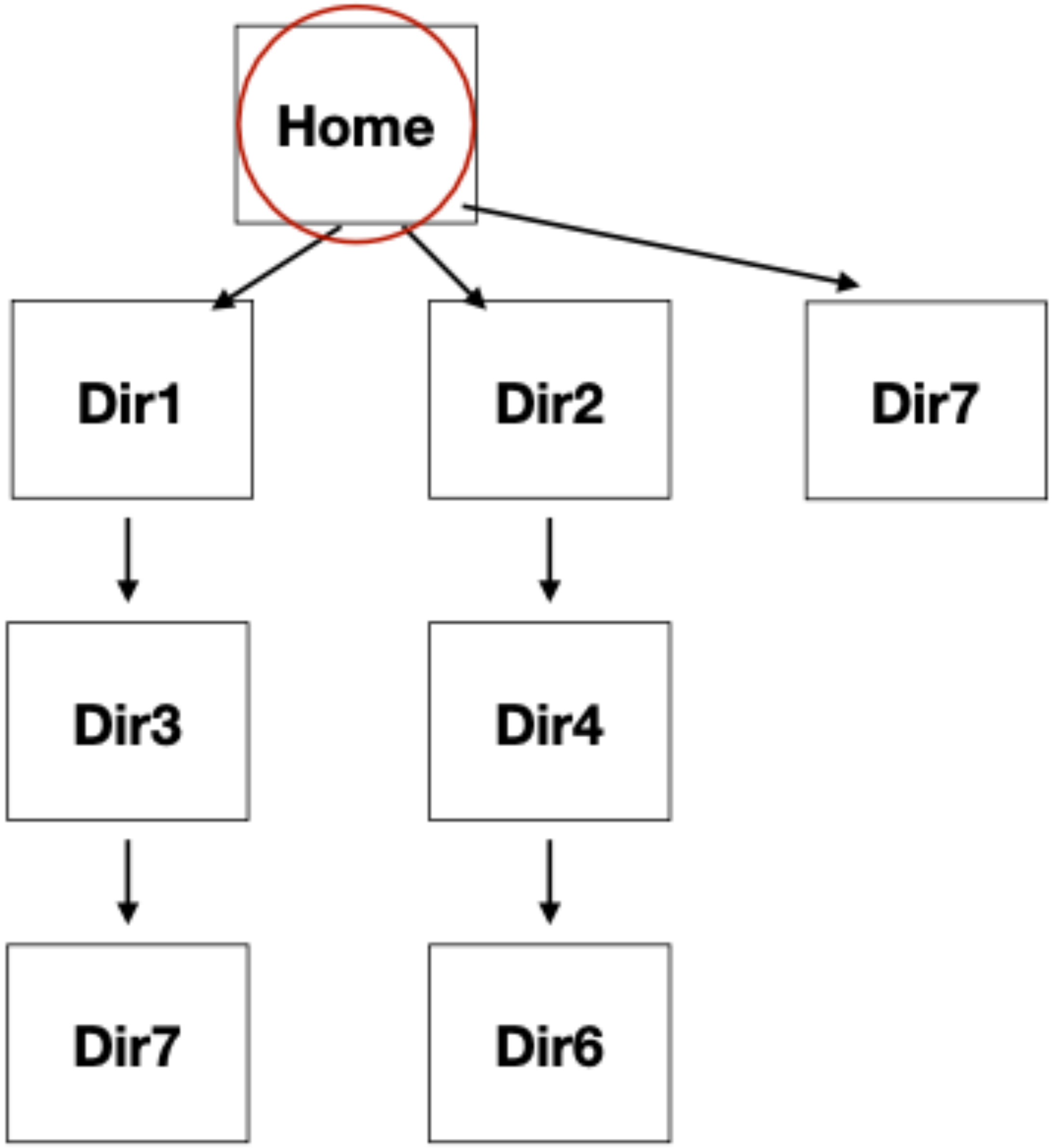
Question 2: currently in Home, how do you rename Dir5 to Dir 7?

Answer: `mv Dir5 Dir7`

`mv` can be used to rename directories

Question 3: currently in Home, how do you copy Dir7 in Dir3?

Exercise II



Question 3: currently in Home, how do you copy Dir7 in Dir3?

Answer: `cp –r Dir7 Dir1/Dir3`

Note: unlike `mv`, `cp` keeps a copy of the folder

Text Editor

Default text editor: `vi` (**vi**sual editor)

- `vi <file>`: if `<file>` exists, it will be opened and if not, it will be created
  - Press `i` to enter *insert mode* to edit the file
  - Press `Esc` to enter *command mode* to quit the file
    - `:wq`: to save changes and quit
    - `:q!`: to not save changes and quit

File Management

- `mv <file> <path>`: move `<file>` to said `<path>`
- `cp <file> <path>`: copy `<file>` to said `<path>`
- `cat <file>`: print entire `<file>` on screen
- `head -n <file>`: print first n lines of `<file>` on screen
- `tail -n <file>`: print last n lines of `<file>` on screen
- `rm <file>`: delete `<file>` **permanently**

Archiving Files

A **tarball** is a set of directories and/or files collected into a single file for distribution or backup purposes

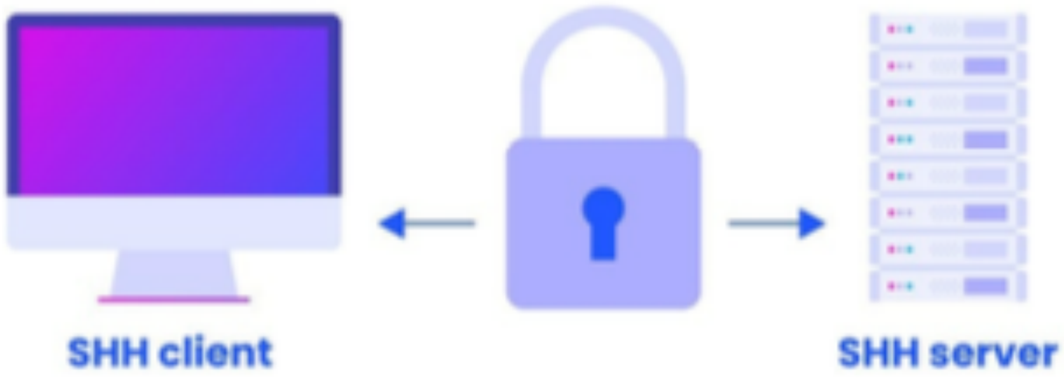
- `tar <options> <file> <dir>`: make `<dir>` into a tarball `<file>`
- `tar <options> <file>`: unpack tarball `<file>`
  - `c`: create a tarball
  - `x`: extract from a tarball
  - `v`: verbose (print out files added/extracted from tarball)
  - `f`: file (it should be followed by the name of the tarball)

Compressing Files

It is sometimes necessary to compress/zip files to save space

- `ls -lh <file>`: to show file size in Kb/Mb/Gb
- `gzip <file>`: compress `<file>`
- `gunzip <file>`: unzip `<file>`

Secure Shell (SSH)



Enables communication with remote computers, e.g: at CERN, Fermilab etc

- `ssh <username+domain>`: log on to remote server
- `logout`: exit remote server
- `scp -r <local:path> <username+domain:path>`: transfer `<file>` in local path to remote server path
- `scp -r <username+domain:path> <local:path>`: transfer `<file>` in remote server path to local path

Summary

- **Managing files and directories** follows a tree structure- the more you use these commands, you'll get a hang of it! Caution: removing a file or directory deletes it forever

- **Text editors**: `vi, emacs, nano`, etc. See what works for you!

- **Archiving and zipping** is important to save space and share files with collaborators

- A lot of work you do will be on remote servers. **SSH commands** are key!

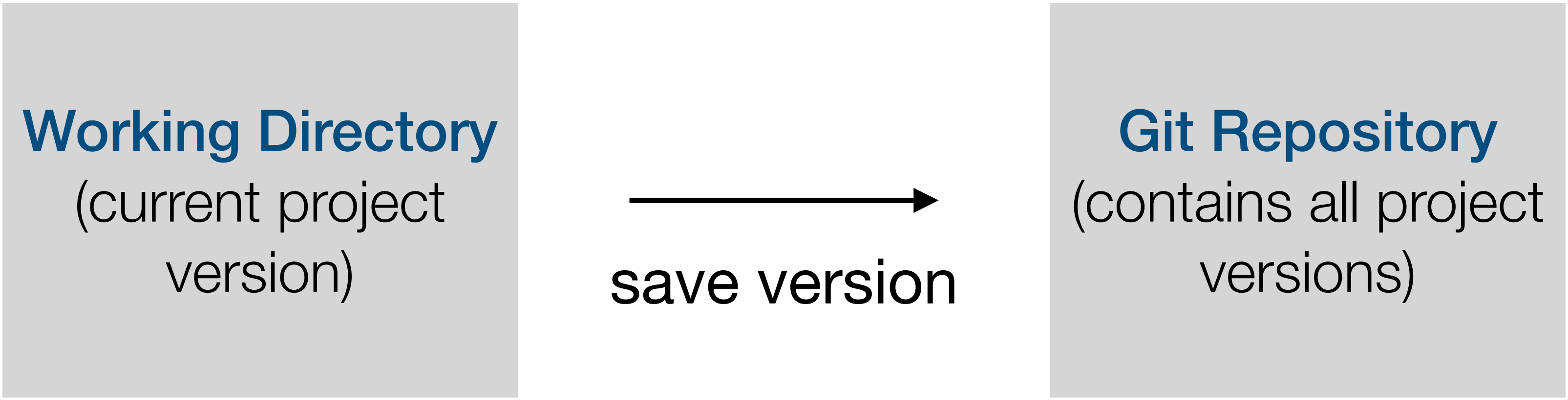
- Tip: use TAB command to autocomplete commands, filenames or directory names! use up and down arrow keys to re-use command prompts!

- Much more documentation online!

Part II: Intro to Git

Git

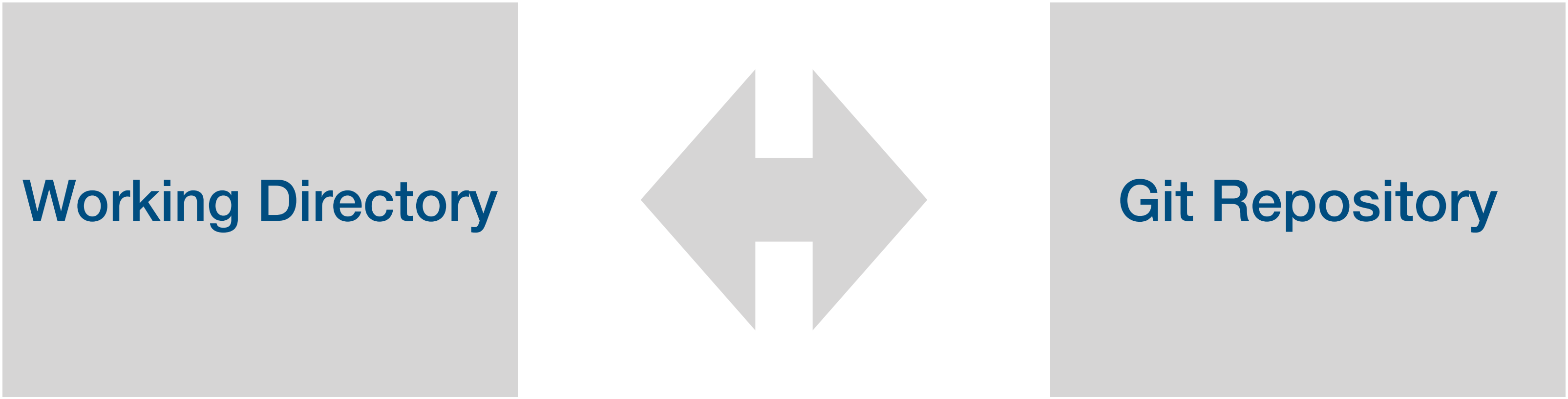
`Git` is a **version control** used to track changes to files



Advantage: can revert to any project version in working directory if necessary

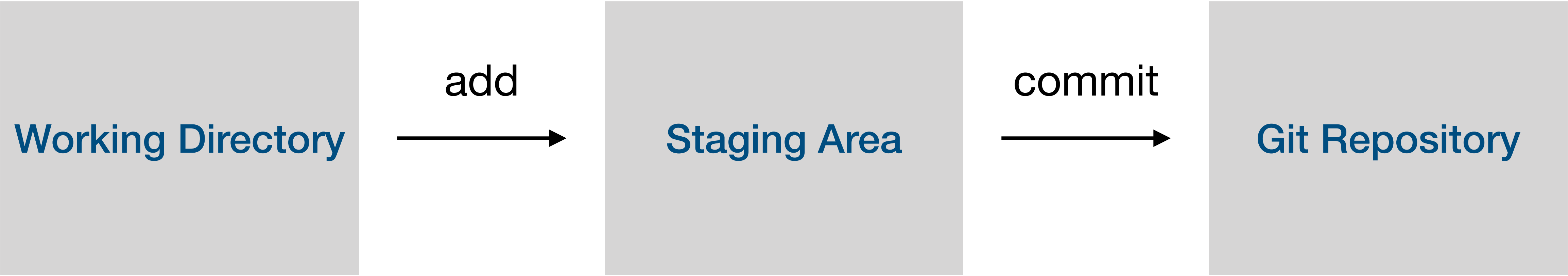
Creating Git Repository

`git init`: creates Git repository



Saving To Git Repository

1. `git add <file>`: put file in working directory to staging area
2. `git commit –m “message”`: put files in staging area to Git repository

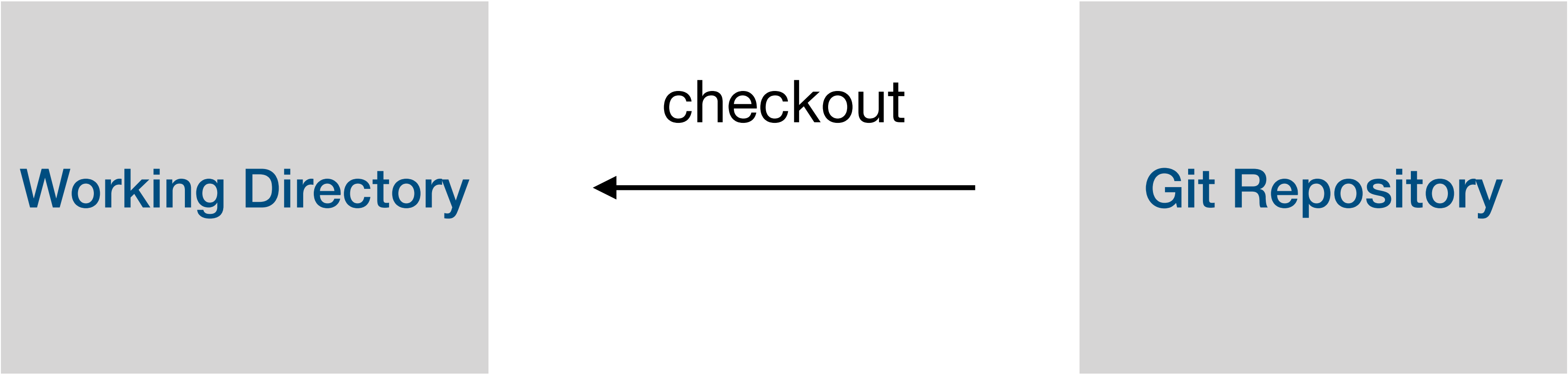


Note: repeat add+commit for every change in working directory

Reverting From Git Repository

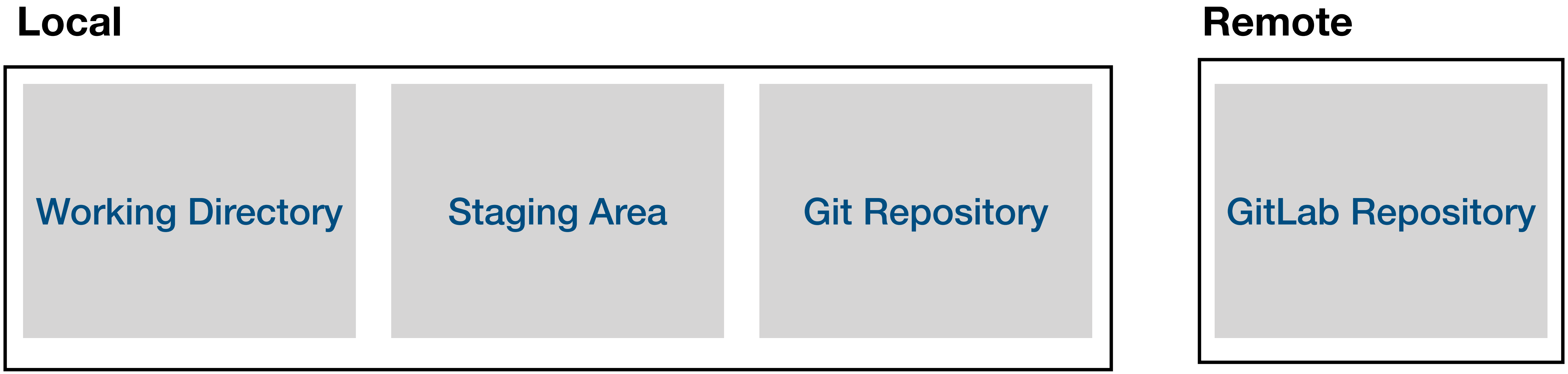
1. `git log`: shows history of every commit made

2. `git checkout <commit number>`: revert to desired version in working directory



GitLab

`GitLab (`or `GitHub` or `BitBucket)`: online website to store local Git repository



Local

Working Directory

Staging Area

Git Repository

Remote

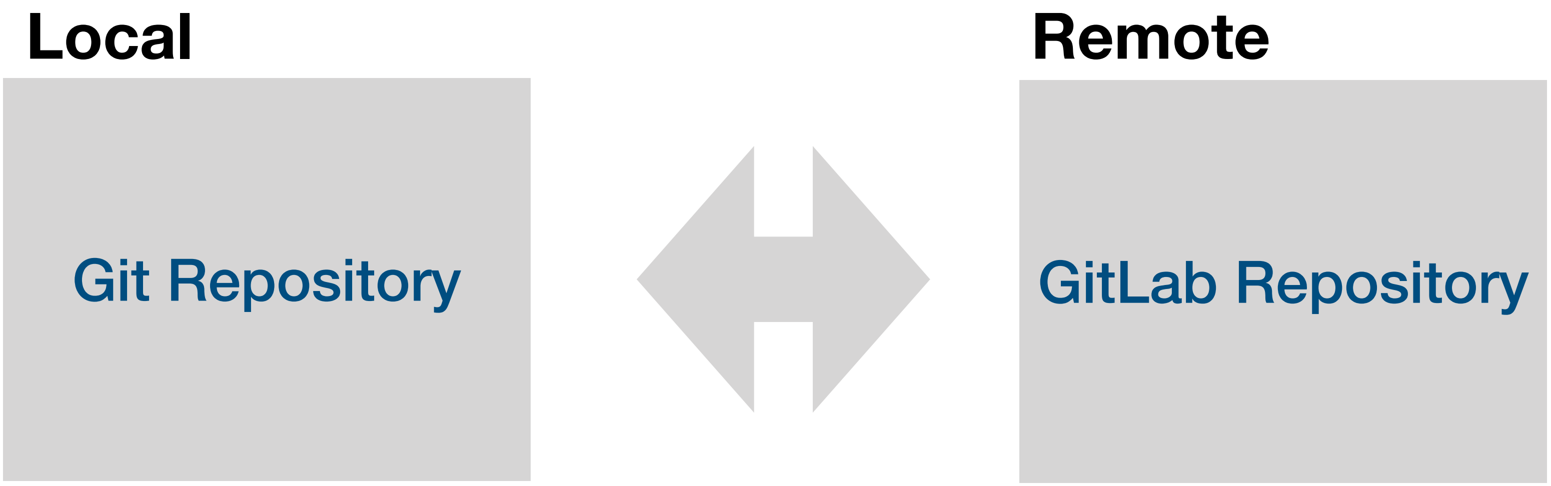
GitLab Repository

Advantage: 1) can access files anywhere 2) can work with collaborators

Connecting Git to GitLab

1. Create project on `GitLab`

2. `git remote add origin <server>`: connecting local Git repository to online <server> repository



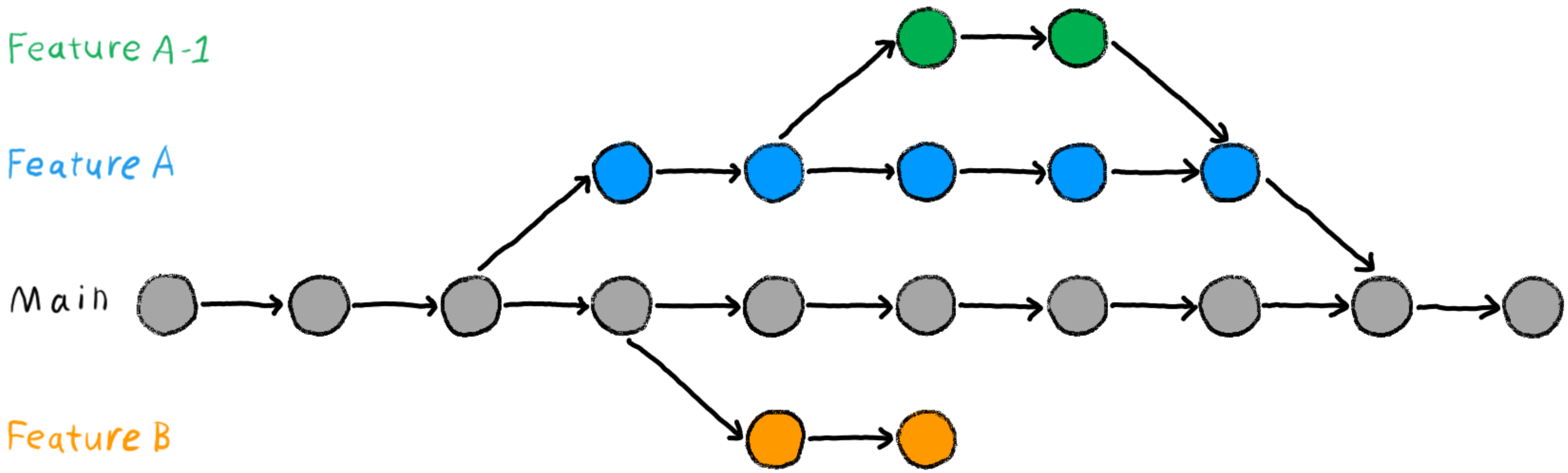
Local

Git Repository

Remote

GitLab Repository

Branch



- The main/master branch is the default branch when you create a repository

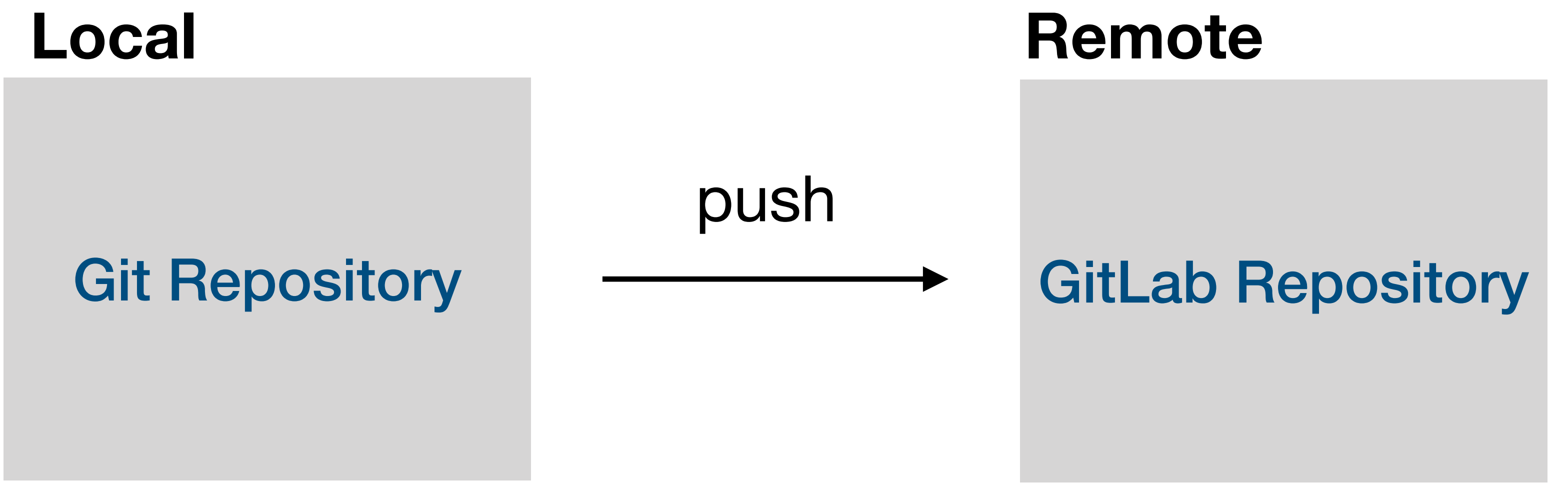
- Use other branches for development and merge them back to the main/master branch if desired

Branch (II)

- `git branch`: all branch names (current branch is marked with *)
- `git checkout -b <branch-name>`: creating `<branch-name>` and switching to it
- `git checkout <branch-name>`: switching to `<branch-name>`

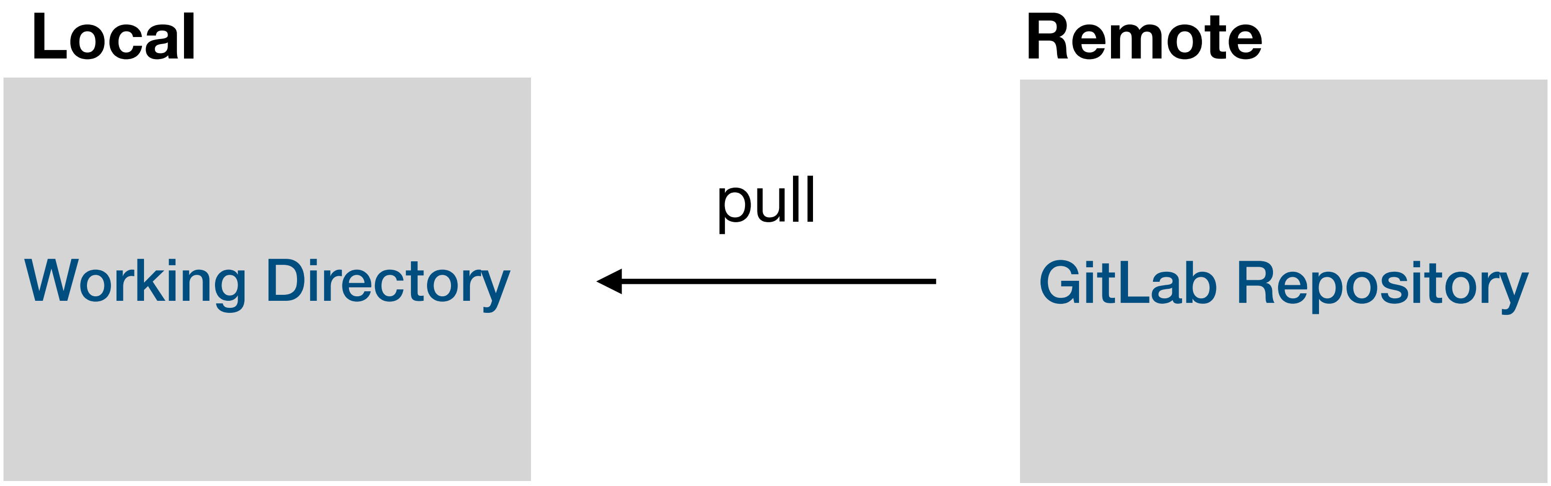
Saving to GitLab Repository

`git push origin <branch-name>`: put `<branch-name>` to GitLab repository

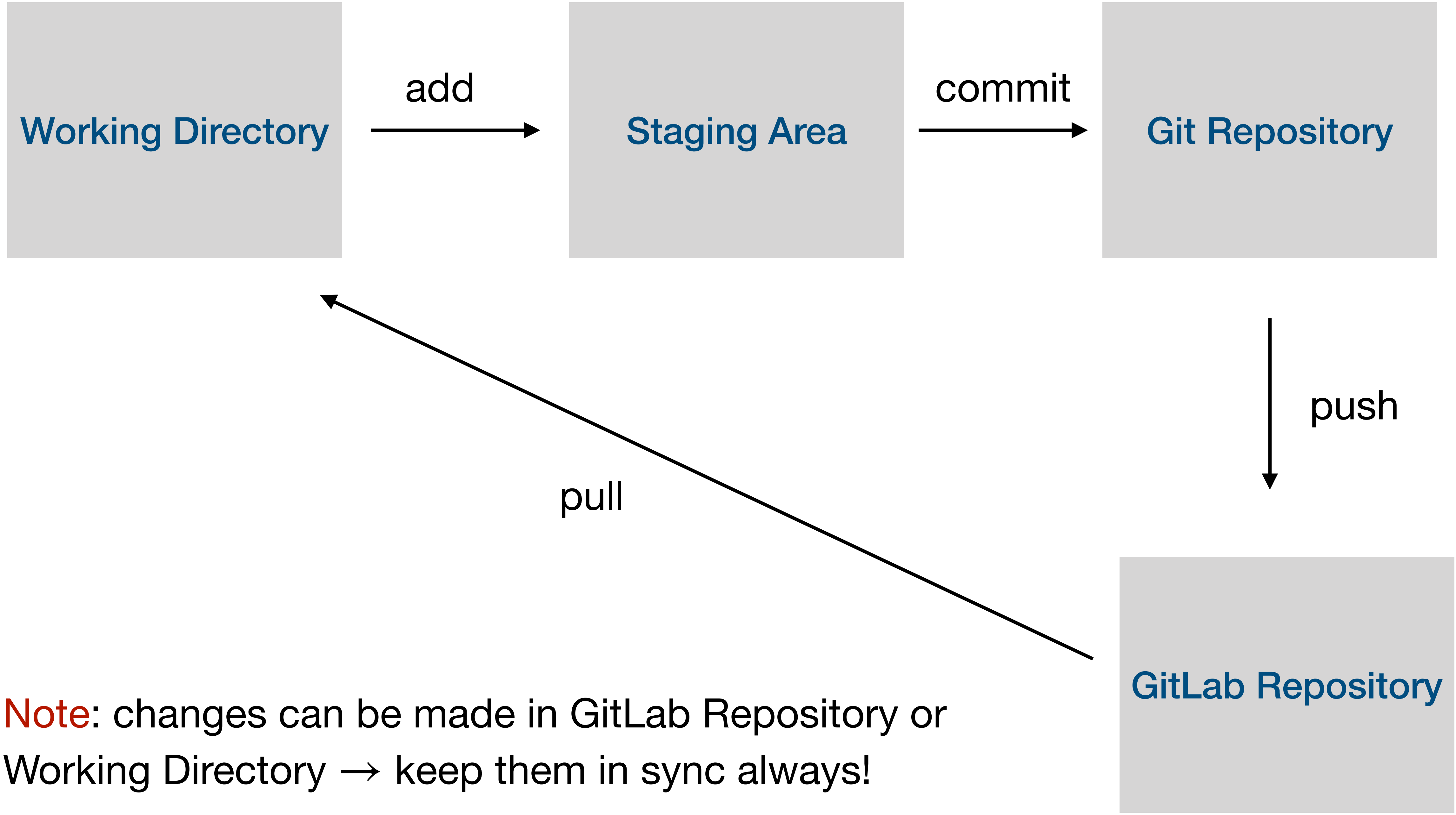


Saving to Working Directory

`git pull origin <branch-name>`: save `<branch-name>` to working directory



Summary



Working Directory

add

Staging Area

commit

Git Repository

push

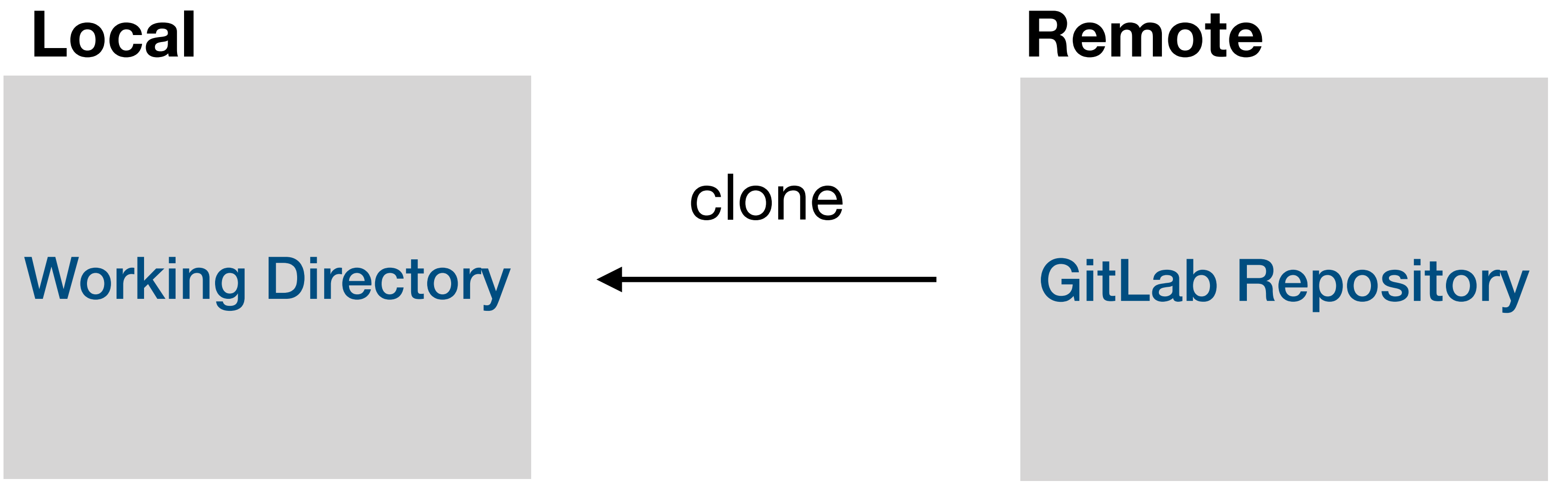
GitLab Repository

pull

Note: changes can be made in GitLab Repository or Working Directory → keep them in sync always!

Cloning

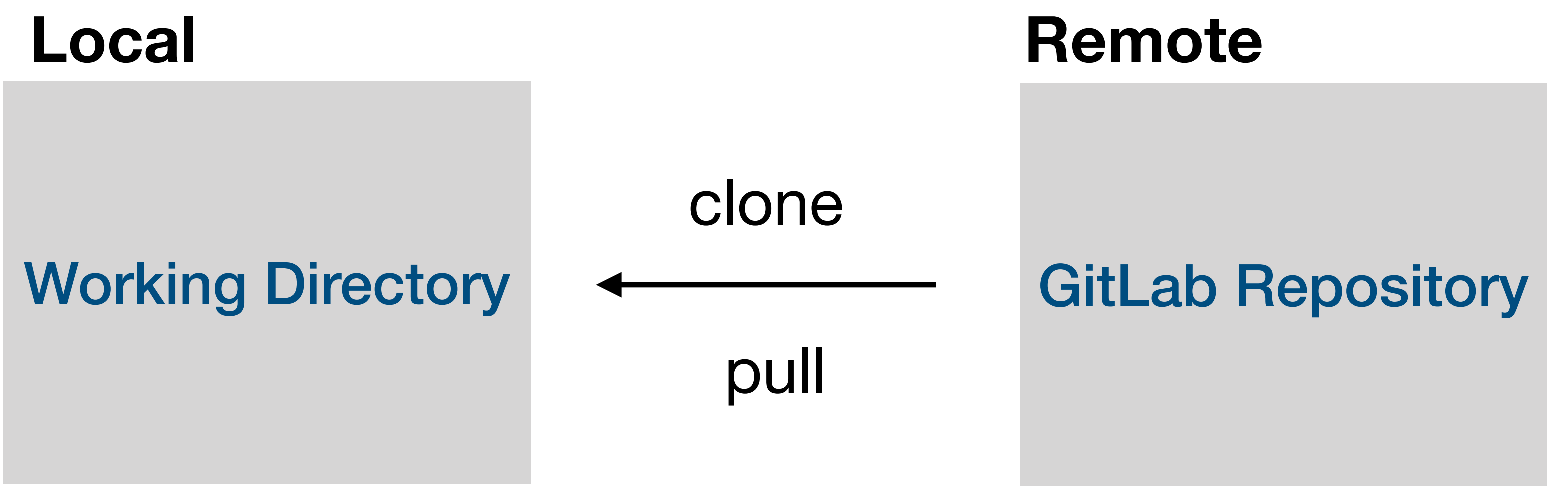
`git clone <url>`: clone remote GitLab repository `<url>`



Advantage: no need to start a project from scratch

Cloning vs Pulling

`git clone`	`git pull`
Copies all files to the working directory	Copies only modified files to the working directory
Creates a connection between online repository and working directory	Requires a connection to have been made already
Typically used once	Typically used multiple times



Setup

- Check if git is installed: `git - -version` in shell
  (Download link: https://git-scm.com/)

- Set up git configuration:
  ```
  git config --global user.name “Gitanjali Poddar”
  git config --global user.email gitanjali.poddar@cern.ch
  ```

- Create GitHub/GitLab account (https://github.com/)

Questions?