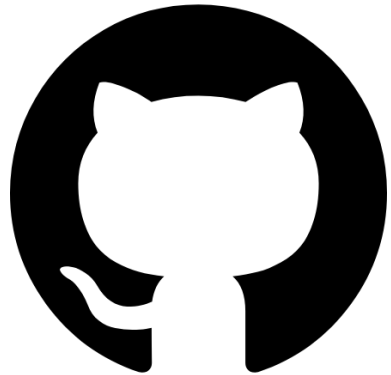
Introduction to GitHub

Cara Marta Messina
NULab Coordinator



Northeastern University
NULab for Texts, Maps, and Networks

Connect to WiFi

Before we begin, I want to make sure everyone is connected to the campus WiFi! If you do not have a MYNEU account and are not part of Northeastern, please follow these steps:

- Turn your WiFi on and click “**NUWave-Guest**”
- Pick the option “**One Day Conference Login**”
- Find this workshop (“**GitHub Workshop**”)
- Use this code: **conf990810**

Brief Introductions

Name

Department/Institutional Affiliation

One-two sentences why you decided to take this workshop

To Do:

Today, we are going to be:

- Creating a GitHub account
- Downloading GitHub Desktop
- Learning particular GitHub terminology
- Discussing best practices for collaboration and sharing
- Discussing briefly naming conventions and folder organization
- Working with an actual GitHub repository

What is GitHub?

GitHub is a version-control repository system designed for software developers, programmers, coders, and anyone working with coding to collaborate and share their work.

The repositories are stored in a “cloud” of sorts, or a server that is outside of your local computer. You can, however, connect your local computer to this “cloud” repository.

GitHub uses **Git**, a language that keeps track of version control, and has created a user-friendly web browser for all levels of learners.

GitHub Repository Examples

GitHub repositories can be stored under one particular user or an organizations.

Example of user repository:
https://github.com/radiolarian/AO3Scraper

Example of organization repository:
https://github.com/NULabNortheastern/digitalassignmentshowcase

Example of repository created for a workshop:
https://github.com/lknelson/text-analysis-2016

Creating a GitHub Account

If you already have a GitHub account, you can simply sign into your GitHub account

- **Sign up:** Got to https://github.com and click “Sign Up”
- **University Email:** If you have an university/organization-affiliated email, I recommend using this
- **Username:** Choose a username that you want to associate with your professional identity

Important things to remember

- Always read the “Read Me” file. It will usually provide some explanation about what this repository is and how to use it (the readme.md file is a “Markdown” file, which is a formatted text file and will show up on the homepage of a repository)
- GitHub repositories are **folders** and **files**. Navigating them is similar to navigating folders and files on your own computer
- Since GitHub is about version control, you can always look at different **versions** of a file and see what changes have been made

Downloading GitHub Desktop

GitHub Desktop connects your **local computer** to a GitHub repository. This way, you can work on your local computer and then make changes.

Download: https://desktop.github.com/

As you're installing GitHub Desktop...

Make sure to login to GitHub Desktop with your GitHub username

As GitHub Desktop is downloading and you're logging in, please fill out the survey with your GitHub Username so I may add you to the workshop GitHub repository: **https://bit.ly/2zwFlLK**

Collaborators

On a repository, there are:

- the ***creator*** of the repository, the user who originally created and whose username is storing the repository
- the ***collaborators***, those who can make changes on the respiratory (add collaborators by going to the “Settings’ in a repository, clicking “Collaborators” and adding by username. Users must then **accept** the invitation to collaborate)

You should all now be “collaborators” in the repository “**introtogithub**” created under my username, **caramessina**.

Cloning a Repository

Cloning means you are copying a GitHub repository and saving it on your computer. This means you can make local changes to a repository on your computer. You can do this whether you are a "collaborator" on a repository or not (if you are not a collaborator, you cannot make changes to the original repository).

It's important to use the cloning feature if you are borrowing someone's code because cloning **tracks who uses whose code** and provides a system for **giving credit**.

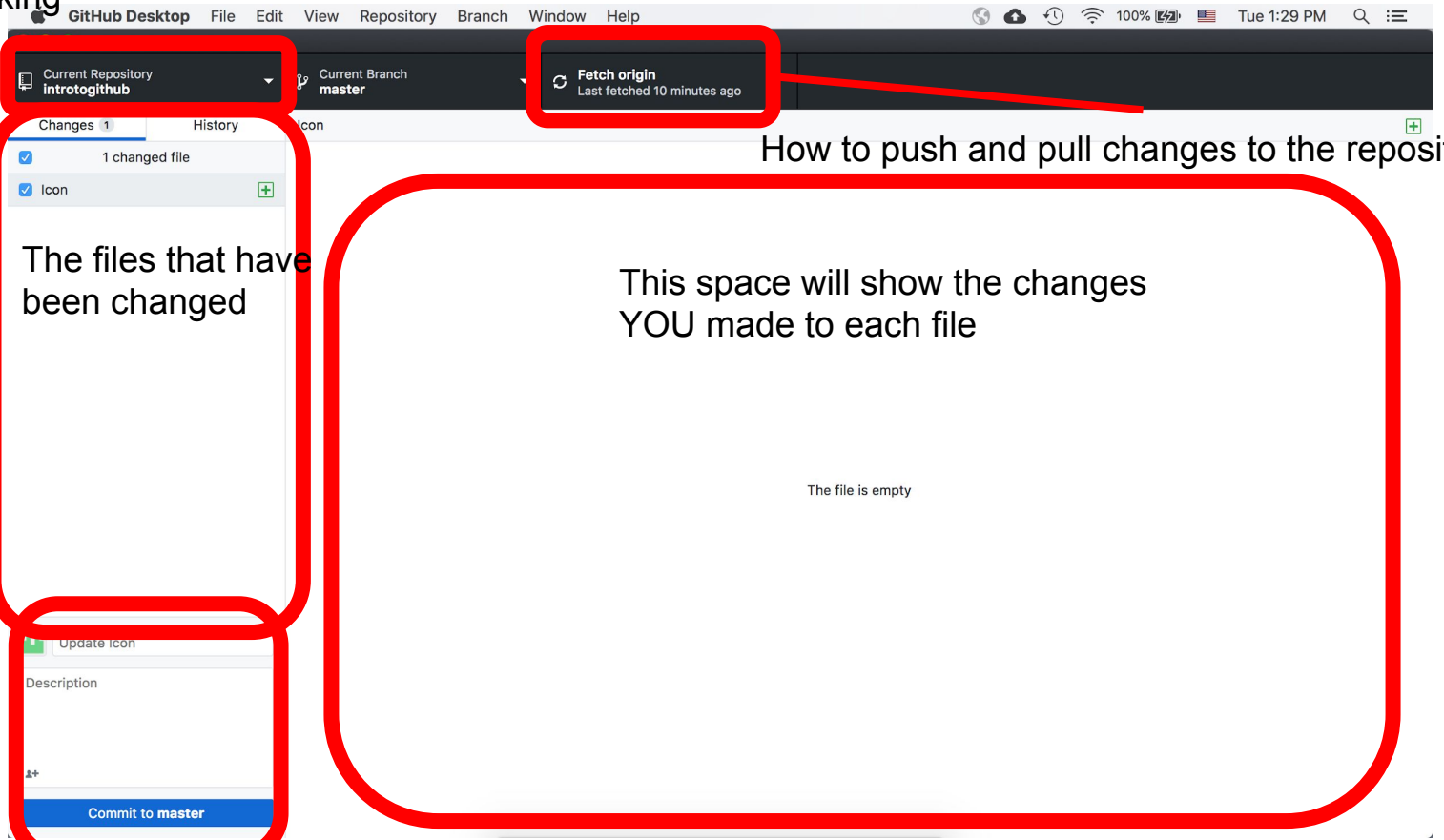
GitHub Desktop: Clone a Repository

In order to clone a repository using GitHub desktop:

- Open GitHub Desktop
- Click “File” in the top menu and “Clone Repository”
- Click “URL” and enter the URL of the repository you want to clone. Our workshop repository URL is: https://github.com/caramessina/introtogithub
- Choose a “Local Path” (your computer’s file structure) that you want to save this repository

Anatomy of GitHub Desktop

The repository with which you're working



How you describe and commit your changes

Naming conventions and file structure

Before we get into playing with the interface, I want to briefly discuss best practices for naming conventions and file structures!

As a group, you should always decide on how your folders will be organized thinking about a) making it easy to navigate for yourselves and b) making it easy for outsiders to navigate

Basic practices:

- Don’t use spaces in your file names!
- Use numbers to help organize your files and folders
- Choose names that make the content in each folder/file clear
- Provide a guide in the “Read Me” if necessary
- Decide among your group what the naming conventions will be so everyone follows them

Examples of good naming practices

Laura Nelson's "Text Analysis" Workshop:

https://github.com/lknelson/text-analysis-2016

From a folder containing template flyers:

M... > Nort... > ... > NULa... > 00.TEMPL... > p... > templat...

Name	Owner	Last modified	File size
speaker-event-onespeaker.pptx	me	12:07 PM me	297 KB
speaker-event-onespeakerExample1.pptx	me	12:12 PM me	217 KB
speaker-event-twospeakers.pptx	me	12:21 PM me	378 KB
speaker-event-twospeakersExample.pptx	me	12:18 PM me	1,012 KB
workshop-flyer-basic.pptx	me	12:25 PM me	449 KB
workshop-flyer-basicExample.pptx	me	12:25 PM me	469 KB
workshop-flyer-twoday.pptx	me	12:33 PM me	876 KB
workshop-flyer-twodayExample.pptx	me	12:33 PM me	1 MB
workshop-flyer-visual.pptx	me	12:32 PM me	864 KB
workshop-flyer-visualExample.pptx	me	12:33 PM me	916 KB

Pushing, Pulling, Committing

Pull means you are downloading changes from the repository to your local computer

Commit means you are choosing the files which you have changed to commit those changes on your local computer to be “pushed” onto the repository

Push means you are uploading changes from your local computer to the repository

Steps to Pulling, Committing, and Pushing

1. **Pull** from the repository in case changes have been made by other collaborators
2. Make your changes, add new files, whatever you are doing
3. **Commit** your changes on GitHub Desktop
4. **Push** your committed changes from your local computer to the “cloud” repository

Practice

Let’s say I want to make changes. Let’s go through the steps together, first on my computer and then on yours!

Activity: Practicing pulling and pushing

Get into groups of 2-3; each group will get a number. Choose **one person's laptop** to work on and follow these steps:

1. Pull from the repository to your local computer ("Fetch Origin")
2. Go to the repository in your finder/local computer file (Repository > "Show in Finder" on Macs)
3. Create new folder called "group#" (insert your group's number)
4. Create a .txt document in your group's folder, put your group members' first names in the file, and save it
5. Commit changes to the file (make sure to "unclick" DS_store)
6. Push the changes!

Uh-oh…CONFLICTS!

So what happens when multiple people are working on the same files at the same time, try to push those changes, and cause conflicts?

Well, this happens! GitHub Desktop is smart enough to know there are conflicts and will tell you. Sometimes, unfortunately, you just need to go through the files and fix those conflicts.

Sample Conflict

In the folder “group0,” there is a file called “samplefile1.txt”. This shows you a sample conflict!

How to fix a conflict

- Open up the file (click on the file in GitHub Desktop and “Reveal in Finder”)
- Find anytime “**<<<<<<<<<HEAD**” appears. This is the **start** of the conflict
- The code/text above “**=======**” is the changes on **your** local file
- The code/text below “**=======**” is the changes made on the repsitory
- Anytime “>>>>>>>>” and a series of numbers+letters appears, this is the **end** of a conflict
- To fix, remove the signifiers of the conflict and make changes based on what needs to stay and what should go. I also highly recommend **checking in with your team members about this conflict.**

Activity: Practice Fixing a Conflict

Get back into your groups.

- Use **two separate laptops** (not three). We're going to try to create conflicts!
- **Pull:** Make sure to "pull" first so that you have all the updated information on your local computer
- On the file that you and your group created, write any color of the rainbow between each person's name. Example: cara yellow sarah
- Once you have finished, "commit" your change and "push" to the repository.
- Once the group has all finished, work on **one person's laptop** and check out the final result of your conflicts! Discuss with your group how to get rid of these conflicts and how you will as a group handle each individual's choice for different colors.

Best Practices for Avoiding Conflicts

- **Designate specific roles:** Assign particular tasks, files, and folders to particular team members. I also recommend choosing one person to make the final decision about the master branch.
- **Work on branches:** Create separate “branches” as you work on the repository. With branches, you fork a repository, taking all its content, and make changes. You then save your changes on your individual branch and once you’ve gotten approval from your team, can merge your changes from your individual branch to the master branch.
- **Consent**: If you would like to work on a file that another team member has created or has been working on, check in with them first! Working on separate branches also encouraged a culture of consent.

Branches

The **Master Branch** is the sort of “homepage” branch that you see when you go to a GitHub repository. However, a repository might have multiple branches with unique changes.

If there are multiple collaborators working on one repository, I recommend having them work on their individual branches to a) avoid conflicts and b) practice consent when making changes to your team’s code.

Merging Branches

New Branches: GitHub Desktop, click “Branch” > “New Branch.” Make the branch your name so it’s easy to identify whose branch is whose. Now you have a branch to work on!

Before you start working:

1. Make sure you are working in **your branch** (Between repository name and “Fetch Origin” on GitHub Desktop)
2. Merge the master branch into your own branch (Branch>Merge Into Current Branch and make sure you are merging the master branch into your branch, not your branch into the master)
3. Start your work! Make your changs, commit, and push to your branch!

Final Thoughts

- GitHub's infrastructure prioritizes consent, credit, and collaboration. Take advantage of this and use it as an
- Before you begin working on a project, designate roles, discuss naming conventions and the file structure, and come up with clear guidelines for collaboration will encourage a healthy and respectful working environment.
- Any questions?

Cara Marta Messina
messina.c@husky.neu.edu
@cara_messina
caramartamessina.com