

Introduction to GitHub, GitHub Desktop, and Pages



Dr. Cara Marta Messina
Assistant Prof of English
Jacksonville State University

<https://caramartamessina.com/>



Brief Introductions

- Name
- Pronouns if you wish to share
- Research interests



Pre-Workshop

- Created GitHub account
- Sent GitHub username to meeting agenda
- Downloaded GitHub Desktop
- Watched the introduction videos

Workshop Agenda

1

Introduction to GitHub

- Define GitHub & important terminology
- Introduce GitHub Desktop

2

Workflow Best Practices

- Creating repositories
- Introducing GitHub workflow
- Naming conventions

3

Repository Activity

- Review GitHub Desktop
- Collaborate on a repository

4

GitHub Pages

- Brief introduction
- Building websites with GitHub Pages

Workshop Agenda

1

Introduction to GitHub

- Define GitHub & important terminology
- Introduce GitHub Desktop

2

Workflow Best Practices

- Creating repositories
- Introducing GitHub workflow
- Naming conventions

3

Repository Activity

- Review GitHub Desktop
- Collaborate on a repository

4

GitHub Pages

- Brief introduction
- Building websites with GitHub Pages

Introduction to GitHub

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.

GitHub is also used to **publish** and **share** their code; repositories can be public or private.

Finally, GitHub Pages offers a **hosting method** to create and publish websites.



So, what can you use GitHub for?

- Sharing and publishing code
- Version control for applications and codes; works well with other tools to implement updates.
- Sharing and publishing any files you want to save and store
- Providing feedback & issues on codes and applications
- Building collaborative projects
- Hosting websites through GitHub pages



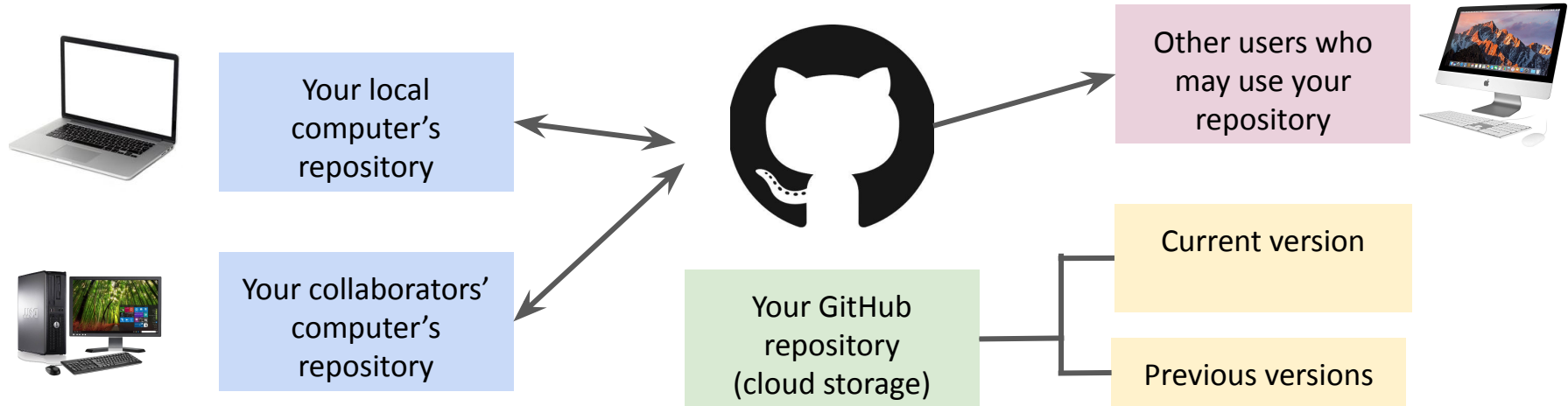
Best Part of Github?

It's FREE!

How does GitHub work?

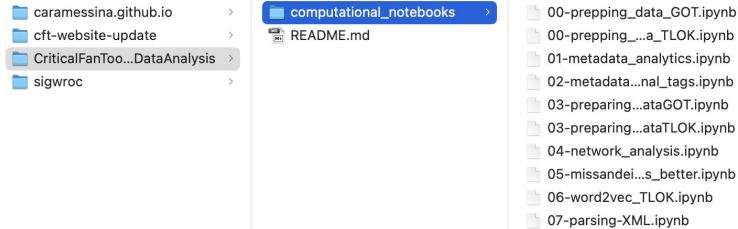
GitHub repositories are stored in the “cloud,” or servers that are outside of your local computer. You can connect your local computer to this cloud repository and, if your code is public, others can clone your repository and use your code!

GitHub uses **Git**, a version-control language provides previous versions of your code, and has a user-friendly web browser for all levels of learners.

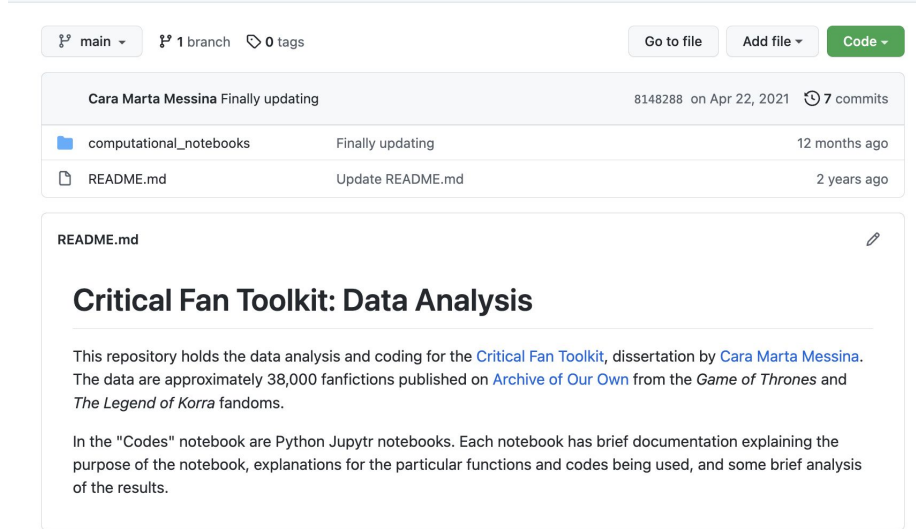


Local Folder VS GitHub Repository

GitHub repositories will match my local folder and file structures on my computer.



Local folders & files



GitHub Repository

https://github.com/caramessina/CriticalFanToolkit_DataAnalysis

GitHub Repository Examples

GitHub repositories can be stored under one particular user or an organizations. Here are some examples:

- [Code-sharing repository](#)
- [Organization repository](#)
- [Workshop repository](#)
- [Website repository \(GitHub Pages\)](#)

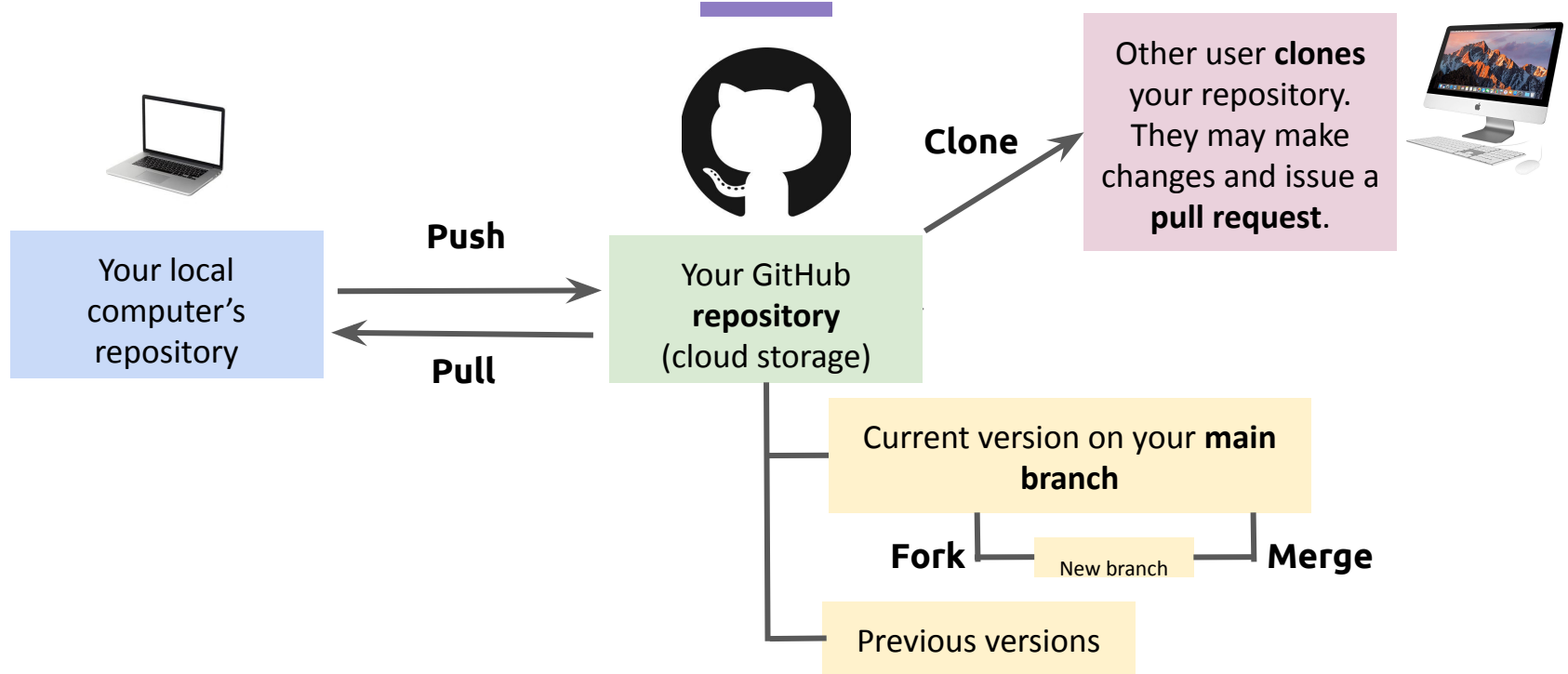
Important Software & Language

- **Git:** The software that tracks version control and repositories. Git is usually used on local computers to save different versions of a code.
- **GitHub:** Online service that uses git so developers can upload and share their code.
- **GitHub Desktop:** A graphical user interface software that uses your GitHub account to connect GitHub repositories on the cloud to your local computer.
- **GitHub Pages:** Repositories that hold your website documents (such as your HTML5, CSS, and Javascript docs), translate them, and publish them! Pretty much a hosting service for websites.

Important Git & GitHub Terminology

- **Repository:** Contains all of your project's files and each file's revision history.
- **Clone:** Copy a repository to your local computer.
- **Pull:** Once you have cloned the repository, download content from it to your local computer.
- **Push:** Upload content from your local computer to your GitHub repository.
- **Main branch:** Your main repository.
- **Fork/new branches:** Different versions of the main branch that are made, but do not yet affect the master branch.
- **Pull Request:** Someone downloads content, makes changes, and pushes new content with an explanation for what happened. This does not change the “Master branch” but instead could spark a discussion about changes.

GitHub Terminology in Action

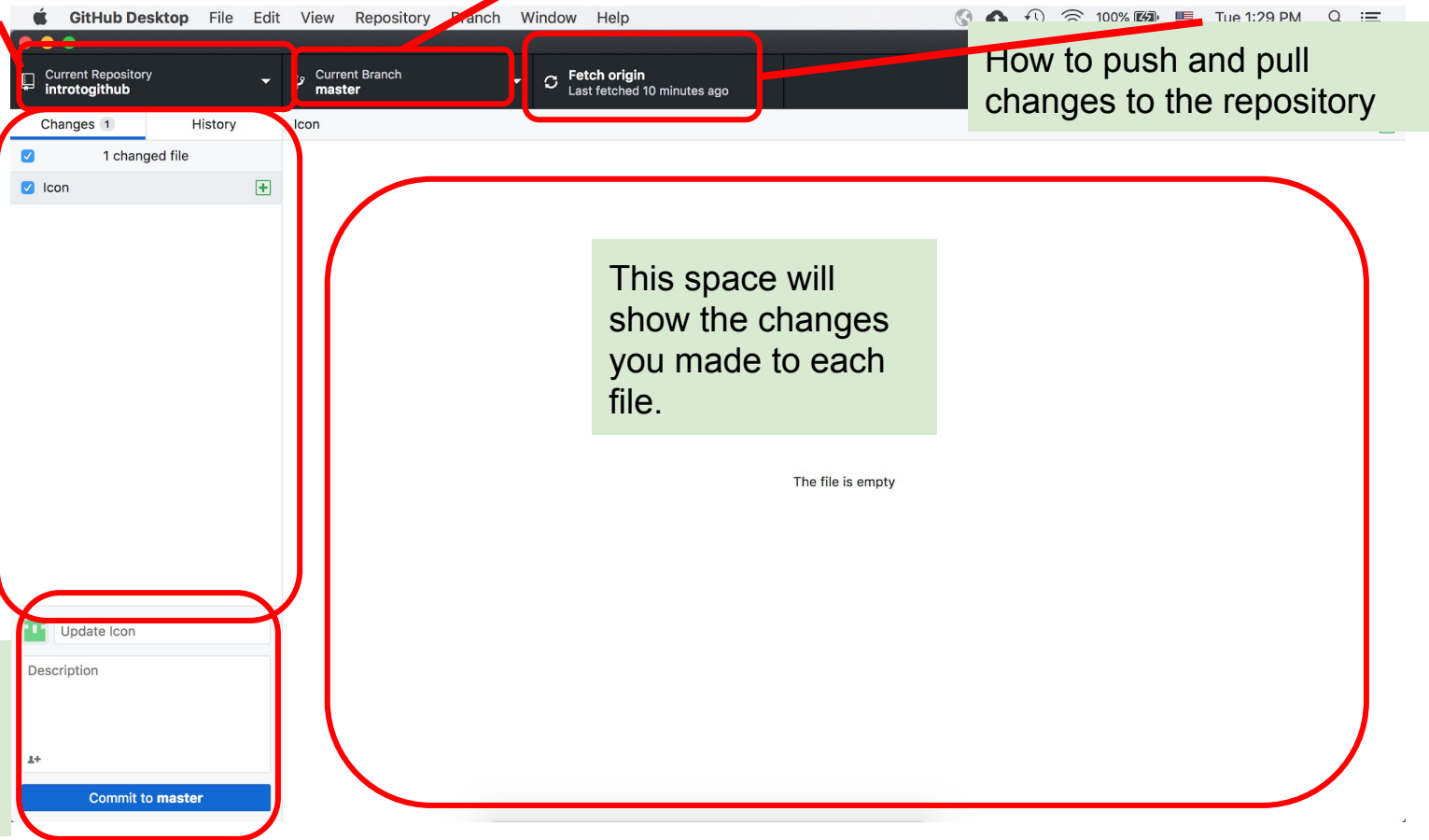


GitHub Desktop

Make sure you're signed into GitHub Desktop
with your GitHub account!

The repository
with which
you're working

The branch you're working on



How to push and pull
changes to the repository

The files
that have
been
changed

This space will
show the changes
you made to each
file.

How you
describe and
commit your
changes

Anatomy of GitHub Desktop

GitHub Desktop VS. Command Line

You can interact with your GitHub repository on your local computer using GitHub Desktop or the command line (your terminal).

If you are interested in using your terminal, here is a [GitHub and git cheatsheet](#).

Workshop Agenda

1

Introduction to GitHub

- Define GitHub & important terminology
- Introduce GitHub Desktop

2

Workflow Best Practices

- Creating repositories
- Introducing GitHub workflow
- Naming conventions

3

Repository Activity

- Review GitHub Desktop
- Collaborate on a repository

4

GitHub Pages

- Brief introduction
- Building websites with GitHub Pages

Creating Repositories

- Create on GitHub web browser, GitHub Desktop, through the command line, or first on your local computer.
- Choose clear and memorable repository name (no spaces).
- Add README file for documentation purposes.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *



caramessina ▾

Repository name *

CLIRworkshop2022 ✓

Great repository names are short and memorable. Need inspiration? How about [super-duper-octo-robot](#)?

Description (optional)



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

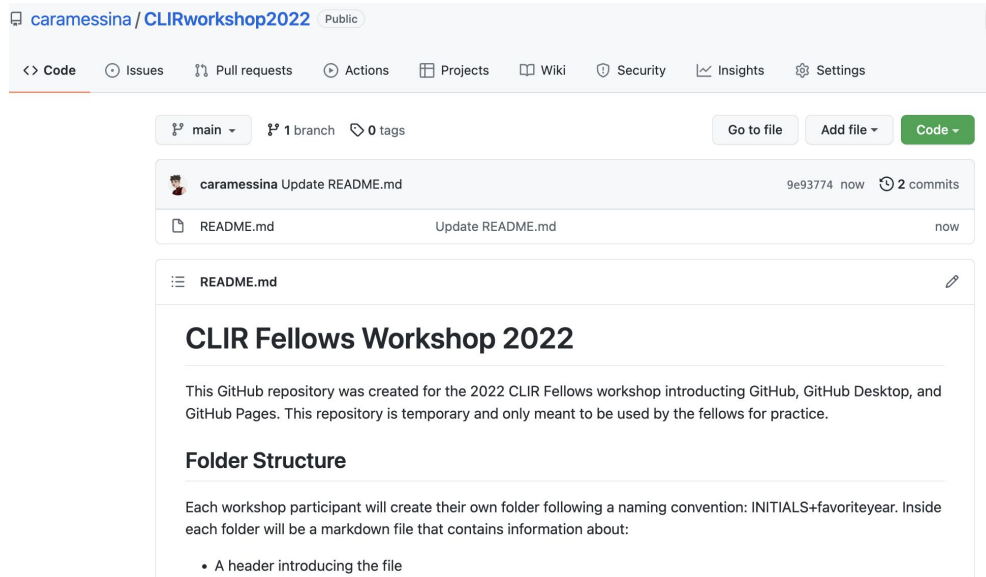
Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: None ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

New Repository!

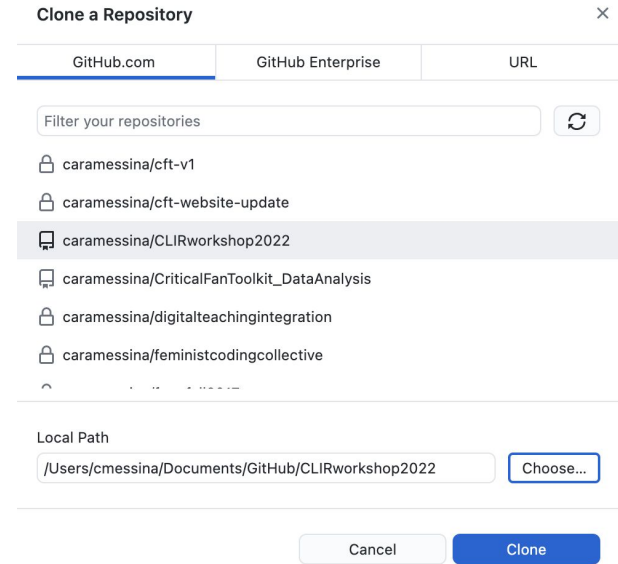


The screenshot shows the GitHub interface for a new repository named 'CLIRworkshop2022' by user 'caramessina'. The repository is public. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, there are buttons for 'Go to file', 'Add file', and 'Code'. The repository's main branch is 'main', and it has 1 branch and 0 tags. A commit history table shows a recent commit by 'caramessina' titled 'Update README.md' with commit hash '9e93774' and '2 commits'. Below the commit history, the 'README.md' file is displayed, showing the title 'CLIR Fellows Workshop 2022' and a description: 'This GitHub repository was created for the 2022 CLIR Fellows workshop introducing GitHub, GitHub Desktop, and GitHub Pages. This repository is temporary and only meant to be used by the fellows for practice.' The 'Folder Structure' section states: 'Each workshop participant will create their own folder following a naming convention: INITIALS+favoriteyear. Inside each folder will be a markdown file that contains information about:' followed by a bullet point: '• A header introducing the file'.

- Edit the README.md (markdown) file directly in web browser or on your local computer.
- Connect the repository to your local computer.
- Start working on your repository!

Connecting A GitHub Repo to Your Local Computer

- Open GitHub Desktop
- Make sure you're signed into your GitHub account
- Click "File" > "Clone Repository"
- Make sure you're in "GitHub.com"
- Click your repository name
- Ensure the local file path works for you
- Click "Clone," then "Find Origin" and "Pull"



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**.

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

GitHub Workflow

Best Practices

GitHub Workflow

- Varies if you're working individually or with a team
- Make sure you "pull" changes before you begin working to avoid conflicts!
- Mostly you will be working on your local computer and "pushing" any changes
- Forking a branch from the main repository will allow you to make changes without directly changing the main repository.

Workflow for GitHub

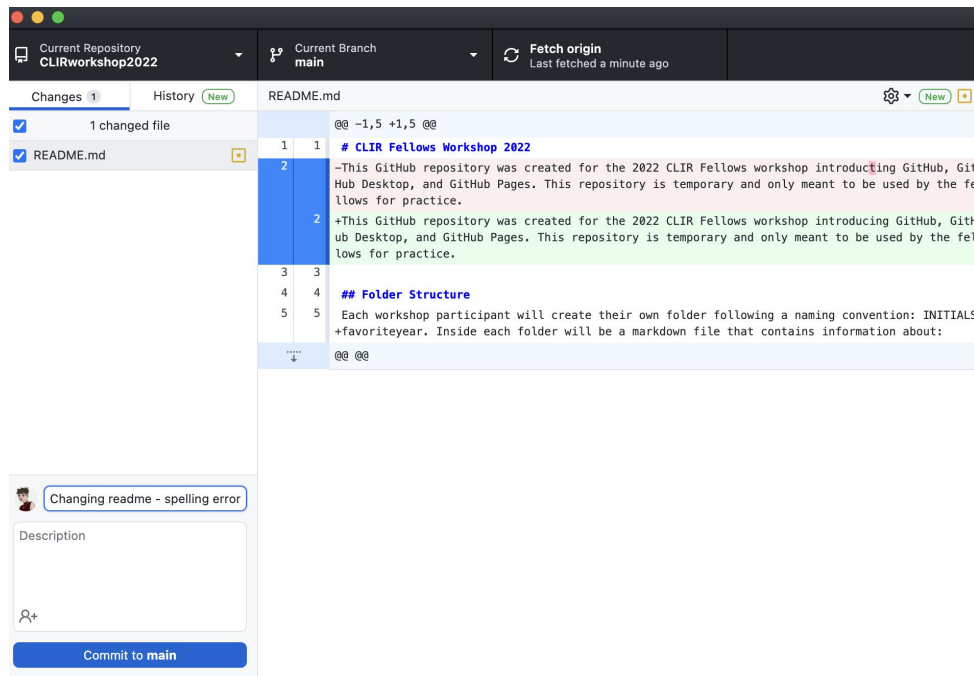
Pulling to stay up-to-date

1. Open GitHub Desktop
2. Click “Fetch Origin”
3. If changes were made to the GitHub repository, “Pull Origin” will appear. Click this to ensure you have the most up-to-date version.

Pushing your changes

1. Your changes will appear on GitHub Desktop when you’re done.
2. Write a brief “Summary” for version control purposes.
3. Click “Commit to main”
4. Click “Push Origin”

How Your Changes Look on GitHub Desktop

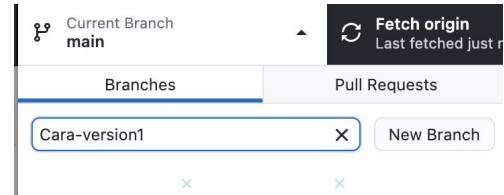


Forking: Creating a New Branch

When you're working with a team or on an application that directly pulls from your GitHub repository, you want to fork a new branch to make changes. This way your team can approve changes and/or you don't accidentally break the application while you're working!

In GitHub Desktop:

1. Click "Current Branch"
2. Click "New Branch" and write your name
3. Create Branch and work!
4. Once you're happy with the results, let your team know so they can review your work.
5. Merge your branch to the main branch.



Merging Branches

Before you start working on your new branch:

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 changes, commit, and push to your branch!

Naming Conventions

Best Practices

Naming conventions and file structure





















As a team, 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

- Don't use spaces in your file names!
- Use numbers to help organize your files and folders (ex: 01 will come before 02)
- Choose names that make the content in each folder/file clear
- Provide a guide in the "Read Me" if necessary
- If you are collaborating, 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.](#)
- 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

Workshop Agenda

1

Introduction to GitHub

- Define GitHub & important terminology
- Introduce GitHub Desktop

2

Workflow Best Practices

- Creating repositories
- Introducing GitHub workflow
- Naming conventions

3

Repository Activity

- Review GitHub Desktop
- Collaborate on a repository

4

GitHub Pages

- Brief introduction
- Building websites with GitHub Pages

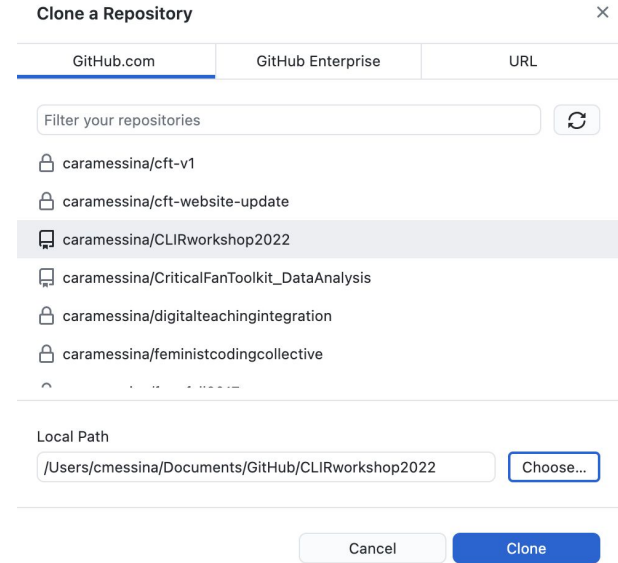
Collaborators

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 repository.
 - Add collaborators by going to the “Settings” in a repository
 - Click “Collaborators” and “Add people”
 - Add by username, email, or full name
 - Users must then **accept** the invitation to collaborate – AKA please accept the invitation to the CLIR Fellows GitHub repo!

Connect the “CLIRworkshop2023” Repo to GitHub Desktop

- Open GitHub Desktop
- Make sure you’re signed into your GitHub account
- Click “File” > “Clone Repository”
- Go to “URL” and enter in the URL for the repository:
<https://github.com/caramessina/CLIRworkshop2023>
- Ensure the local file path works for you
- Click “Clone”



Activity: Let's Practice Practice

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

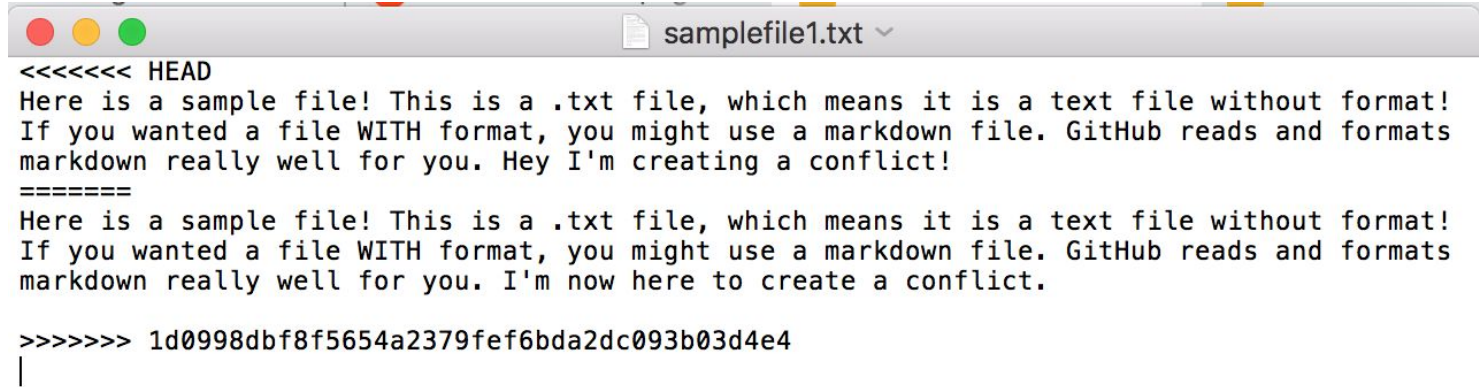
Activity: Practicing pulling and pushing

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 that follows our naming convention: YOURINITIALSfavoriteyear. (ex: CMM2012)
4. Create a .txt document in your folder. In this file, put:
 - a. Your first name (or whatever name you prefer to share)
 - b. Your favorite color
 - c. How you may use GitHub
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



```
<<<<<<< HEAD
Here is a sample file! This is a .txt file, which means it is a text file without format!
If you wanted a file WITH format, you might use a markdown file. GitHub reads and formats
markdown really well for you. Hey I'm creating a conflict!
=====
Here is a sample file! This is a .txt file, which means it is a text file without format!
If you wanted a file WITH format, you might use a markdown file. GitHub reads and formats
markdown really well for you. I'm now here to create a conflict.

>>>>>>> 1d0998dbf8f5654a2379fef6bda2dc093b03d4e4
|
```

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 repository
- 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

Can I get a volunteer 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 created, write any color of the rainbow after your favorite color.
- Once you have finished, “commit” your change and “push” to the repository.
- I will do the same thing, and we will create a conflict and solve it together!

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.

Workshop Agenda

1

Introduction to GitHub

- Define GitHub & important terminology
- Introduce GitHub Desktop

2

Workflow Best Practices

- Creating repositories
- Introducing GitHub workflow
- Naming conventions

3

Repository Activity

- Review GitHub Desktop
- Collaborate on a repository

4

GitHub Pages

- Brief introduction
- Building websites with GitHub Pages

GitHub Pages

Create a repository using “GitHub Pages” that takes your files and folders and makes them into a website for you! In a sense, **Github can also host your website!**

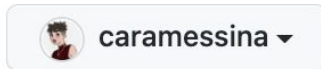
Example:

<https://github.com/caramessina/caramessina.github.io>

Creating a GitHub Pages Repository

Creating a GitHub Pages repository and connecting it to your local computer is pretty much the same workflow as a regular repository, EXCEPT your repository name should be “<name>.github.io”.

Owner *



Repository name *



Great repository names are cmmtesting.github.io is available. Inspiration? How about **fantastic-barnacle**?

Description (optional)

Making Your Repo a Website

Right now, your repository is *just* a repository. To make it into Pages, or the website, follow these instructions:

- Go to “Settings”
- Under “Code and automation” click “Pages”
- Under “Source” click the “None” drop down menu and choose your main branch.
- You can even choose a theme for your website! You will have to download it, mov it to your local file connected to GitHub, and make changes there.
- You can connect domain names to your GitHub pages, too.

Example: My Website

- GitHub Repository:
<https://github.com/caramessina/caramessina.github.io>
- How website looks: <https://caramessina.github.io/>
- Domain name connected to the pages:
<https://caramartamessina.com/>

Resources to Spruce Up Your Site

- [Jekyll themes](#): you can download a “theme” directly from GitHub.
- [HTML5 UP](#): download pretty HTML5 themes that you can edit to make as your own.
- You can also create your own website from scratch!

Final Thoughts

- GitHub's infrastructure prioritizes consent, credit, and collaboration. Take advantage of this!
- 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.

A horizontal purple bar with a small rectangular notch on its left side, positioned to the left of the 'Thank You!' text.

Thank You!

Dr. Cara Marta Messina

cmessina@jsu.edu

@cara_messina

caramartamessina.com