



# Git and GitHub In-depth

# Working with Git and GitHub

## Mohal Khandelwal

- Research Computing
- Website: [www.rc.colorado.edu](http://www.rc.colorado.edu)
- Helpdesk: [rc-help@colorado.edu](mailto:rc-help@colorado.edu)

## Matthew Murray

- Center for Research Data & Digital Scholarship (University Libraries)
- Website: [colorado.edu/crdds/](http://colorado.edu/crdds/)
- Helpdesk: [crdds@colorado.edu](mailto:crdds@colorado.edu)

Slides:

[https://github.com/ResearchComputing/git\\_github\\_in\\_depth\\_short\\_course](https://github.com/ResearchComputing/git_github_in_depth_short_course)

Survey: <http://tinyurl.com/curc-survey18>

# Goals

- Convince you that basic Git/GitHub fluency is:
  - Easy
  - Practical
  - An extremely important tool in your tool belt!



# Learning Goals

- Know the differences between Git and GitHub
- Understand the basics of version control
- Learn basic Git and GitHub fluency
  - Creating a repo, add, commit, pull, clone, push
- Collaboration in GitHub



# Outline

- What is version control?
- Brief overview of Git and GitHub
- When not to use GitHub
- Creating your own repository locally
- Pushing local changes to GitHub
- Documentation

# What is version control?

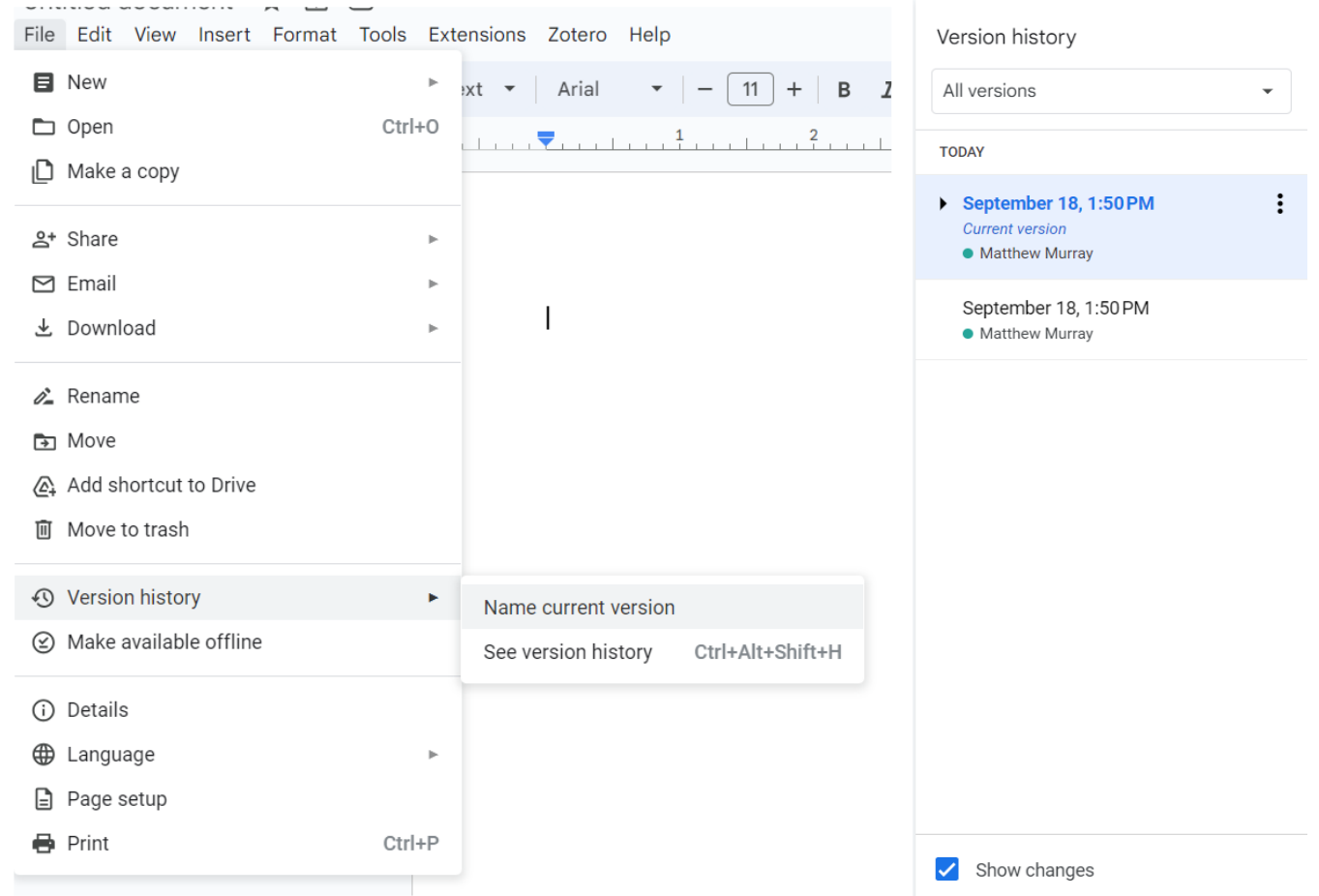
Version control is the practice of tracking and managing changes to files.

- Why do I need it?
  - Revert to various states of files
    - You can think of this as a backup
  - Allows you to modify items without harming the original copy
  - Not limited to code
    - Can be used for documents, images, etc.



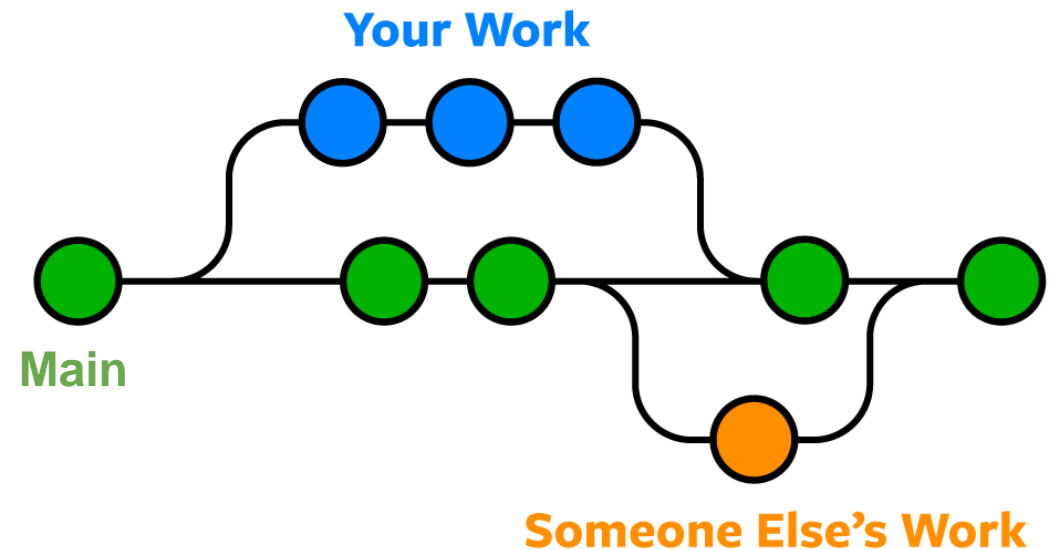
# What is version control?

- Google Docs includes "Version History"
- This allows you to see what changes were made, when those changes happened, and who made them
- You can also revert to a previous version of your file



# Additional benefits of version control

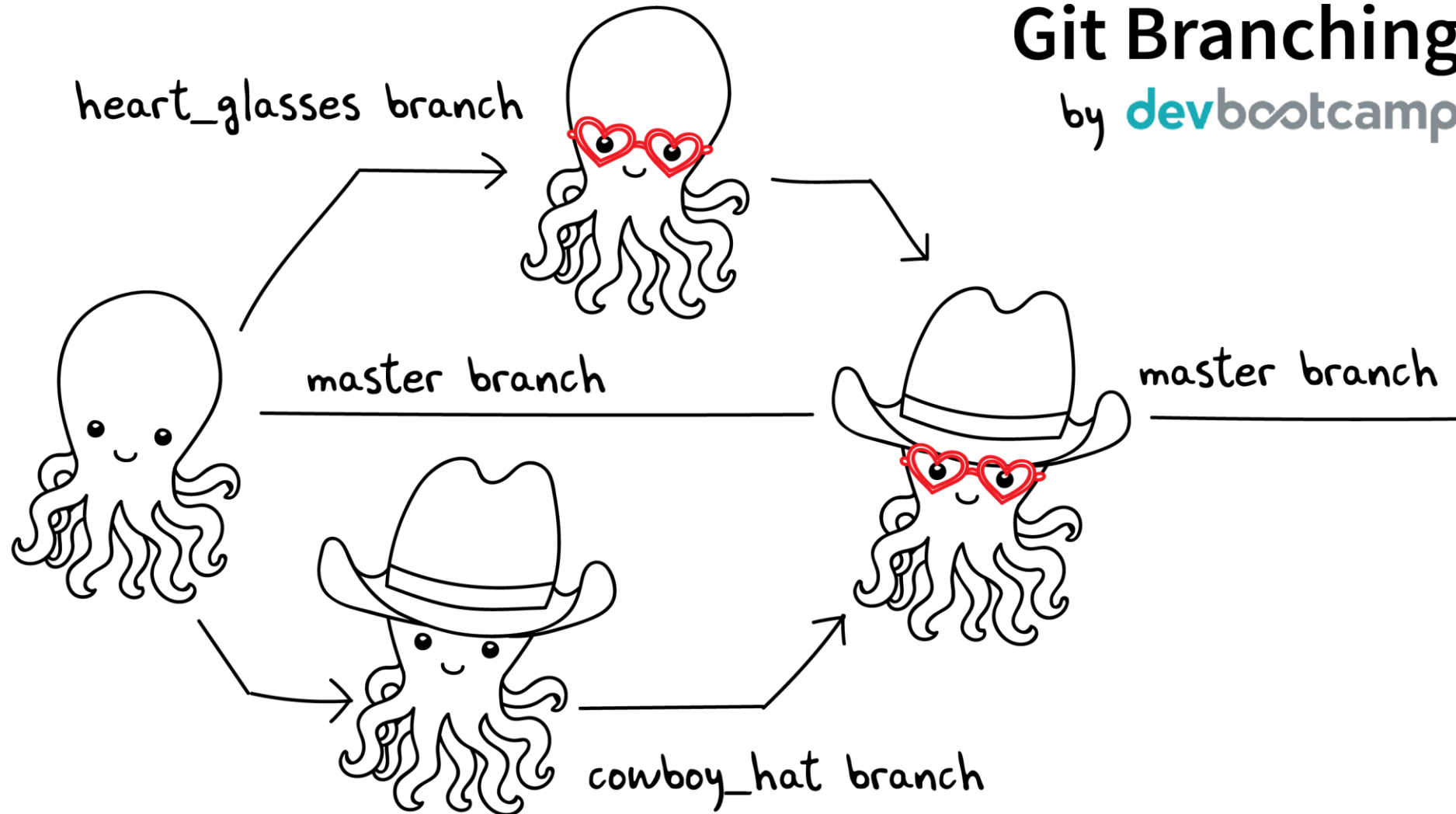
- Using version control provides
  - Clear tracking of the repo's history
  - Management and view of different branches (work)
  - Collaboration through merging of branches



Images: nobledesktop.com

# Branching & Merging

Git Branching  
by **dev**bootcamp





# Git vs GitHub

- Git: version control system
  - the actual software
- GitHub
  - cloud-based storage website



# What is Git?

- Git is version control software, created by Linus Torvalds, the same person who created the Linux operating system.
- Monitor files on your computer and tracks changes made to them over time
- Uses the command line

# Why is it called “Git”?

- Linus Torvalds: "I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'Git'".
- Git: “A silly, incompetent, stupid or annoying person (usually a man).” (Wikitionary)

git / README 



Linus Torvalds Initial revision of "git", the information manager from hell

Code

Blame

168 lines (135 loc) · 8.2 KB

```
1
2     GIT - the stupid content tracker
3
4     "git" can mean anything, depending on your mood.
5
6     - random three-letter combination that is pronounceable, and not
7       actually used by any common UNIX command. The fact that it is a
8       mispronunciation of "get" may or may not be relevant.
9     - stupid. contemptible and despicable. simple. Take your pick from the
10      dictionary of slang.
11     - "global information tracker": you're in a good mood, and it actually
12       works for you. Angels sing, and a light suddenly fills the room.
13     - "goddamn idiotic truckload of sh*t": when it breaks
14
```

# What is GitHub?

- GitHub is a Microsoft subsidiary that offers cloud hosting for Git repositories
- Provides a GUI (Graphical User Interface) for Git
  - Most (but not all) features are available in the GitHub Desktop client
- Allows for easy collaboration and sharing
  - Issue queues for bugs and features, pull requests, and more
- GitHub basic is free (up to 5GB of storage)
  - Hosts both open and private repositories
- GitHub Enterprise (free for CU affiliates)
  - Includes cloud-based development environments
  - <https://oit.colorado.edu/services/business-services/github-enterprise>

# Alternatives to...

- Git (version control)
  - Apache Subversion (SVN)
  - Mercurial SCM
  - CVS (Concurrent Versions System)
- GitHub (hosting)
  - GitLab
  - BitBucket
  - SourceForge



# When not to use GitHub

- When you are looking for long-term preservation
  - There's no guarantee Microsoft will keep GitHub around forever
  - Lots of source-code-hosting platforms no longer exist
  - Thousands of URLs in research articles no longer work as they point to code hosted by defunct services
- When you want your code to be cited
  - [Zenodo](#) is an Open Access repository that can be linked to GitHub
  - Allows you to archive a specific release of public GitHub projects
  - Creates a DOI for the archive that you can use in citations





















Zenodo.org will be unavailable for 2 hours on September 29th from 06:00-08:00 UTC. See [announcement](#).

September 2, 2023

Software

Open Access


# PyGMT: A Python interface for the Generic Mapping Tools

 Tian, Dongdong;  Uieda, Leonardo;  Leong, Wei Ji;  Schlitzer, William;  Fröhlich, Yvonne;  Grund, Michael;  Jones, Max;  Toney, Liam;  Yao, Jiayuan;  Magen, Yohai;  Jing-Hui, Tong;  Materna, Kathryn;  Belem, Andre;  Newton, Tyler;  Anant, Abhishek;  Ziebarth, Malte;  Quinn, Jamie;  Wessel, Paul


PyGMT is a library for processing geospatial and geophysical data and making publication quality maps and figures. It provides a Pythonic interface for the [Generic Mapping Tools \(GMT\)](#), a command-line program widely used in the Earth Sciences.

The development of PyGMT has been supported by NSF grants OCE-1558403 and EAR-1948603.


Preview

 baseline-images.zip

 baseline-images

-  test\_basemap.png 6.2 kB
-  test\_basemap\_compass.png 71.2 kB
-  test\_basemap\_loglog.png 24.5 kB
-  test\_basemap\_map\_scale.png 31.0 kB
-  test\_basemap\_polar.png 31.9 kB

18,263

 views

1,446

 downloads

[See more details...](#)

Available in

**GitHub**

Indexed in










**OpenAIRE**

Publication date:

Zenodo.org will be unavailable for 2 hours on September 2, 2023

September 2, 2023


# PyGMT: A Python interactive Mapping Tools

 Tian, Dongdong;  Uieda, Leonardo;  Leong, Wei Ji Jones, Max;  Toney, Liam;  Yao, Jiayuan;  Magen, Newton, Tyler;  Anant, Abhishek;  Ziebarth, Malte; 






PyGMT is a library for processing geospatial and geophysical data. It provides a Pythonic interface for the [Generic Mapping Tools](#) (GMT) Sciences.

The development of PyGMT has been supported by N

Preview

 baseline-images.zip

 baseline-images

-  test\_basemap.png
-  test\_basemap\_compass.png
-  test\_basemap\_loglog.png
-  test\_basemap\_map\_scale.png
-  test\_basemap\_polar.png

## Versions

Version v0.10.0 Sep 2, 2023

10.5281/zenodo.8303186

Version v0.9.0 Mar 31, 2023

10.5281/zenodo.7772533

Version v0.8.0 Dec 30, 2022

10.5281/zenodo.7481934

Version v0.7.0 Jul 1, 2022

10.5281/zenodo.6702566

Version v0.6.1 Apr 11, 2022

10.5281/zenodo.6426493

[View all 17 versions](#)

**Cite all versions?** You can cite all versions by using the DOI [10.5281/zenodo.3781524](#). This DOI represents all versions, and will always resolve to the latest one. [Read more.](#)

24.0 KB

31.0 kB

31.9 kB

18,263

 views

1,446

 downloads

[See more details...](#)

Available in

GitHub

Indexed in

OpenAIRE

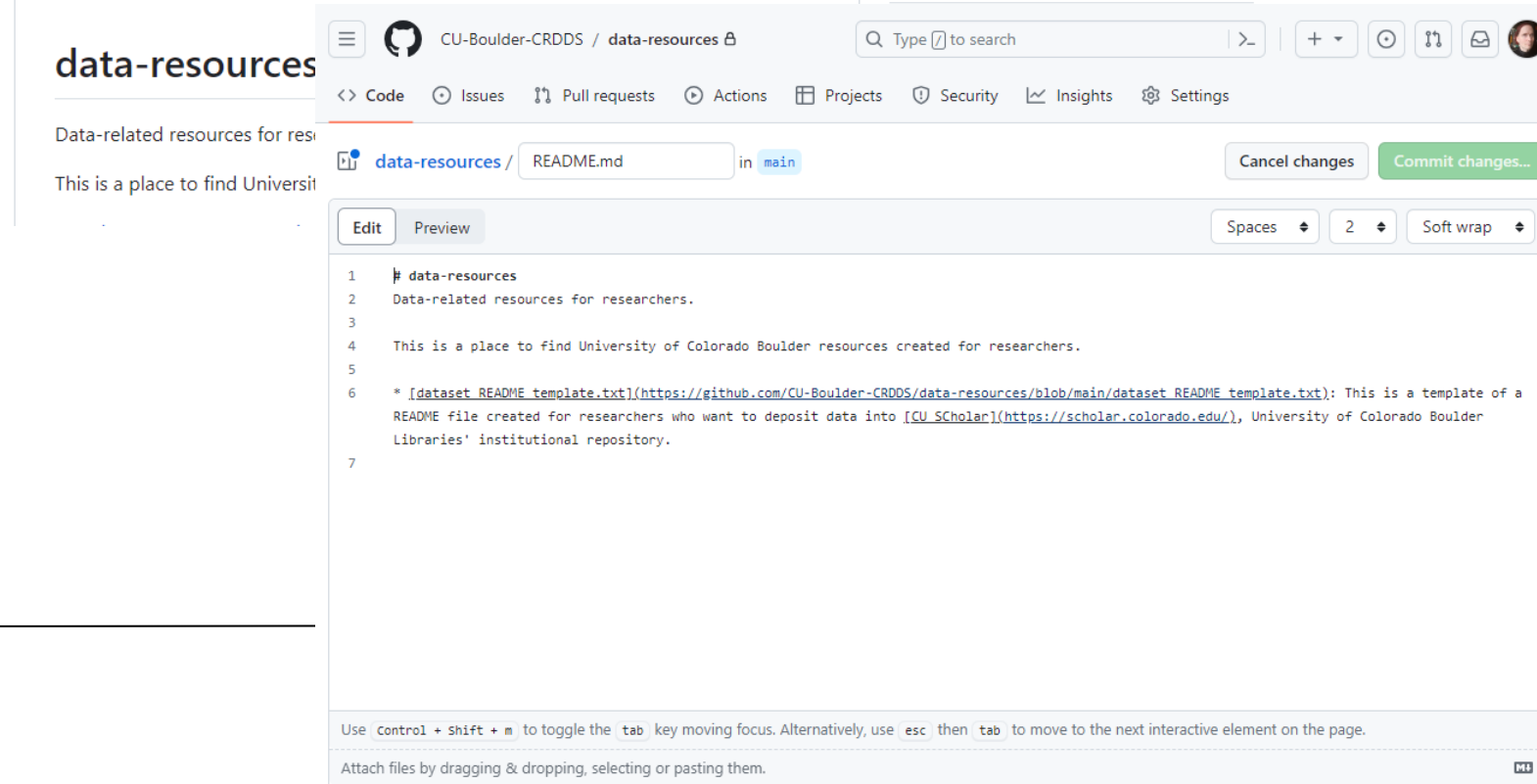
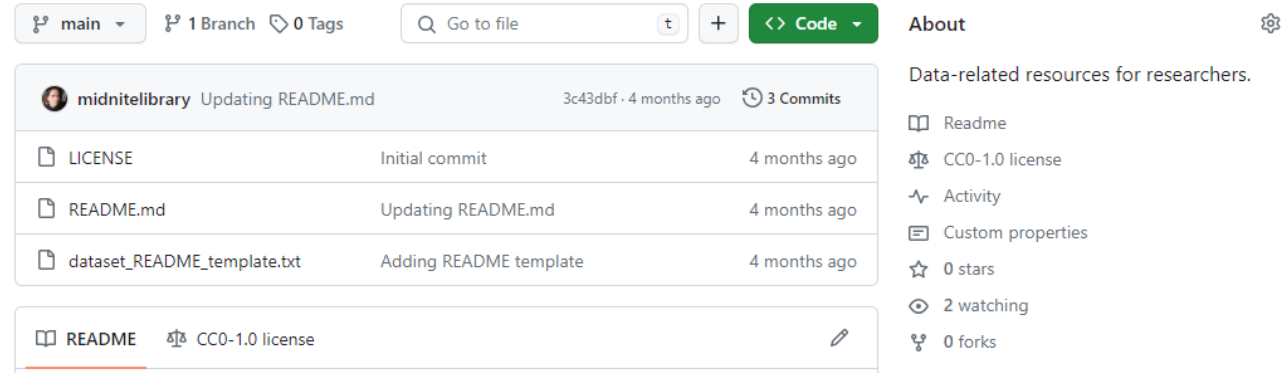
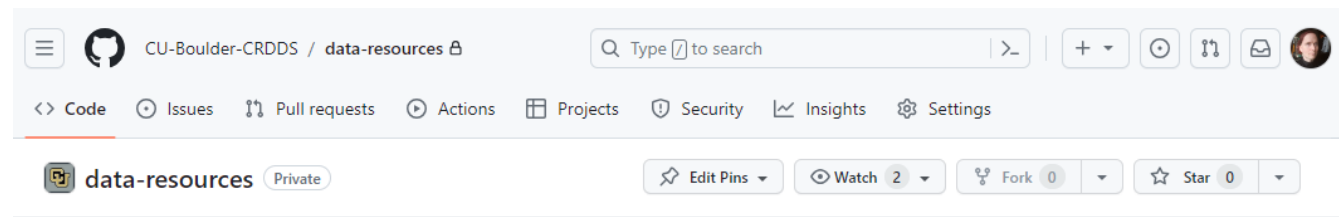
Publication date:

# Ways to interact with GitHub

- The website
- GitHub Desktop
- Using Git and the command line

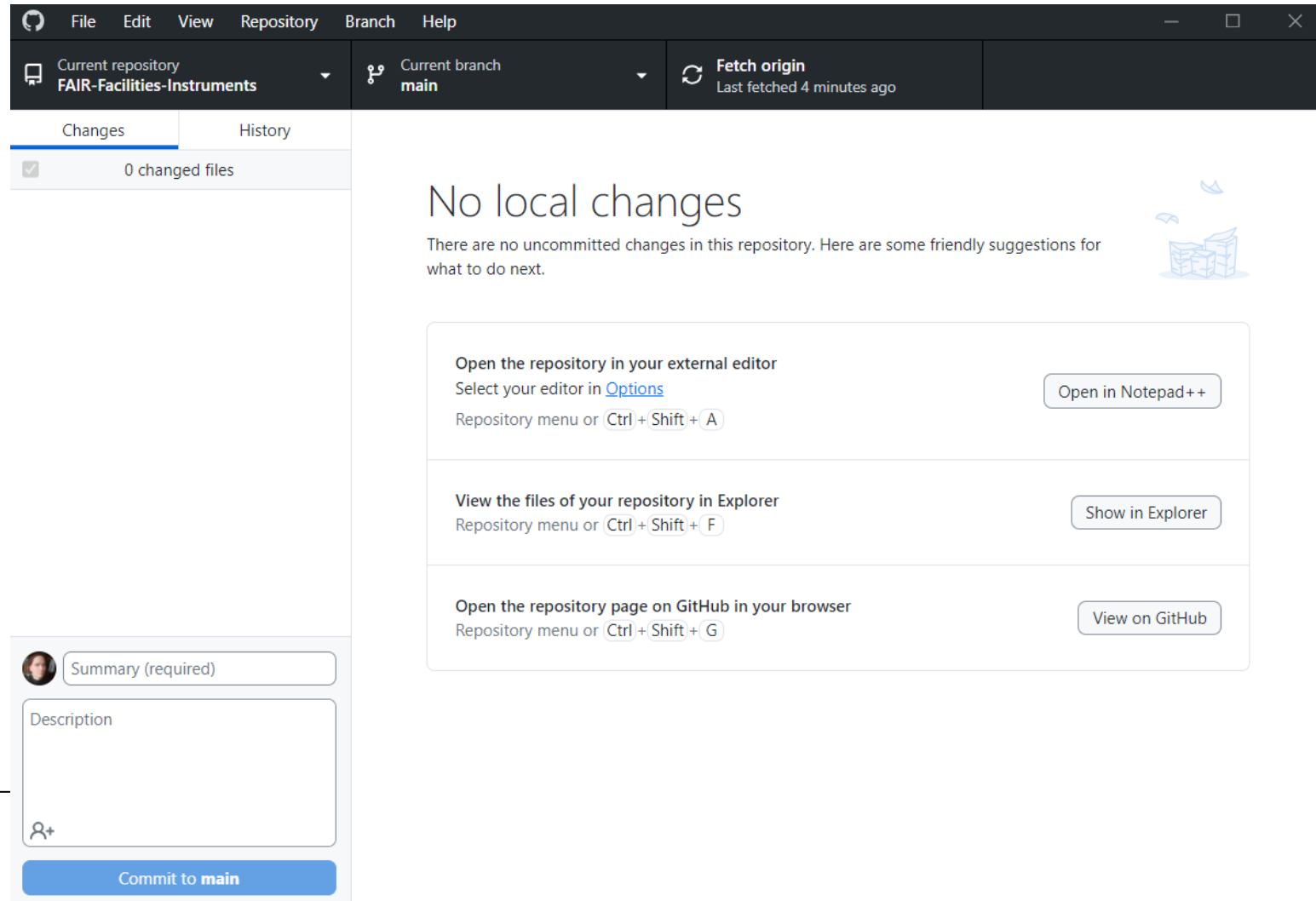
# Website

- Create and manage projects
- Upload and download files
- Write documentation

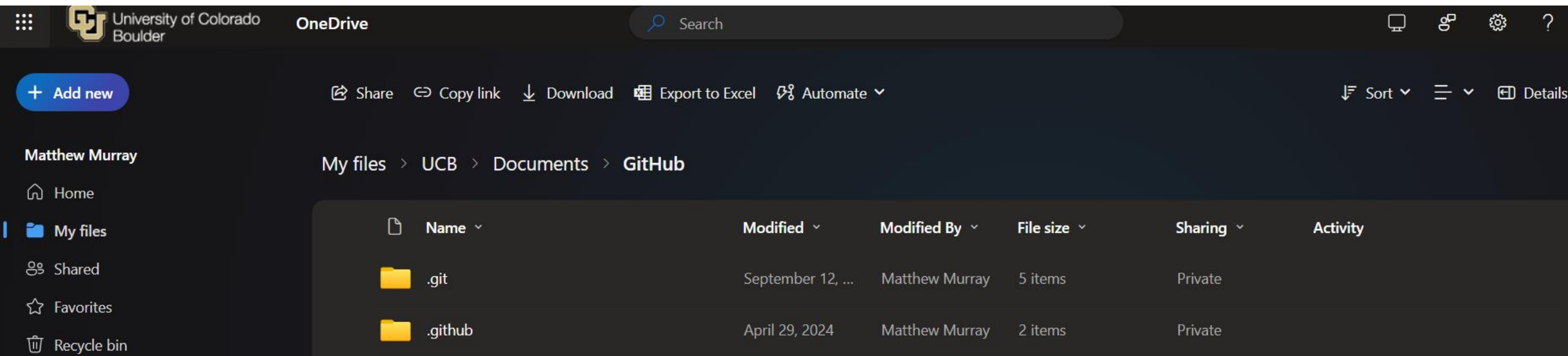


# GitHub Desktop

- Create, clone, and fork projects
- Commit changes and submit contributions
- Many people who use Git frequently prefer to use the command line
- Other graphical Git interfaces exist



# Be careful where you put your GitHub directory



University of Colorado Boulder OneDrive

Search

+ Add new

Share Copy link Download Export to Excel Automate

Sort Details

Matthew Murray

My files > UCB > Documents > GitHub

Name	Modified	Modified By	File size	Sharing	Activity
.git	September 12, ...	Matthew Murray	5 items	Private	
.github	April 29, 2024	Matthew Murray	2 items	Private	

- Microsoft OneDrive uploads a lot of stuff to the cloud, which you don't always want



# Getting Started with Git

# Getting Started

- A GitHub account set up.
  - If you don't have one, sign up at [GitHub.com](https://github.com).
- Many systems have Git installed; however, you may need to download it on your local machine:
  - See <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git> for more information on installing Git
  - To check if Git is installed, run:

```
$ git -version
```

# Linking Git to GitHub

- Once downloaded, we can then configure Git with our GitHub username and email via the command line. This allows us to interact with our GitHub more easily.

- First, set your username. For example, if my GitHub username is gh-user, then I would do the following:

```
$ git config --global user.name "gh-user"
```

- Now, set your email. For example, if my email for GitHub is gh-user@gmail.com, then I would do the following:

```
$ git config --global user.email "gh-user@gmail.com"
```

- Confirm Git has been configured (should show your entered info)

```
$ git config --list
```

# Personal Access Token

- Allows you to verify your identity with GitHub
  - For more information, see <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens>
- Downside is that you need to enter your username and then the Personal Access Token as your password for events such as:
  - Any interaction with a private repo
  - Pushing to a public repo
- There are ways to store your username and token, but these require third-party software

# SSH Keys

- Alternative way to verify your identity with GitHub
  - For more information see:  
<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>
  - Be sure to select the proper Operating System when using the link
- Setup is more involved, but makes it so that you never have to enter your username and token when interacting with a private repo or pushing to a public repo

# Getting Started with Git (local)



# Hands on tutorial

Goal: Create a simple project that contains a markdown file

First let's create a new directory for our project:

```
$ mkdir git_work
```

```
$ cd git_work
```

```
$ mkdir git-tutorial
```

```
$ cd git-tutorial
```

# Git Repository (Repo)

- A Git repository tracks and saves the history of all changes made.
  - All this information is stored in “.git”, which is the repository folder
- We can make a directory (folder) a Git repo using “git init”

# Git Init

- In your “git-tutorial” directory run

```
$ git init
```
- Git creates the "hidden" directory called “.git”

```
$ ls -la
```
- Your directory is now a repo!
  - Git is now ready to be used
  - Allows us to tell Git what items to watch

# Create the main branch

Now that we have a repo, we can create branches. Branches are a version of the repository.

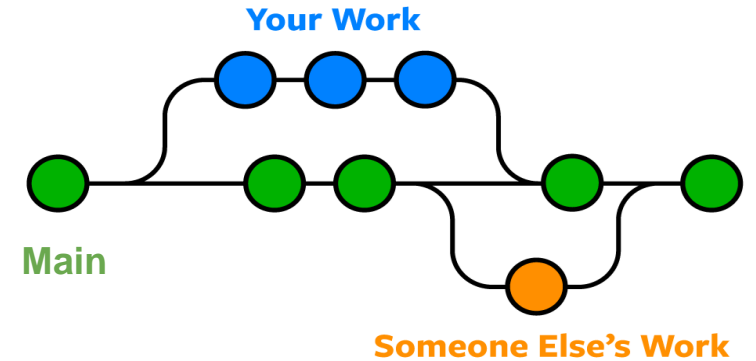
- It is customary to name the primary branch “main”

- This can be done as follows (after an init)

```
$ git checkout -b main
```

- You can switch between branches

```
$ git checkout <branch-name>
```



# Let's add a file!

- It is customary to add a README.md
  - Description of repo and any helpful information
- To add a README.md, in “git-tutorial” create and edit the file using nano (or an editor of your choice)

```
$ nano README.md
```

- Add anything you would like!
- Be sure to save the file when you exit.

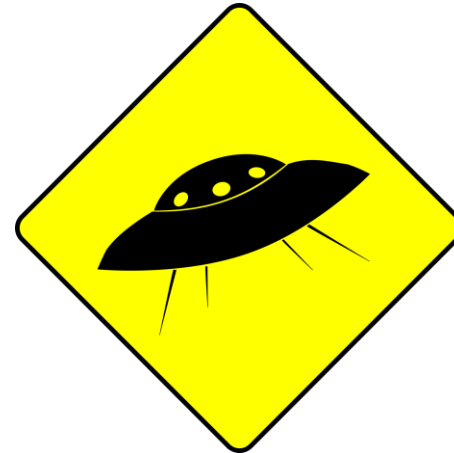
# Best Practices: Documentation

- Include documentation with your project in GitHub so that others (and you) know what your project is and how it should be used.
- A README.md (markdown) file can be included in your GitHub project and will display on the front page
- What to include in a README:
  - What your project does
  - How people can use it
  - Who you are and how to contact you
  - License information
- Lots of examples and templates available
- We can provide feedback on documentation





**Git does not know about README.md yet!!**



# Areas of Git Workflow

## Working Area

- Items that you are currently working on
- Are not tracked by Git!
- Exists locally

## Staging Area

- When Git starts tracking and saving your work
- Exists locally
- Items are added to this area by using “git add”

## Snapshot Area

- All staged items are captured
- Version of the repo
- Exists locally
- Items are added to this area by using “git commit”

## GitHub

- Exists locally and on GitHub!
- Items are added to this area using “git push”

# Git Status

- The git status command displays the state of the working and staging area.
- Let's see what area README.md is in  
`$ git status`
  - We see it is an untracked file, so it is in the working area

# **What if you don't want Git to track something?**

# .gitignore

We can add a file named “.gitignore” to our repo

- Specifies what items (files, directories, etc.) should never be tracked

- Let's create a file to ignore!

```
$ echo "Super secret stuff" > confidential_data.txt
```

- Add “.gitignore” to “git-tutorial” and put “confidential\_data.txt” in it

```
$ echo confidential_data.txt > .gitignore
```

**Let's add our files to the  
staging area now!**

# Git Add

- The git add command adds a change in the **working area** to the **staging area**.

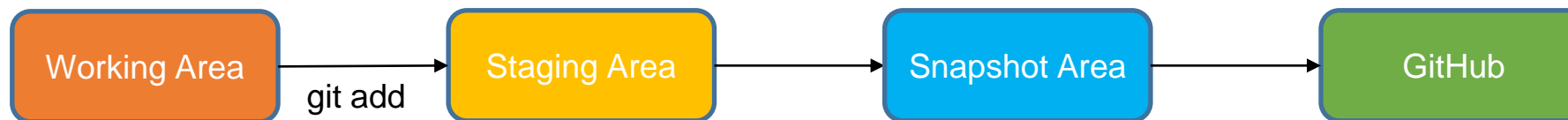
- Let's add our README.md to the staging area

```
$ git add README.md
```

or add everything in the current directory

```
$ git add .
```

- Anytime a change is made, you need to do a git add (to track them)



# Git Reset

- If you've already staged files using git add and you want to move them back to the working directory, you can use the following command:

```
$ git reset <file>
```

- To unstage the README.md file:

```
$ git reset README.md
```

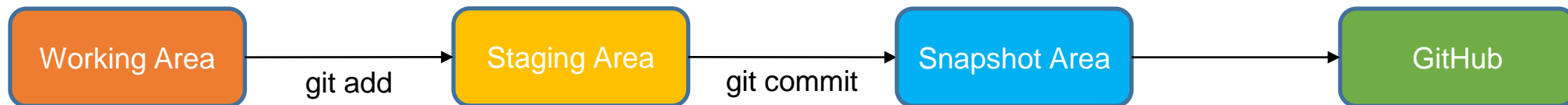


# Git Commit

- The git commit command captures a snapshot of all staged items
  - Commits can be thought of as a version of the repo
  - Commits should be accompanied with a brief message
- Let's commit our staged item!

```
$ git commit -m "Create repo, add README.md, add .gitignore"
```

```
$ git status
```



# Common practice – add, commit

- `git add`
  - Can be performed as much as you want
  - Doesn't need to be done after every change
- `git commit`
  - Always include a comment!!
  - Bundle common staged items together
  - Try not to put too many things in a commit

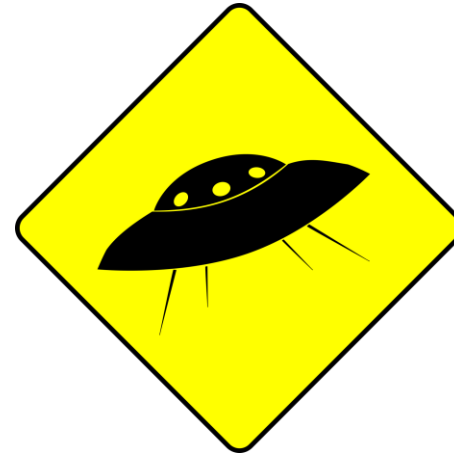
# Git Log

- The command git log **lists the commits made in that repository**
- Lists the most recent commits first

\$ git log



All changes and files are only locally stored right now!



# **To GitHub we go!**

# GitHub

- When you first create a repo locally, you will need to setup a new repository on GitHub too
  - Go to <https://github.com>
  - Sign in
  - Click on “Create New Repository” or just “New”

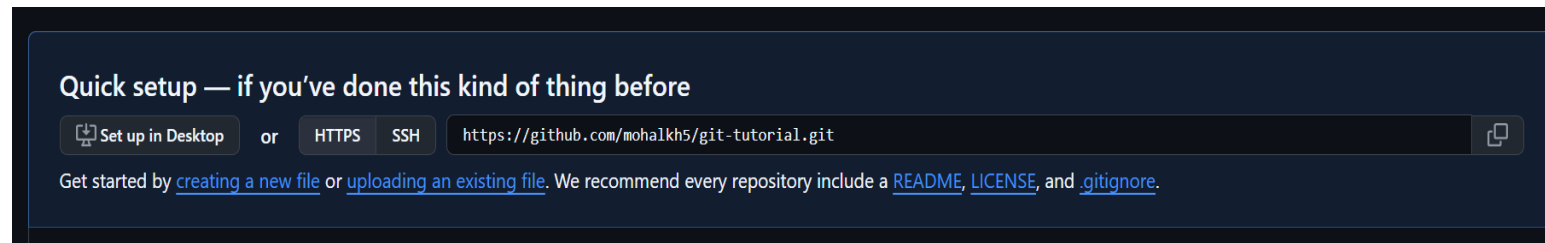
Recent Repositories

 New

Find a repository...

# Create Repo in GitHub

- Name your repo, I chose “git-tutorial”
- Don’t add a README or a .gitignore
- Click “Create repository”
- We have set everything up in the previous slides, we only need to copy the ssh link!



# Linking local repo to GitHub repo



# Git Remote

- Used to identify the remote (e.g. GitHub) repos are linked to your local repo
- Used to link remote repos to your local repo
- To view currently linked remote repos:

```
$ git remote -v
```

- To link our remote repository:
  - When using an SSH key do:

```
$ git remote add origin git@github.com:<user>/git-tutorial.git
```

- When using a Personal Access Token do:

```
$ git remote add origin https://github.com/<user>/git-tutorial.git
```

# **Sending local changes to GitHub**

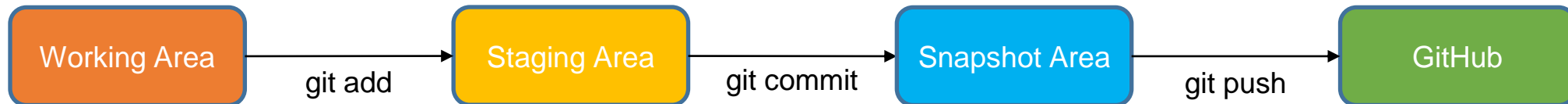
# Git Push

**Uploads local repository content to a remote repository**

- Pushing is how you transfer commits from your local repo to a remote repo

```
$ git push <name of remote repo> <branch>
```

```
$ git push origin main
```



# GitHub

- Go back to GitHub and refresh your page
  - should see the files we have added (and not the ones we've ignored)
- Some cool features!
  - look at our commits
  - directly edit/commit in the browser
- Let's do that! Let's something and commit it on GitHub
  - But now our remote repo is one commit ahead of our local one...

# Git Fetch & Merge

- Git fetch retrieves the changes from the remote repo

```
$ git fetch
```

- Git merge combines two branches

```
$ git merge origin/main
```

There's an easier way!

# Git Pull

- Git pull combines the fetch and merge commands

```
$ git pull <name of remote repo> <branch>
```

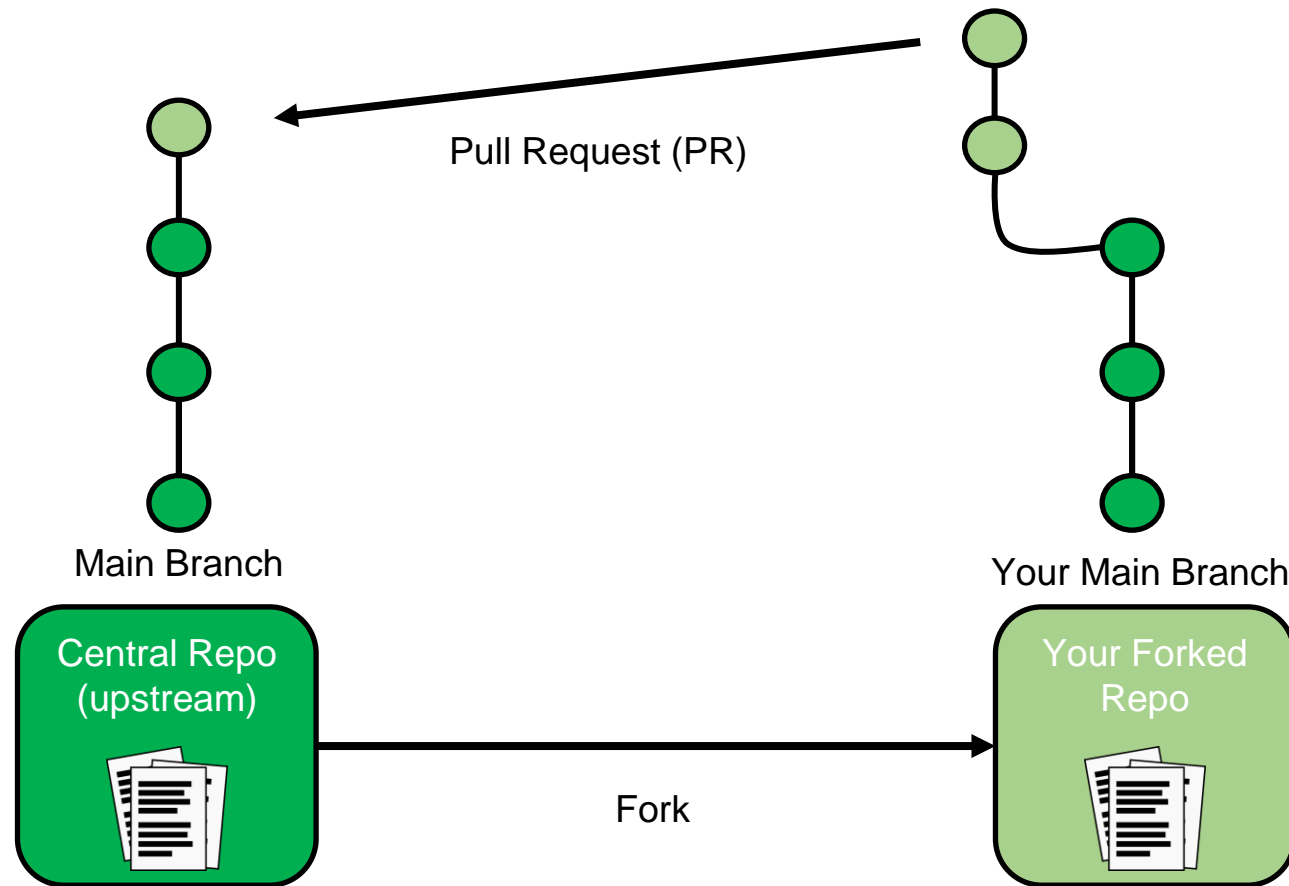
```
$ git pull origin main
```

## IMPORTANT!

- Make sure no commits have been done on local branch
- It is fine to have staged items (git add)
- ALWAYS do git pull before any commits!

# **Advanced topic: Collaboration**

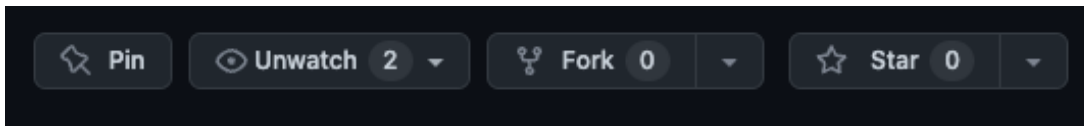
# GitHub Forks





# GitHub Forks

- Improves collaboration
  - Don't have to worry about disturbing the upstream repo
  - Improves transparency through pull requests
- Go ahead and Fork my repo:
  - Go to <https://github.com/mohal-k/git-tutorial>
  - Click “Fork” button
  - Click “Create fork”
- Creates your own version of my repo under your GitHub



# Git Clone

- Git clone **makes a clone (or copy) of a remote repo in a new directory, at another location.**  
`$ git clone <url> <optional new name>`
- Easy way to grab third-party code, or pre-existing code you might need to work on
- Cloning when you have SSH keys (be sure to make “git\_work\_cloned”):

```
$ cd git_work_cloned
```

```
$ git clone git@github.com:<user>/git-tutorial.git
```

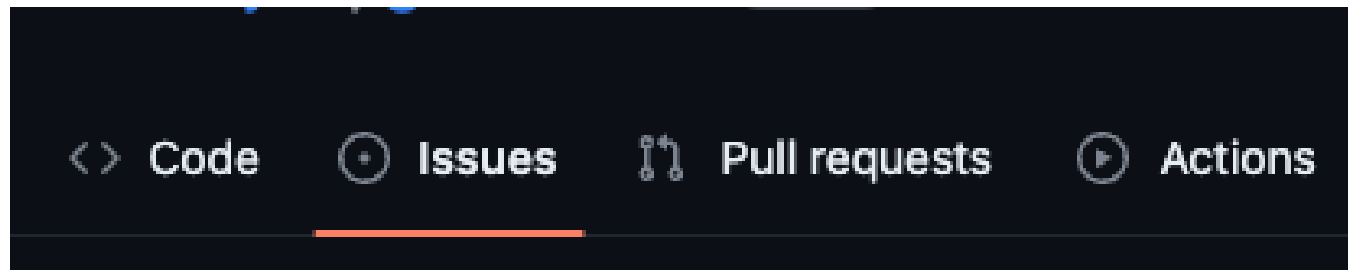
- Cloning when you are using Personal Access Tokens (be sure to make “git\_work\_cloned”):

```
$ cd git_work_cloned
```

```
$ git clone https://github.com/<user>/git-tutorial.git
```

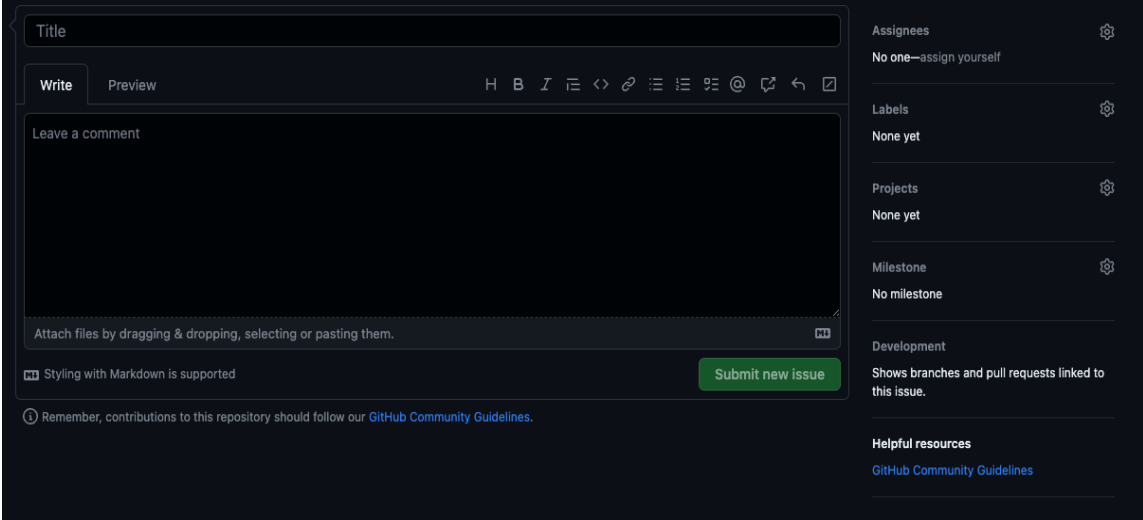
# GitHub Issues

- Allows you to discuss the project
- Point out issues, request features, ask for help
- Useful place to see past user discussion



# GitHub Issues

- Include as much detail as possible
  - Version of software
  - Operating system
- Provide a simple minimal example, if possible
- If a feature request
  - Outline possible implementation
  - Highlight its value



The screenshot shows the GitHub 'New Issue' form. It features a 'Title' input field at the top. Below it are 'Write' and 'Preview' tabs, with a rich text editor toolbar containing icons for bold, italic, code, link, and other formatting options. The main body of the form is a large text area with the placeholder text 'Leave a comment'. Below the text area is a file upload section with the text 'Attach files by dragging & dropping, selecting or pasting them.' and a 'Submit new issue' button. At the bottom, there is a note about GitHub Community Guidelines. On the right side, there are sections for 'Assignees' (showing 'No one—assign yourself'), 'Labels' (showing 'None yet'), 'Projects' (showing 'None yet'), and 'Milestone' (showing 'No milestone'). There is also a 'Development' section with a link to 'Shows branches and pull requests linked to this issue.' and a 'Helpful resources' section with a link to 'GitHub Community Guidelines'.

# Pull Requests (PRs)

- A request that an upstream repo pull your branch into their branch
- Starting a PR does not automatically merge changes
  - Notifies maintainers of upstream repo
  - Allows maintainers to review your changes
    - Discussion of changes
    - Requested additional changes
- Maintainers of upstream repo merge in the changes

# PR steps

1. Fork upstream repo
2. Clone the forked repo
3. Connect forked and cloned repo to upstream repo. Ex. using SSH keys:  
`$ git remote add upstream git@github.com:rctutorial/hello-world.git`
4. Create a new branch specific to your change  
`$ git checkout -b <new-branch> <branch-to-copy>`
5. Make your changes on this branch
6. Perform a git add, commit, and push to origin
7. Create a PR from GitHub

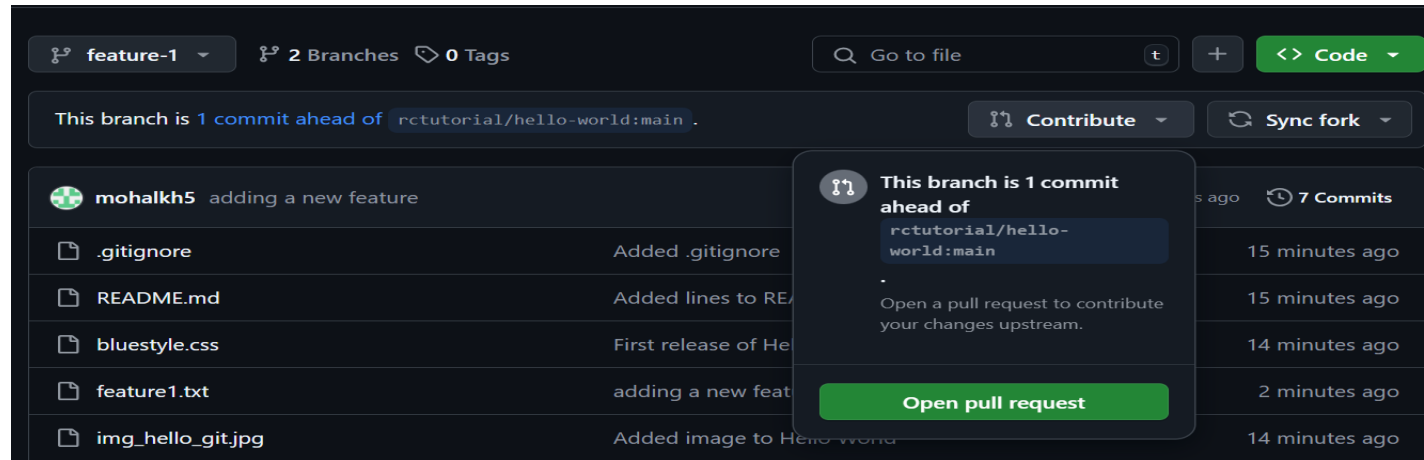
# Creating a PR

- After you push your changes to the forked repo, you can click the pop-up “Compare & pull request” on GitHub
  - Will disappear after some time



# Creating a PR

- After you push your changes to the forked repo
  1. Switch to your new branch
  2. Click the drop-down arrow next to “Contribute”
  3. Click “Open pull request”
- Will NOT disappear!

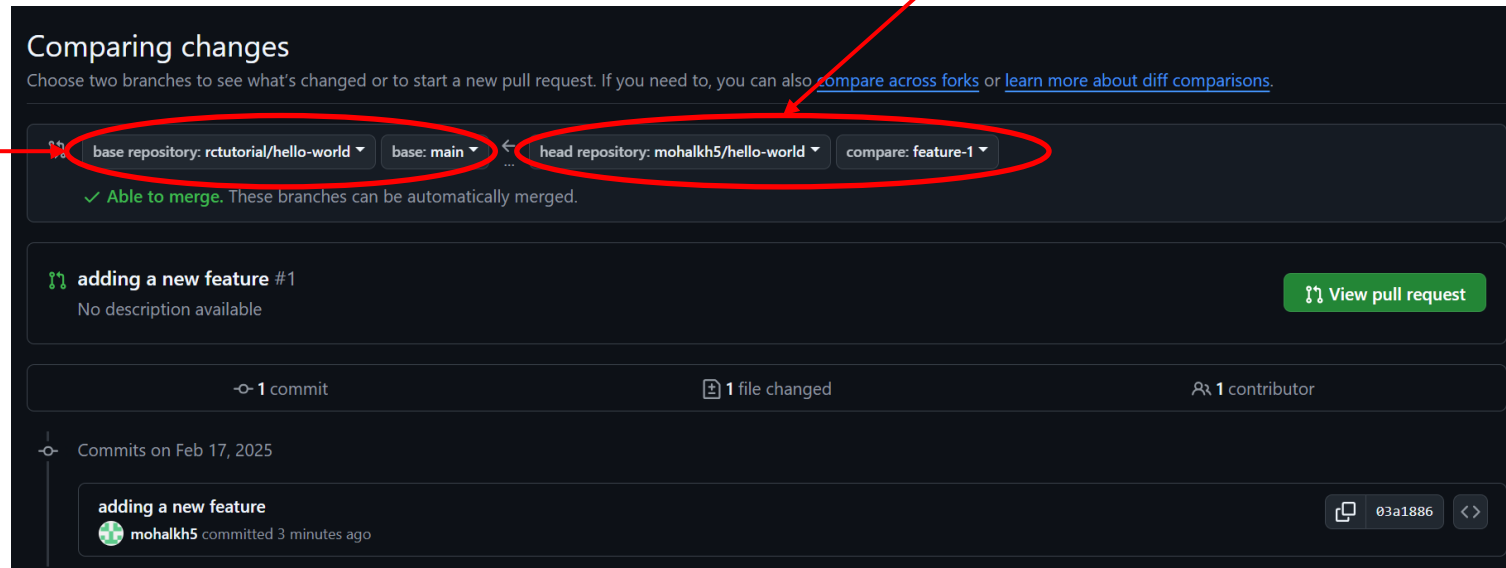




# Creating a PR

Upstream repo and branch we want to merge into

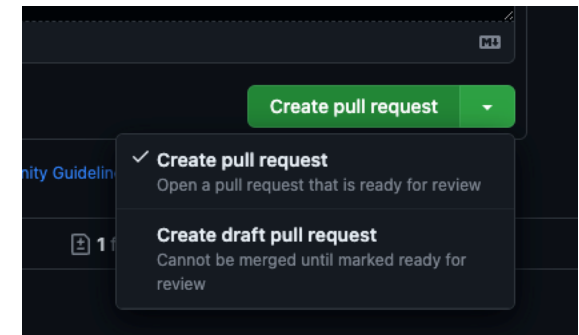
Our forked repo and branch with our changes



The screenshot shows the GitHub 'Comparing changes' interface. At the top, it says 'Comparing changes' and provides a link to 'compare across forks' or 'learn more about diff comparisons'. Below this, there are two dropdown menus for the 'base repository' and 'head repository'. The 'base repository' is set to 'rctutorial/hello-world' and the 'head repository' is set to 'mohalkh5/hello-world'. Both are circled in red. Below the dropdowns, it says 'base: main' and 'compare: feature-1'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' Below this, there is a section for 'adding a new feature #1' with a 'View pull request' button. At the bottom, it shows '1 commit', '1 file changed', and '1 contributor'. A commit message 'adding a new feature' by 'mohalkh5' is visible, committed 3 minutes ago.

# Pull Requests – Best Practice

- Create a new feature branch of forked repo
- When submitting a PR
  - Provide a short descriptive title
  - In comment section
    - Link to any current issue
    - Describe what the PR does and reasons for it
- Draft pull requests
  - PR is a work in progress
  - Can be used for discussion



# Merging

- When doing a “git pull” you are merging in changes
- This process can be done manually
- When collaborating, multiple individuals can be working on the same item
  - Conflicts can happen!
- One needs to manually resolve conflicts
- Fantastic tutorial on merging:

<https://www.atlassian.com/git/tutorials/using-branches/git-merge>

# Help! I'm stuck, where do I go?

- Trainings with Center for Research Data and Digital Scholarship (CRDDS): <https://www.colorado.edu/crdds/>
- Software Carpentries tutorial: <https://swcarpentry.github.io/git-novice/index.html>
- GitHub Student Developer Pack: <https://education.github.com/pack>
- Helpdesk: [rc-help@colorado.edu](mailto:rc-help@colorado.edu)

# Survey and feedback

Survey: <http://tinyurl.com/curc-survey18>



Slides: [https://github.com/ResearchComputing/git\\_github\\_in\\_depth\\_short\\_course](https://github.com/ResearchComputing/git_github_in_depth_short_course)