Introduction to Git and GitHub

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

Be Boulder.

Working with Git and GitHub

Brandon Reyes

- Research Computing
- Website: www.rc.colorado.edu
- Helpdesk: rc-help@colorado.edu

Matthew Murray

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

- Slides: https://github.com/ResearchComputing/intro_git_github_primer
- 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

- Understand the basics of version control
- Know the differences between Git and GitHub
- Learn basic Git fluency

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

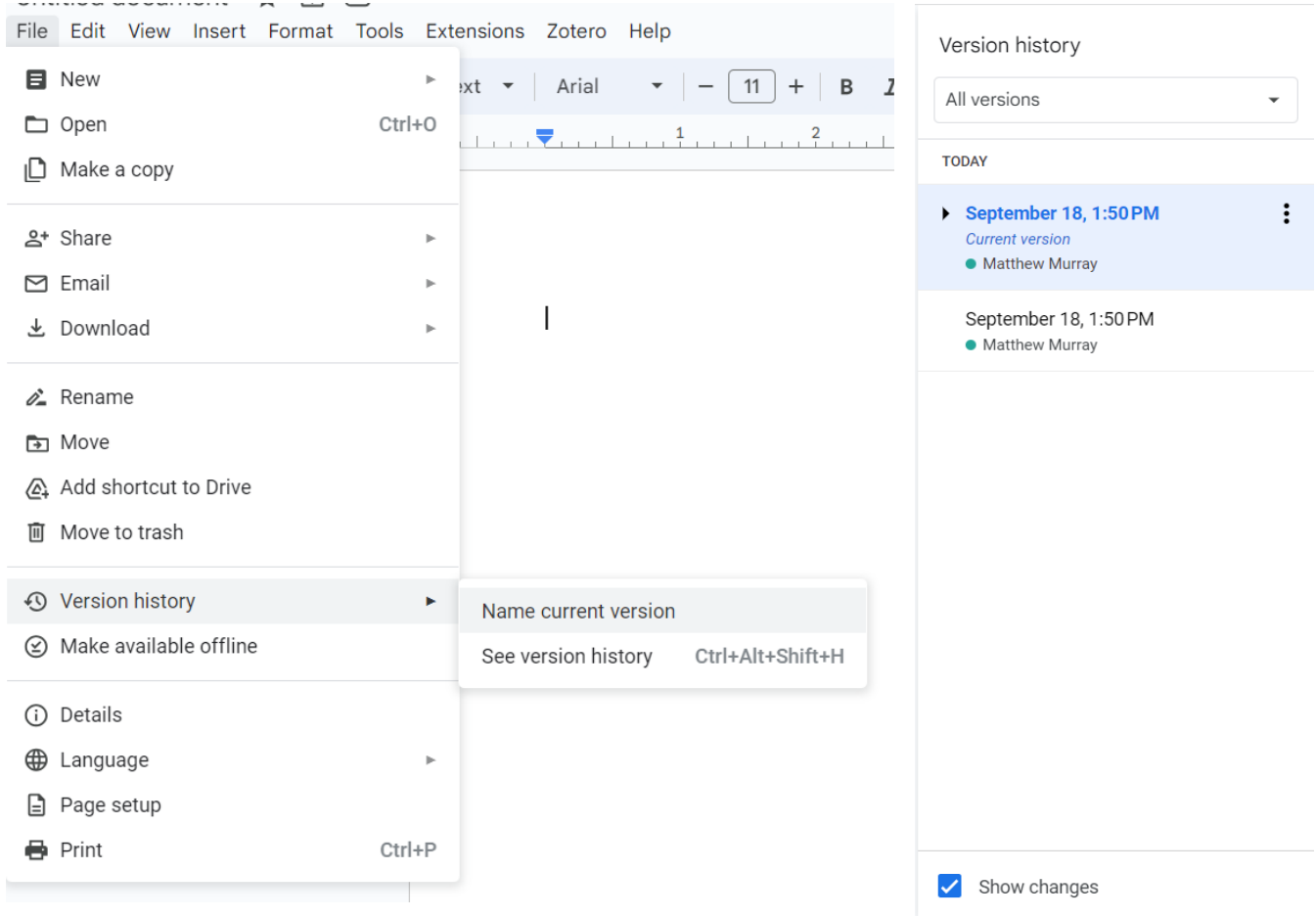
What is version control?

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.

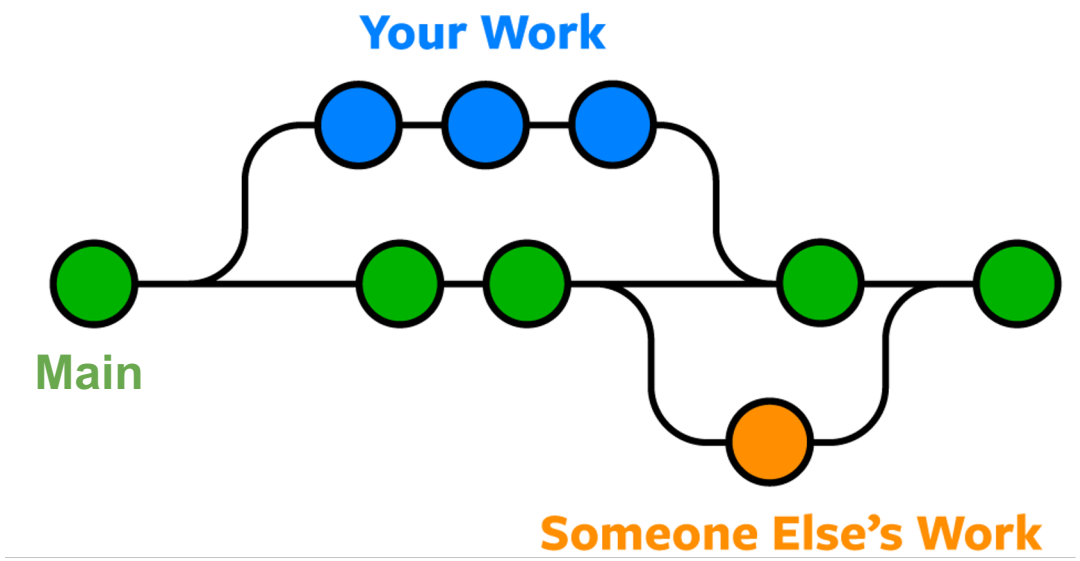
What is version control?

- 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

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       mispronounciation 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
  - 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

Images from Wikipedia

When not to use GitHub

- When you are looking for long-term preservation
  - There's no guarantee Microsoft will keep GitHub around forever
  - Thousands of URLs for projects in Git Hosting Platforms in articles no longer work
- 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

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

Publication date:

zenodo

Search

Upload Communities

themattthewmurray@gmail.com

Zenodo.org will be unavailable for 2 hours on Septembe

September 2, 2023

PyGMT: A Python inte 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 geophy provides a Pythonic interface for the Generic Mapping T 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

24.5 kB
31.0 kB
31.9 kB

Versions

Version v0.10.0 10.5281/zenodo.8303186	Sep 2, 2023
Version v0.9.0 10.5281/zenodo.7772533	Mar 31, 2023
Version v0.8.0 10.5281/zenodo.7481934	Dec 30, 2022
Version v0.7.0 10.5281/zenodo.6702566	Jul 1, 2022
Version v0.6.1 10.5281/zenodo.6426493	Apr 11, 2022

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.

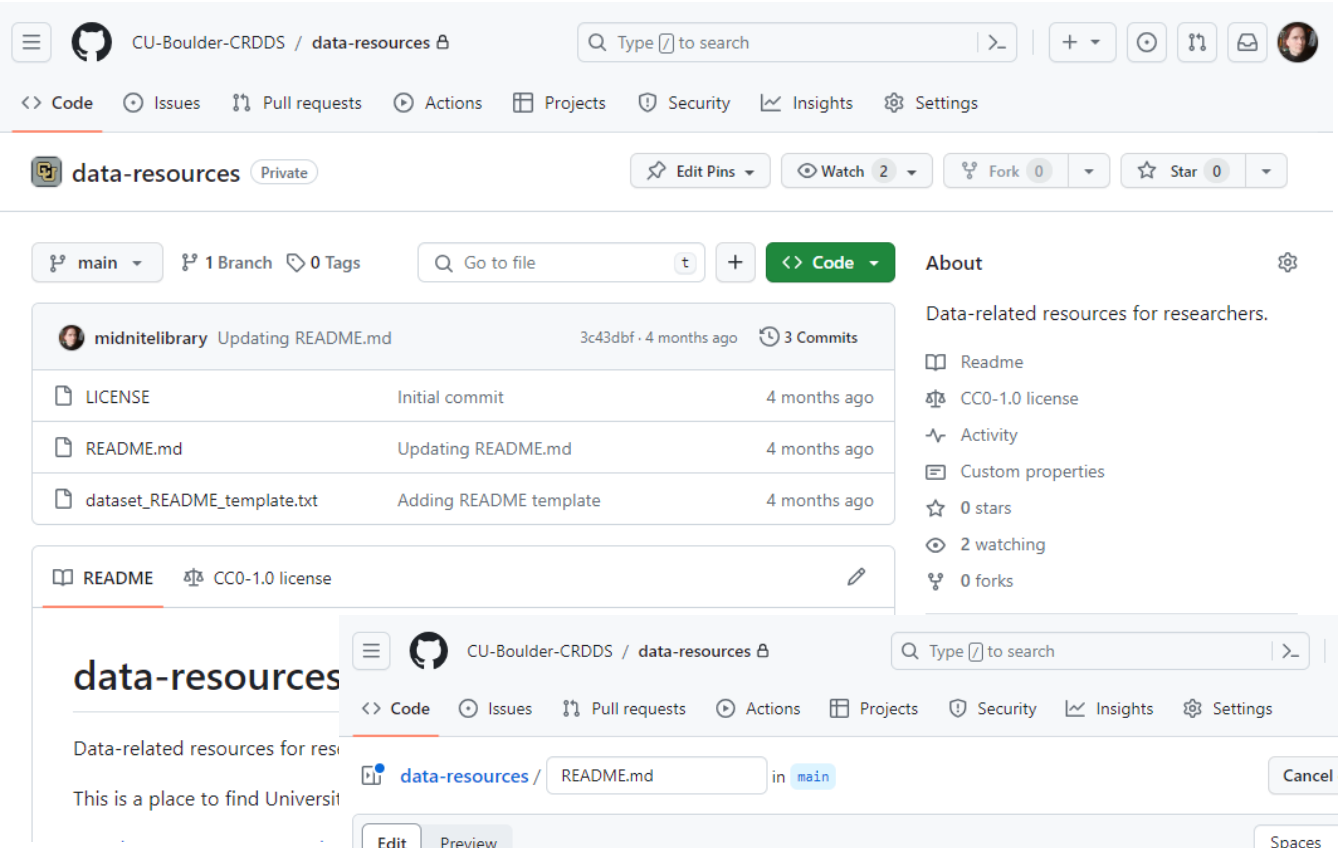
Publication date:

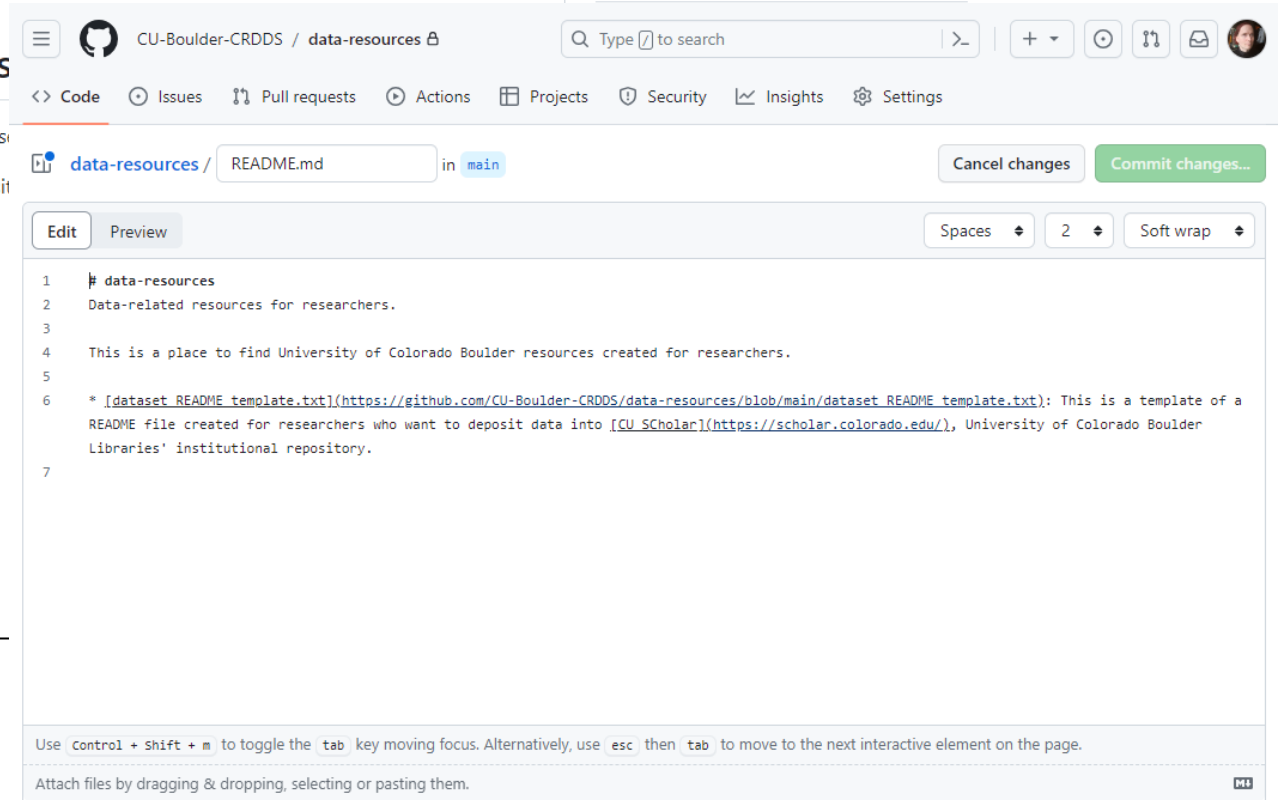
Ways to interact with GitHub

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

Website

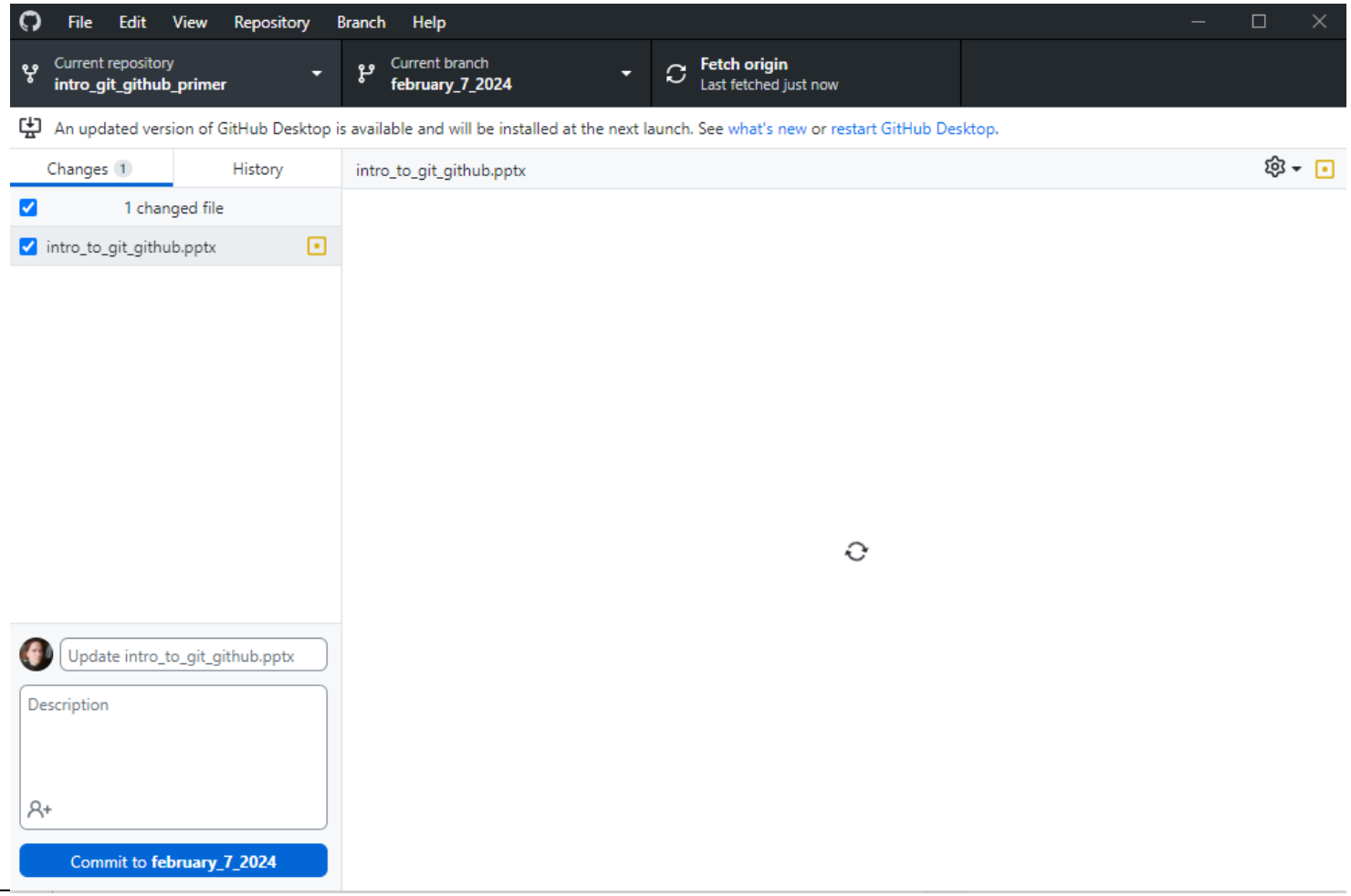
- Create and manage projects
- Upload and download files
- Write documentation





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



Getting Started with Git

Setting Git up locally

Many systems have Git installed; however, you may need to download it on your local machine

- See https://support.atlassian.com/bitbucket-cloud/docs/install-and-set-up-git for more information

Today I am going to stick with using Git on a CURC login node

Demo

Goal: Create a simple project that contains a markdown file

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

```
$ cd /projects/$USER
$ mkdir git-tutorial
$ cd git-tutotial
```

Git Repository (Repo)

A Git repository tracks and saves the history of all changes made.

- All of 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 -a
```

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

Create the main branch

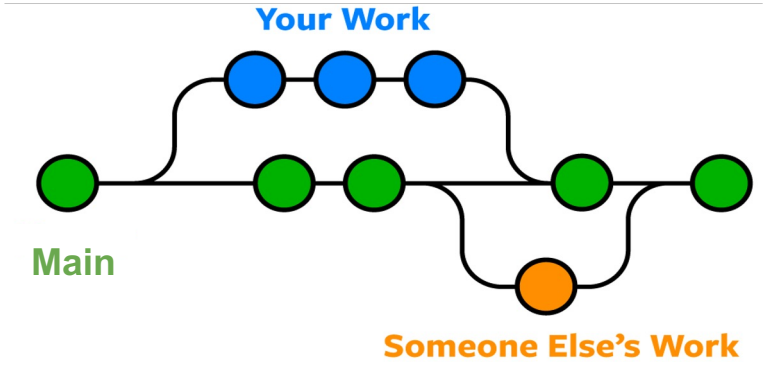
Now that we have a repo, we can create branches. Branches are a version of the repository.

- 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)

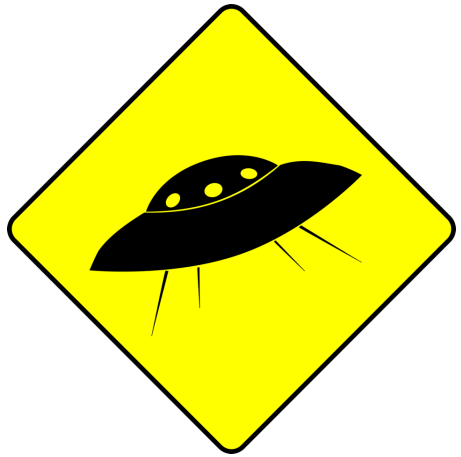
```
$ nano README.md
```

- 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

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**

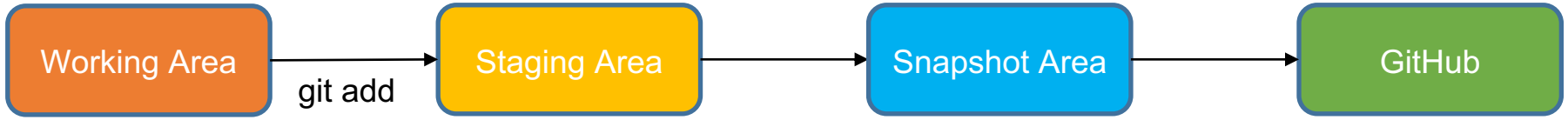
Let's add our README.md to the staging area

```
$ 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)



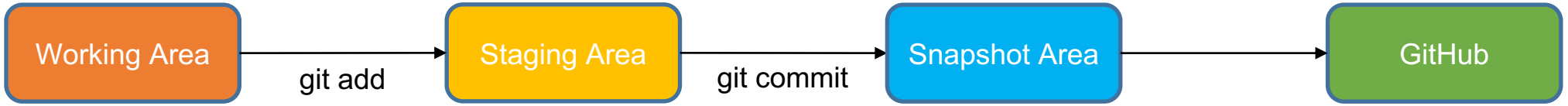
Git Commit

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

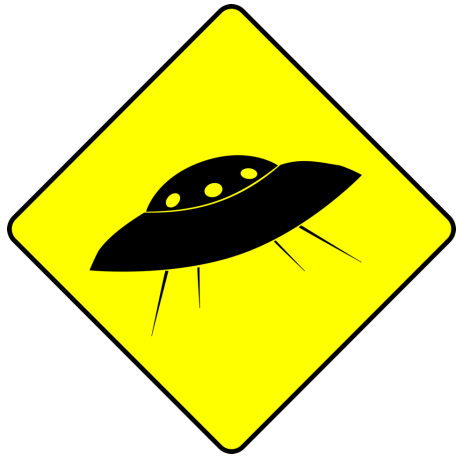
Git Log

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!

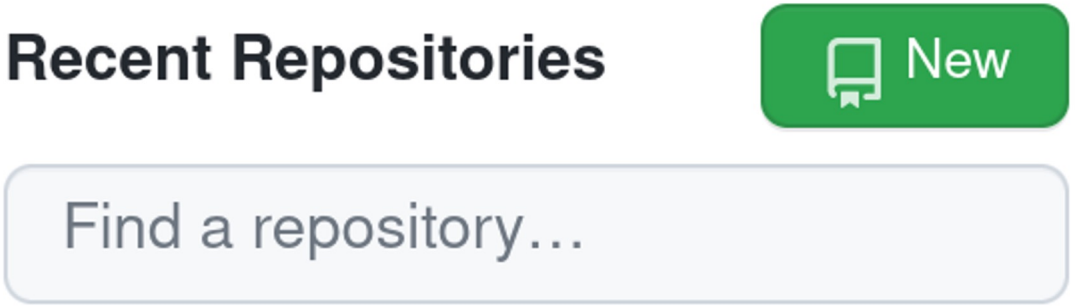


To GitHub we go!

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”



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:

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

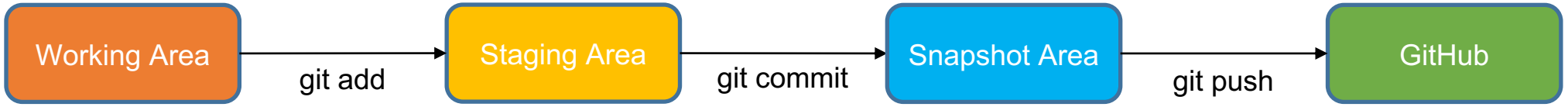
Sending local changes to GitHub

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!

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

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

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

Thank you!!

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