MÃ NGUỒN MỞ TRONG KHOA HỌC DỮ LIỆU

Bài 01.
QUẢN LÝ MÃ NGUỒN
Git / Github (tiếp theo)

ThS. Lê Nhật Tùng

Cheatsheet: Basic commands

Basic working commands:

• `git init`	initialize empty repo
• `git clone`	deep copy existing repo
• `git diff`	show changes current dir vs. HEAD
• `git add`	stage files to snapshot
• `git commit`	save a code snapshot
• `git reset`	alter state of saved snapshots

Basic branch commands:

• `git branch`	list/create/delete branches
• `git checkout`	alter branch state
• `git fetch`	sync cached branches with remote
• `git rebase [x]`	join branch `x` into current
• `git merge [x]`	join branch `x` into current
• `git push`	publish local to remote branch
• `git pull`	same as fetch + rebase/merge

Cheatsheet: Advanced commands

Advanced commands:

• `git stash`	put current changes into cache
• `git stash pop`	apply cached changes into branch
• `git push --force`	disregard safety checks when pushing *(!)*
• `git commit --amend`	add some stuff to latest commit
• `git rebase -i [X+1]`	squash multiple commits into single one
• `git blame [file]`	check who changed what in `file`
• `git cherry-pick`	merge single commit(s) into current branch
• `git revert`	remove single commit(s) from current branch
• `git tag`	create a labeled snapshot of the repo

Note: these commands might break your toys -- use with caution!

Các lệnh Git: làm việc với kho lưu trữ trung tâm

- **git init –bare**: để tạo barebone cho một kho lưu trữ trung tâm (không có commit ban đầu)
- **git clone [repo_name] [clone_name]:** để tạo một bản sao được liên kết của kho lưu trữ repo_name.
- **git fetch:** lấy thông tin về các commit/nhánh mới từ kho lưu trữ trung tâm.
- **git pull:** để lấy các commit (mới) từ kho lưu trữ trung tâm vào kho lưu trữ của chúng ta.
- **git push:** để đưa các commit (mới tạo) của chúng ta vào kho lưu trữ trung tâm.
- Chúng ta sẽ giả vờ rằng chúng ta có hai nhà phát triển và một kho lưu trữ trung tâm.

Quick recap

- Trong không gian làm việc của chúng ta, hãy tạo một thư mục mới RR_git2 và vào bên trong
  - **mkdir RR_git2**
  - **cd RR_git2**

- Hãy tạo một kho lưu trữ có tên là RecapRepo và đi vào bên trong
  - **git init RecapRepo**
  - **cd RecapRepo**
  - **git status**
- Thay đổi local user.name thành "RecapRepoUser"
  - **git config --local user.name "RecapRepoUser"**

Quick recap

- Tạo ba tệp: commit tệp đầu tiên; chỉ dàn dựng tệp thứ hai; không làm gì với tệp thứ ba
  - **echo "this file will be commited" > file1.txt**
  - **echo "this file will be staged" > file2.txt**
  - **echo "this fille will be in the working directory" > file3.txt**
  - **git add file1.txt**
  - **git commit -m "Added file1.txt"**
  - **git add file2.txt**
  - **git status**

Quick recap

- Trong RR_git2 tạo một bản sao của kho lưu trữ đầu tiên; đi vào bên
  - **cd ..**
  - **git clone RecapRepo RecapCloned**
  - **cd RecapCloned**
-
- Xem nội dung; kiểm tra trạng thái
  - **ls**
  - **git status**

Bài tập 5: mô phỏng kho lưu trữ trung tâm với hai máy cục bộ

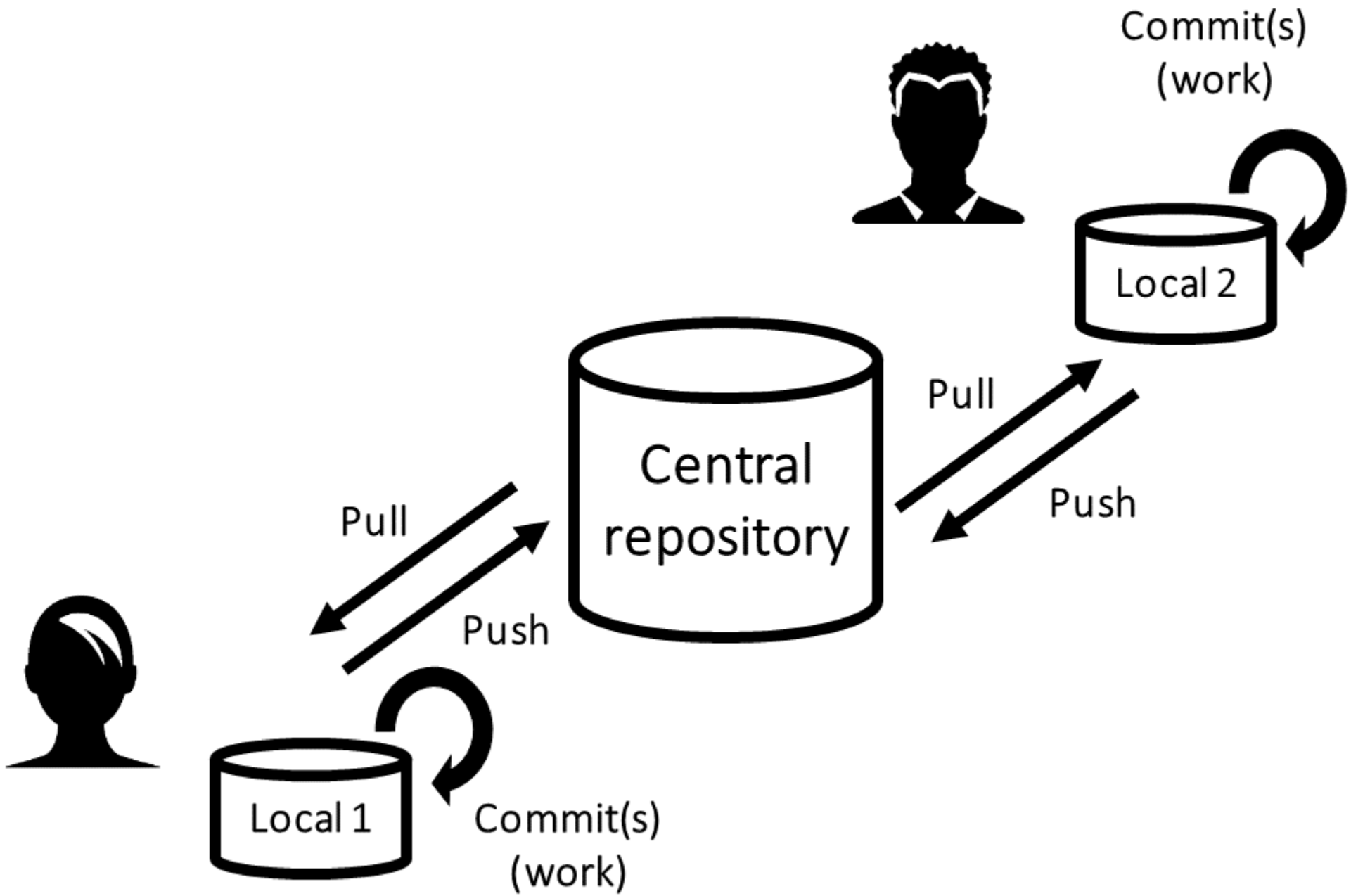
- Bây giờ chúng ta sẽ giả vờ có hai người cộng tác, sử dụng kho lưu trữ trung tâm để quản lý quy trình làm việc.
- Trong thư mục RR_git2 của bạn, hãy tạo một kho lưu trữ mới có tên là "CentralRepo"
- (Hãy nhớ sử dụng tùy chọn phù hợp khi khởi tạo kho lưu trữ!)
    - **cd RR_git2**
    - **git init --bare CentralRepo**

Bài tập 5: mô phỏng kho lưu trữ trung tâm với hai máy cục bộ

- Trong thư mục **RR_git2** của bạn, hãy tạo một bản sao của CentralRepo, có tên là **Dev1**
  - **git clone CentralRepo Dev1**
- Trong thư mục **RR_git2** của bạn, hãy tạo một bản sao của CentralRepo, có tên là **Dev2**
  - **git clone CentralRepo Dev2**
- Đặt tên người dùng cục bộ cho kho lưu trữ Dev1 thành **"Developer_1“**
  - **cd Dev1**
  - **git config --local user.name "Developer_1"**

Bài tập 5: mô phỏng kho lưu trữ trung tâm với hai máy cục bộ

- Đặt tên người dùng cục bộ cho kho lưu trữ Dev1 thành "**Developer_1**“
  - **cd Dev1**
  - **git config --local user.name "Developer_1“**
- Đặt tên người dùng cục bộ cho kho lưu trữ Dev2 thành "**Developer_2**“
  - **cd Dev1**
  - **git config --local user.name "Developer_2"**


Commit(s)
(work)
Local 2
Pull
Push
Central
repository
Pull
Push
Local 1
Commit(s)
(work)

Bài tập 5: mô phỏng kho lưu trữ trung tâm với hai máy cục bộ

- Dev1 khởi động dự án bằng cách tạo nhánh với một cam kết và gửi nó ngược dòng
  - **echo "This will be the file with code" > code.R**
  - **git add .**
  - **git commit -m "Added the file with code"**
  - **git status**
  - **git push**
  - **git status**
- Dev2 muốn bắt kịp tốc độ nên nắm bắt những thay đổi: **git pull**

Bài tập 6: fetching and updating before pushing

- Tạo tệp readme.txt trong kho lưu trữ Dev1 của bạn.
    - **cd ../Dev1**
    - **touch readme.txt**
    -
- Stage, commit and push the file.
    - **git add .**
    - **git commit -m "Added readme.txt"**
    - **git push**

Bài tập 6: fetching and updating before pushing

- Go to the Dev2 repository.
  - **cd ../Dev2**
- Run git status
  - **git status**
- Fetch information about changes from the central repository. Run git status.
  - **git fetch**
  - **git status**
- Pull the information. Check folder contents.
  - **git pull**
  - **ls**

Bài tập 6: fetching and updating before pushing

- Add a new line to readme.txt and create a readme.md file.
  - **echo "second line of text" >> readme.txt**
  - **touch readme.md**

- Stage, commit and push the file to the central repository.
  - **git add .**
  - **git commit -m "Added readme.md and changed readme.txt"**
  - **git push**

Bài tập 6: fetching and updating before pushing

- In Dev1, create a text3.txt file with the line "A line added by Dev1". Stage, commit and push - what happens? Resolve the issue following the hints.

- **cd ../Dev1**
- **echo "A line added by Dev1" > text3.txt**
- **git add .**
- **git commit -m "Added text3"**
- **git push**

- Follow the instructions from Git to resolve merge conflicts.

Bài tập 7: xử lý đụng độ

- Trong Bài tập 6, chúng ta đã có sự phân kỳ, nhưng không có xung đột (các cam kết khác nhau, nhưng không có thay đổi nào xung đột với các tệp).

- Điều gì sẽ xảy ra nếu hai nhà phát triển tạo ra hai phiên bản khác nhau của một tệp?

- Trong Dev2, tạo một tệp mới có tên là text3.txt với dòng "Một dòng được Dev2 thêm vào", Stage, commit và thử push. Đọc các gợi ý, làm theo và chuyển sang bước 2.

Bài tập 7: xử lý đụng độ

- Có xung đột vì Dev1 đã cam kết một tệp text3.txt khác trước đó.
- Chúng ta đang ở chế độ hợp nhất. Chúng ta có thể chỉnh sửa tệp (ví dụ: trong Notepad) để tìm ra những gì nên giữ lại. Sau đó dàn dựng và cam kết.

- Hợp nhất trở thành một cam kết mới (thử git log)