Lecture 2: Data Management and Version Control, Git/GitHub

LING 1340/2340: Data Science for Linguists

Na-Rae Han

Objectives

- Tools:
  - Git and GitHub

- To-do #1
  - How was Git?

You should be taking NOTES!

First thing to do every class

1. Open up a Terminal/Git Bash window ("shell" window).

2. Move into your Data_Science directory.

   `cd Documents/Data_Science` ◄········ Hit TAB for auto-completion.

3. Make sure you are in the right directory.

   `pwd` ◄········ "Print Working Directory"

4. Look at what's inside the directory.

   `ls`

   or

   `ls -la`

   `ls` for "list directory".
   `-la` for "long/all". Shows all hidden files in long output.


```
MINGW64:/c/Users/narae/Documents/Data_Science

narae@X1Yoga MINGW64 ~
$ cd Documents/Data_Science/

narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ pwd
/c/Users/narae/Documents/Data_Science

narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ ls
Class-Exercise-Repo/  languages/

narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ ls -la
total 12
drwxr-xr-x 1 narae 197121 0 Jan 10 14:30 ./
drwxr-xr-x 1 narae 197121 0 Jan  8 18:33 ../
drwxr-xr-x 1 narae 197121 0 Jan 10 14:30 Class-Exercise-Repo/
drwxr-xr-x 1 narae 197121 0 Jan  8 18:34 languages/
```

Your first local repository: getting started

Steps in Tutorial Part 1, [Creating a Repository](#)

1. Create a directory called
   `languages`

2. Initiate it as a Git repository:

   `git init`

3. Create a new text file 'zulu.txt', add lines to it

4. Add files to staging area:

   `git add zulu.txt`

5. Commit the change:

   `git commit -m "started zulu"`

6. Edit the text file again

7. Add files to be committed:

   `git add zulu.txt`

8. Commit the change:

   `git commit -m "details on…"`

Check status between steps:
`git status`

Your first local repository: tracking, history

Steps in Tutorial Part 1: [Tracking Changes](#), [A Commit Workflow](#), and [Exploring History](#).

- To view entire version history:

  ```
  git log
  ```

- To view changes:

  ```
  git diff
  git diff HEAD~1 file.txt
  git diff --staged
  ```

- To view what changed in a particular version:

  ```
  git show HEAD~1
  ```

- To scrap new changes since the last commit:

  ```
  git checkout HEAD file.txt
  ```

- To restore an earlier version:

  ```
  git checkout VERSION file.txt
  ```
  ← commit to make this the new HEAD

If thrown into pagination, use SPACE to page down, q to quit.

HEAD: the last committed version
HEAD~1: one before that

To-do #1: Your first local repository

▶ Your directory `languages/` was set up with a **Git repository**.

▶ `languages/` is now:

- ◆ tracked by Git
- ◆ all changes will be documented
- ◆ able to revert back to earlier version, if needs be

▶ But is this all?

- ◆ How about backup? collaboration? social?

commit
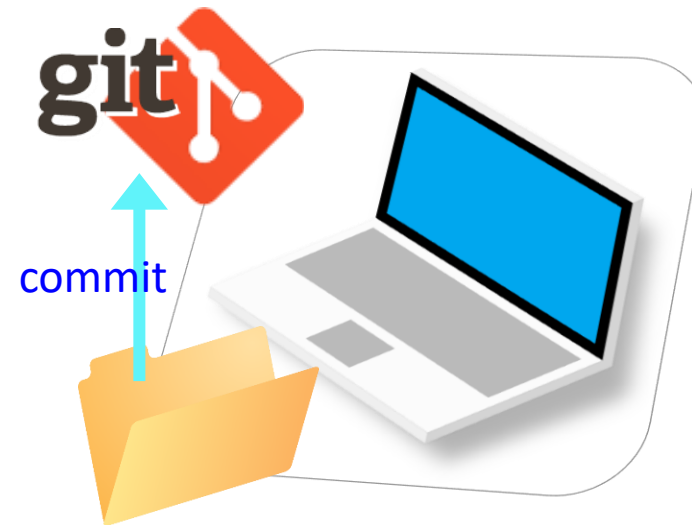
GitHub: a *remote* repository

▶ This is where **GitHub** comes in.

▶ GitHub is a **repository hosting service**.

   ← A website where you can keep a copy of your Git repository.

   ← REMOTE repository on GitHub, LOCAL repository on your laptop.

   ← Great way to backup, and also showcase your work

pull

push

commit
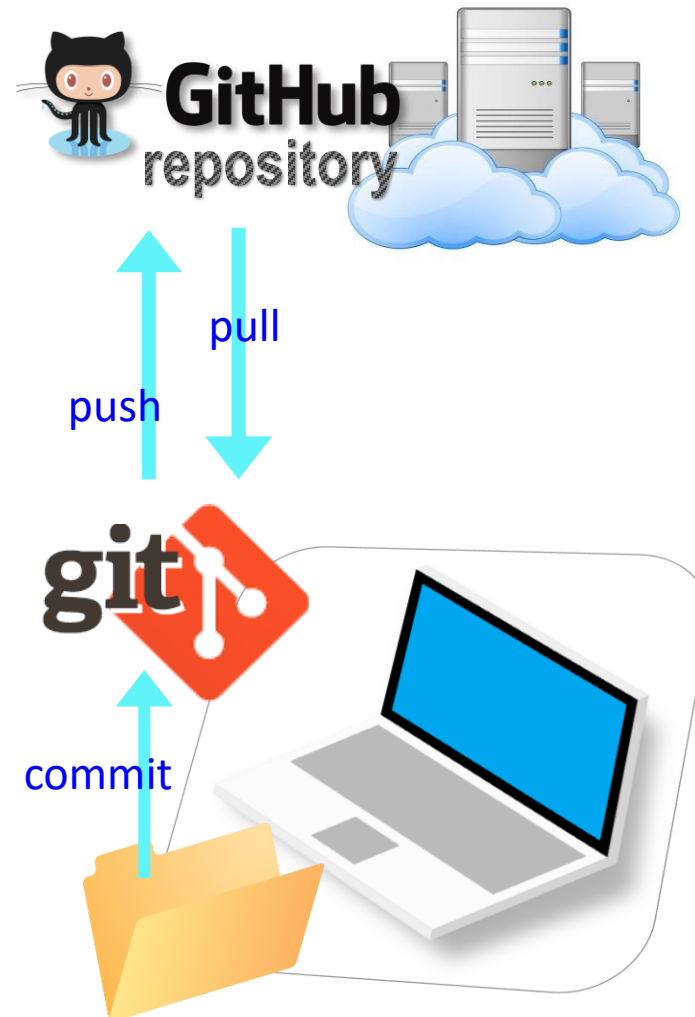
Setting up GitHub

▶ Create a GitHub account at https://github.com/

- ◆ Use your **Pitt email address.**

- ◆ If you already have an account with a different email, add your Pitt email to your account.

- ◆ GitHub sends you a verification email. Confirm.

  ⬅ The verification email might go to the SPAM folder. You MUST resolve and verify!

▶ You get your own profile page. This is mine:

- ◆ https://github.com/naraehan

- ◆ Check your URL!

- ◆ I also have a secondary "student" account: https://github.com/narae-student

Setting up a remote ("GitHub") repo

▶ There are TWO main methods of setting up a remote GitHub repo.

Scenario 1: Your laptop already has an **existing LOCAL Git repo**. You configure it to link it up to a new, empty repo on GitHub, then push up the content.

- ◆ We can set up our `languages` repo with a GitHub repo this way.
- ◆ My LSA tutorial Part 2 [Linking Git with GitHub](#) goes this route.

Scenario 2: Start from remote. Create a **new repository on GitHub**, and then **clone it onto your laptop** as a brand-new local repository.

← Let's try this!

Your first GitHub repo

▶ On GitHub, create a new repository called "practice-repo".

◆ Provide a short description.

◆ Keep it public.

◆ Initialize it with a README.

Owner * | Repository name *

👤 narae-student ▾ / practice-repo ✓

Great repository names are short and memorable. Need inspiration? How about sturdy-oct

Description (optional)

Will be using this repository for Git/GitHub practicing.

⦿ 📘 Public
Anyone on the internet can see this repository. You choose who can commit.

◯ 🔒 Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☑ Add a README file
This is where you can write a long description for your project. Learn more.

☐ Add .gitignore
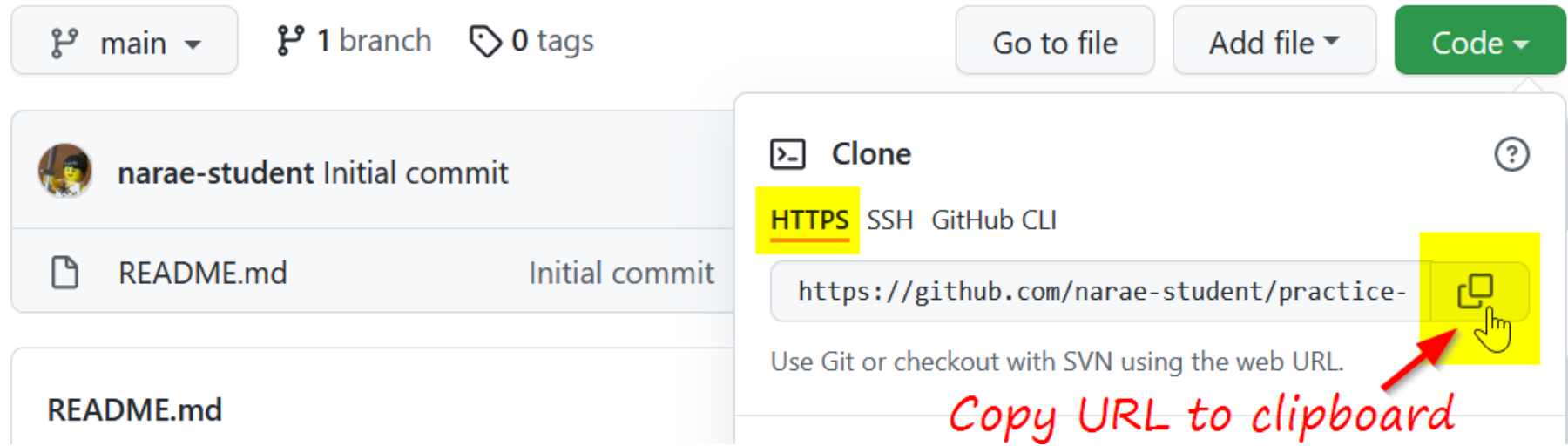Choose which files not to track from a list of templates. Learn more.

☐ Choose a license
A license tells others what they can and can't do with your code. Learn more.

This will set ⑂ main as the default branch. Change the default name in your settings.

Create repository

Cloning first GitHub repo

▶ GitHub shows a URL to use in cloning. Copy to clipboard.

Copy URL to clipboard

▶ In Terminal/Git Bash, move into your Data_Science/ directory (use `cd` command,) then execute (paste copied URL):

`git clone https://github.com/yourid/practice-repo.git`

Sign in with your browser

← A "Sign in" window will pop up. Choose "Sign in with your browser" and continue.

← practice-repo directory is cloned as a **local** repository.

Local repository ↔ remote repository

▶ After committing, you now need to *push* to remote repo.

1. Create a new text file 'notes.txt'
2. Add files to be committed:

   `git add notes.txt`

3. Commit:

   `git commit -m "first commit"`

4. **Push change to GitHub: git push**

5. Edit the text file

6. Add files to be committed:

   `git add notes.txt`

7. Commit:

   `git commit -m "changed x, y, z"`

8. **Push change to GitHub: git push**

Check frequently:
`git status`
`git diff`
`git log`

pull

push

No need unless collaborating
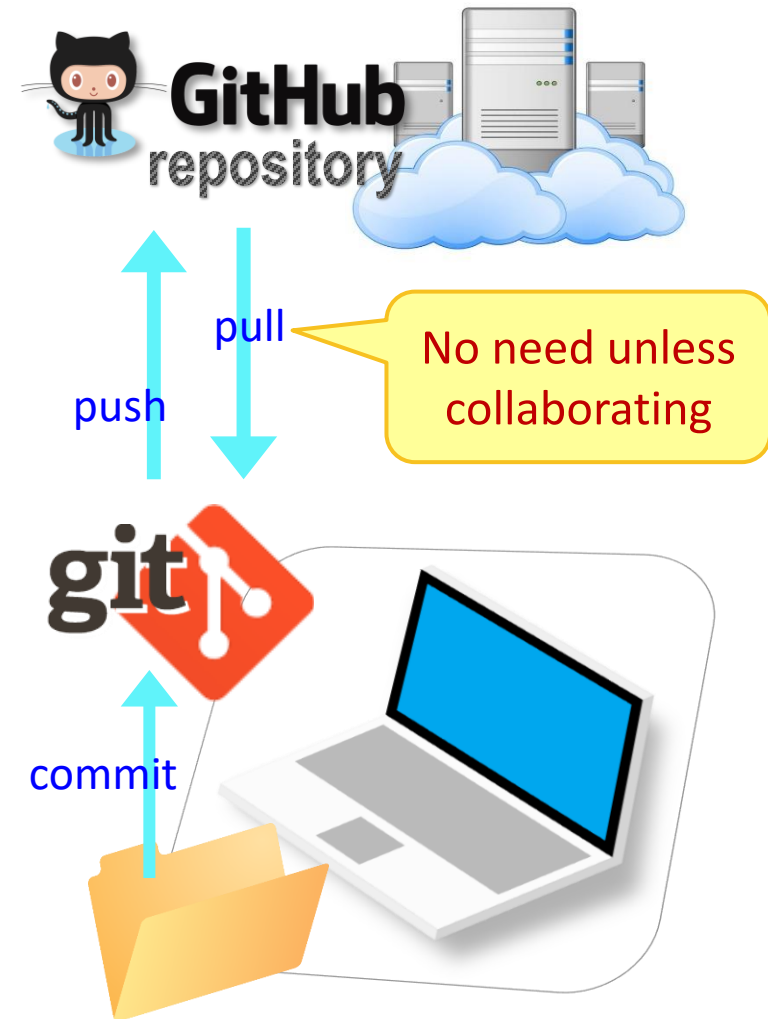
commit

Try it out: Your first git/GitHub repo

5 minutes

▶ Create "Practice-Repo" on GitHub

▶ Clone it to your laptop

▶ Make changes to your local repo

▶ Local Git operations: git status, git add, git commit
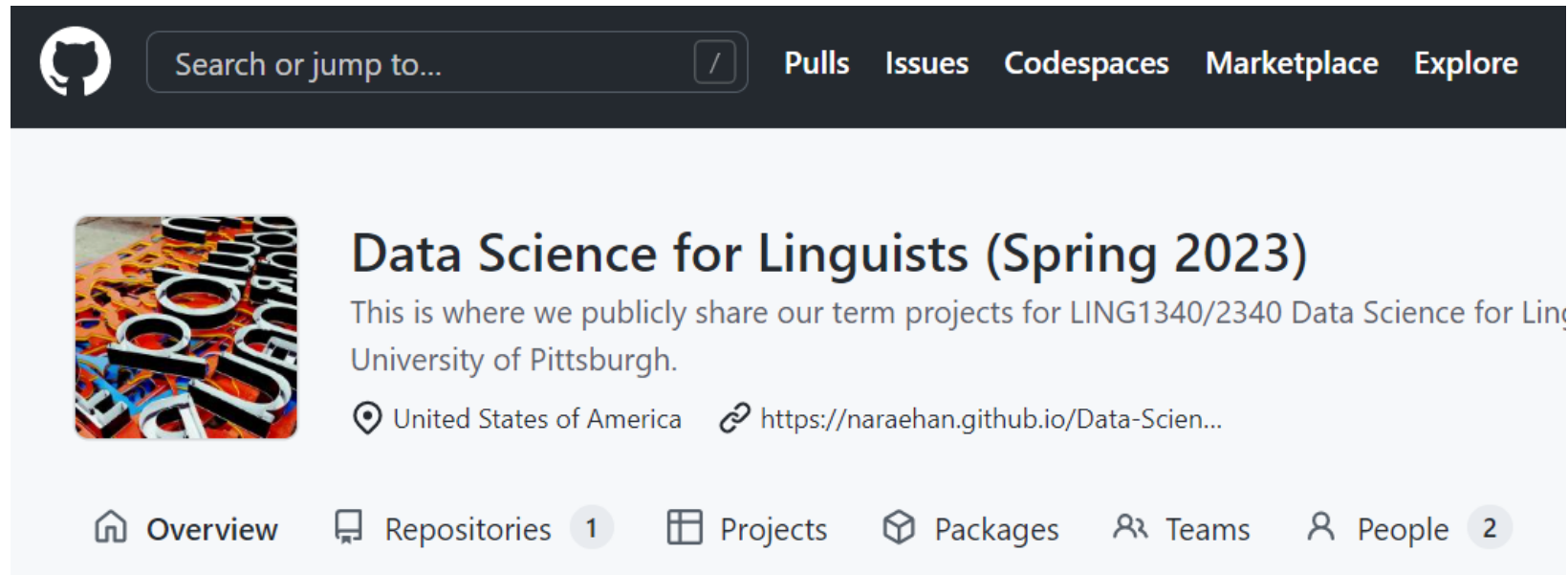
▶ Remote (= GitHub) operations: git push

GitHub: a *social*, remote repository

▶ GitHub also works as a central remote repository among a group of **collaborators** working on a shared project.

◆ Everyone works on their own *local* copy of the repository, making changes.

◆ Git is able to keep track and merge changes submitted by everyone.

central (remote) repository

push & pull

push & pull

push & pull

local repo

local repo
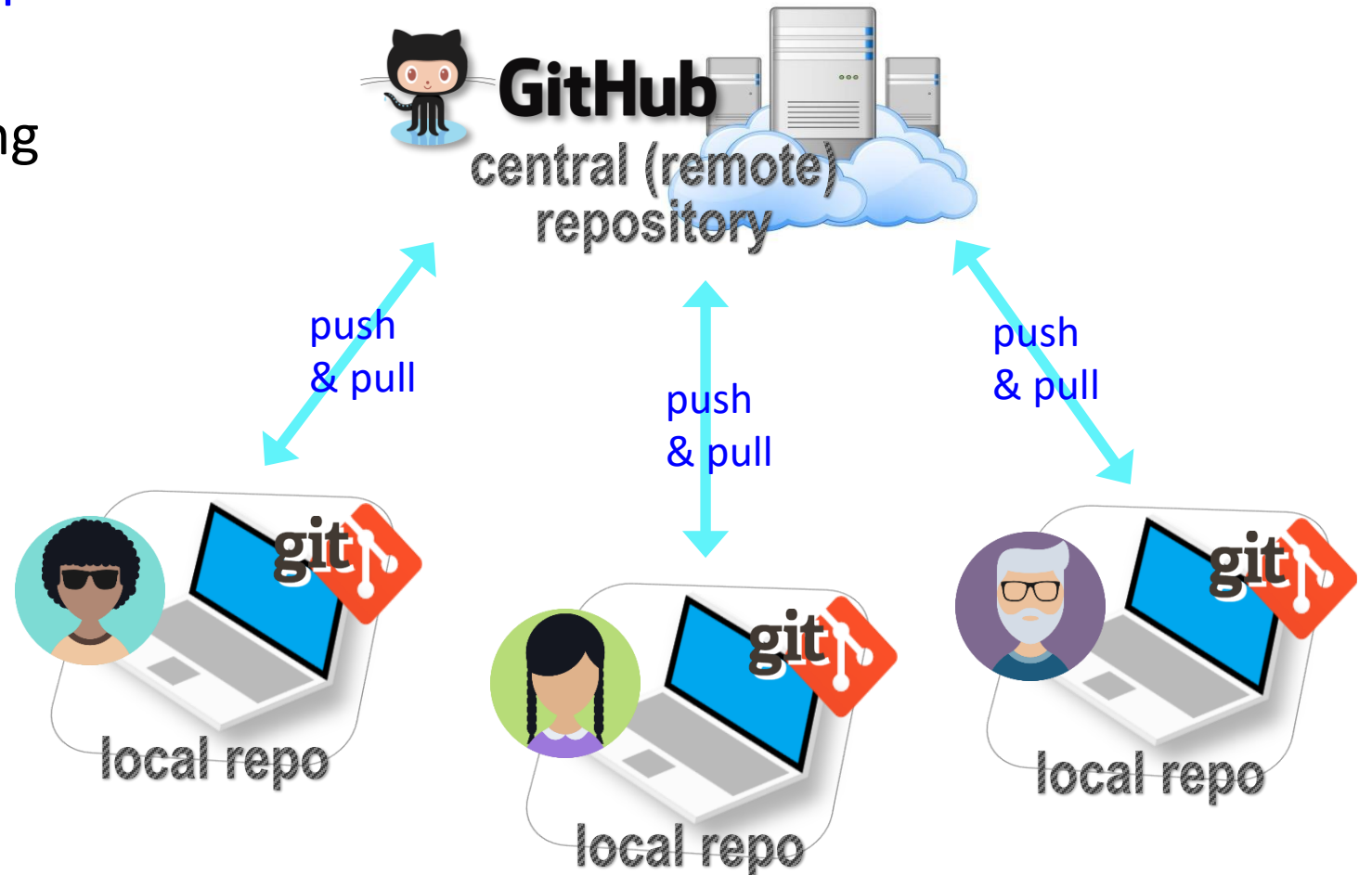
local repo

GitHub: a *social*, remote repository

▶ GitHub also works as a central remote repository among a group of **collaborators** working on a shared project.

◆ Everyone works on their own *local* copy of the repository, making changes.

◆ Git is able to keep track and merge changes submitted by everyone.

◆ Everyone is an **equal collaborator** with push (=write) access.

GitHub
central (remote) repository

push & pull

push & pull

local repo

local repo

local repo

We are not ready to do this just yet!

1/11/2023

15

Introducing... GitHub Class Organization

▸ https://github.com/Data-Science-for-Linguists-2023

Accept invitation from "notifications" page

Click "People", and make your membership public

▸ So we can:

 ◆ have everyone in one spot.
 ◆ have all class materials in one spot.
 ◆ have everyone's term project in one spot.
 ◆ share *private* repos as a group.
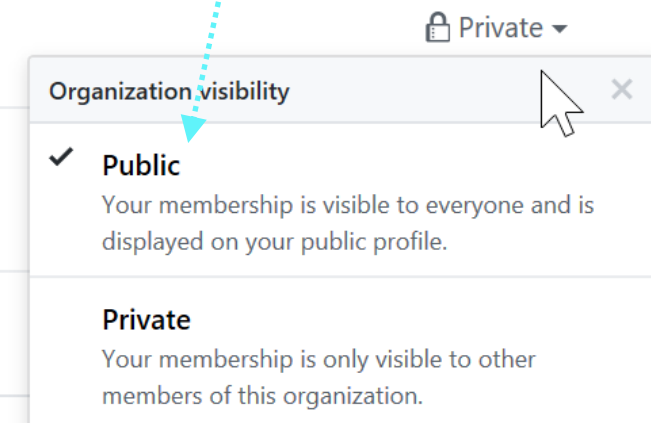
But first, forking
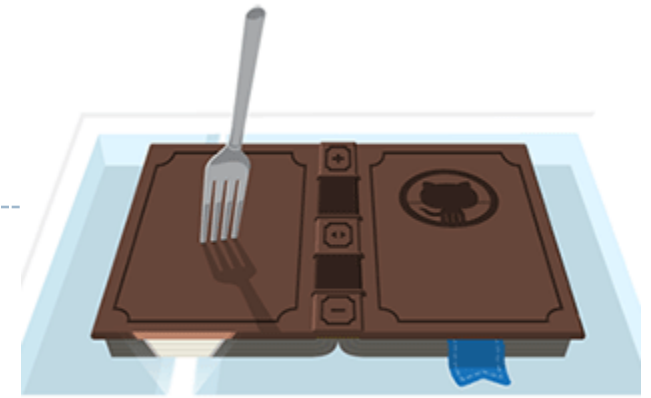
▸ When you **start with someone else's project**.

 ◆ You are *not* a collaborator in their repo. (No push access)

▸ https://help.github.com/articles/fork-a-repo/

▸ You **fork** the original repo into your own GitHub account, creating your own "fork".

▸ You make changes in your own fork. The original repo is not affected!
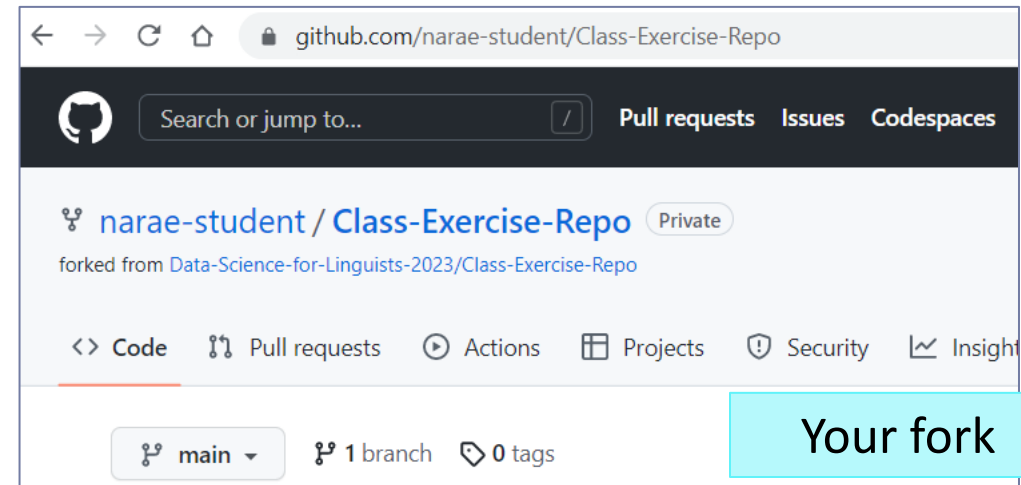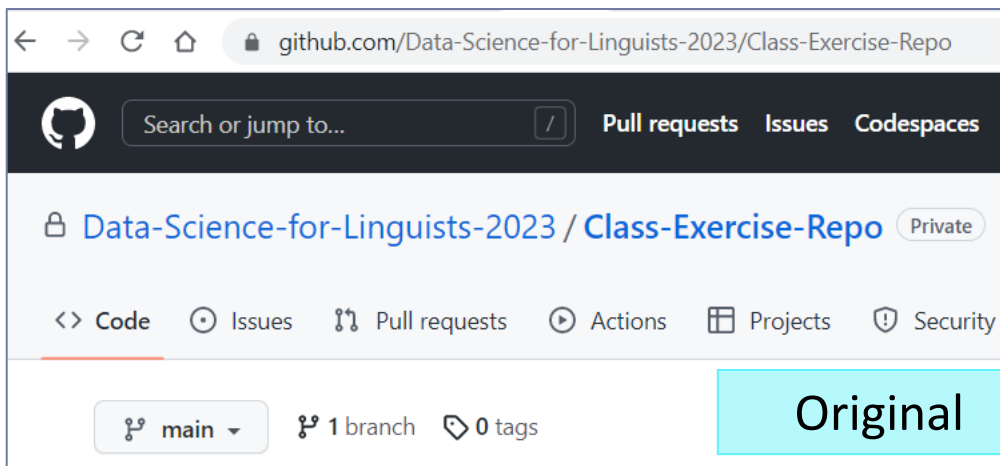
Original

Your fork

Forking, one-way
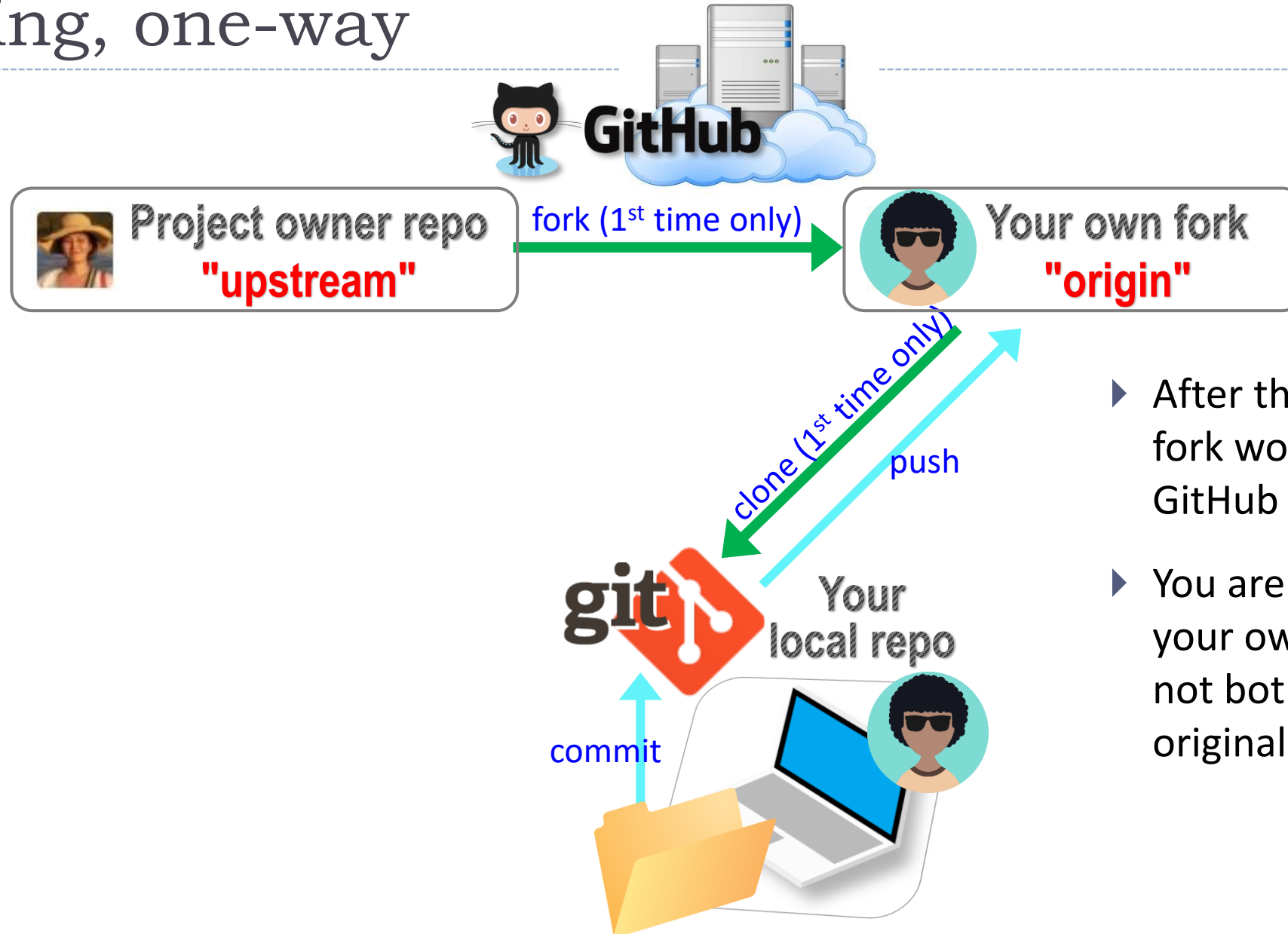
- After the spin-off, your fork works as if your own GitHub repo.

- You are content to do your own development, not bothering the original project owner…

Your first fork

▶ On **GitHub**:

1. Go to class GitHub org.

2. Fork "Class-Exercise-Repo". You will now have the exact same content in your own GitHub account.
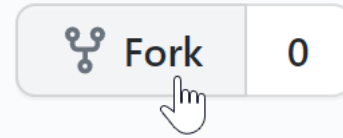
▶ On your **laptop**:

1. Move into your `Data_Science/` directory. Clone your fork there via git clone URL.

2. In the `activity1/` folder, make a new file test_yourname.txt and then add whatever new line you want. Make sure "yourname" is your actual name!

3. Do Git operations: add, commit, and then push to your fork.

▶ Back on **GitHub**:
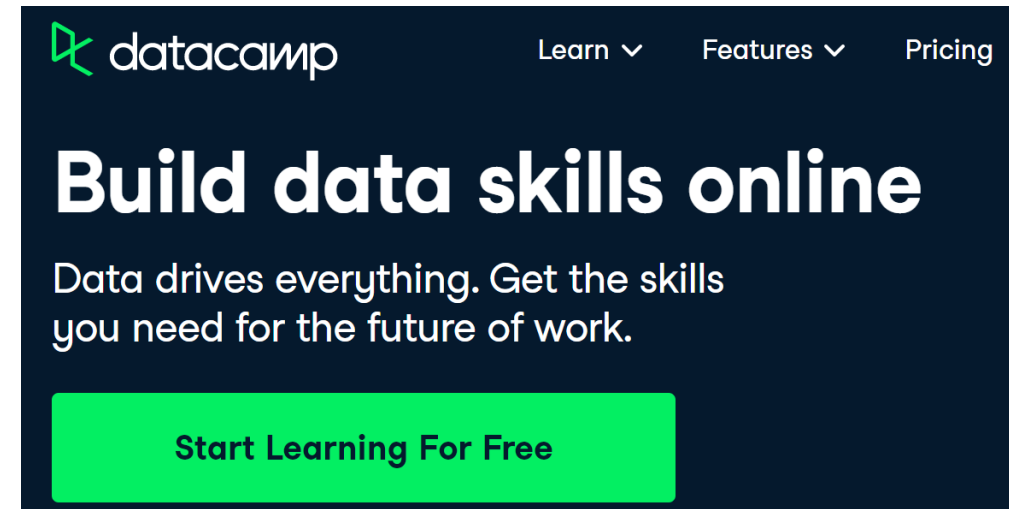
1. Confirm your GitHub fork now has your file.

> If you are familiar with GitHub, you might be itching to press "pull request". Don't! We learn about it on Friday.

Wrapping up

▸ To-do #2 out: explore two linguistic datasets.

▸ Homework #1 is out (due next Wed): process a linguistic dataset of your choice in Python, using Jupyter Notebook

  ◆ Don't be too ambitious! This HW is about taking stock of what you already know and where to go from there. And also new tools.

▸ Office hours
  ◆ This week's hours are posted on Canvas, MS Teams
  ◆ Need help with Git and GitHub set up? Come to our office hours

▸ I will be sending out DataCamp invitation →