Git/Github

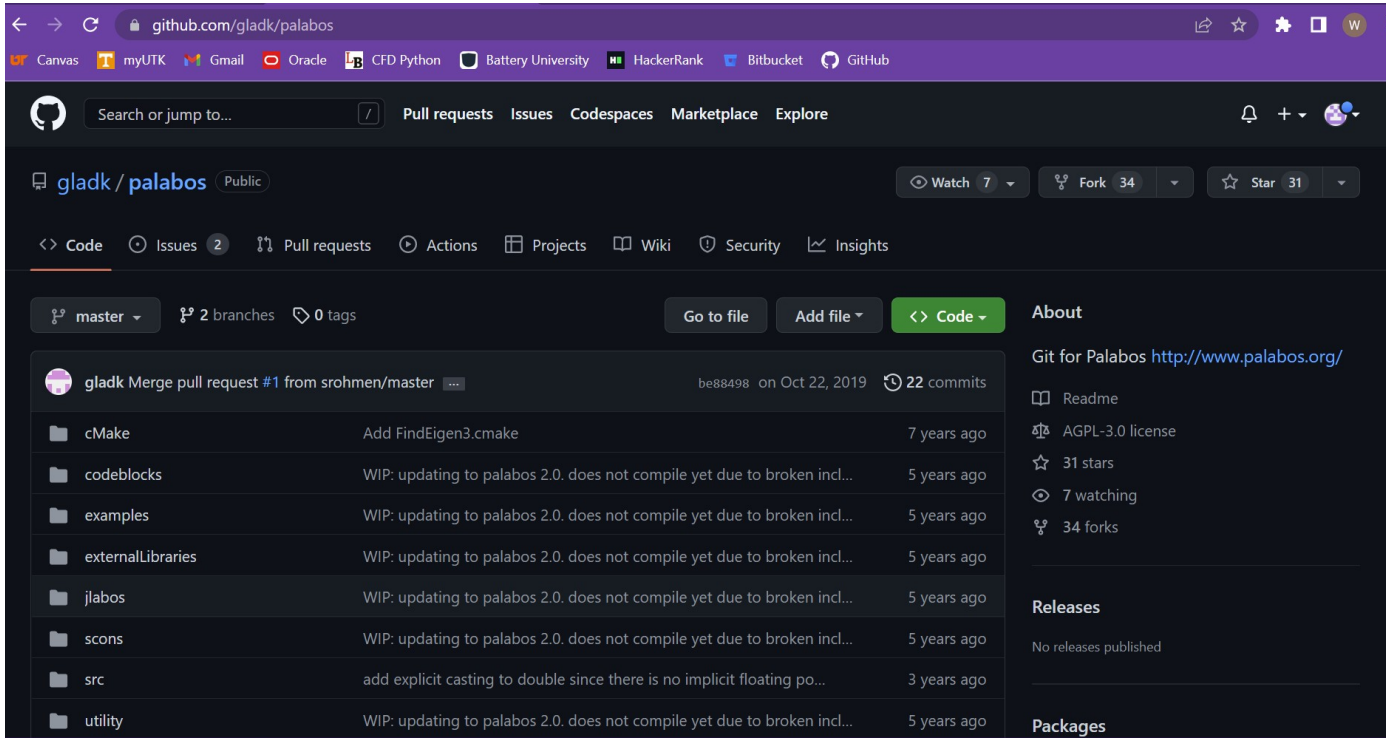
introduction/tutorial

Will Buziak

1/11/23

Introduction – Git vs. Github

- **Git** is a “form of distributed **version control**”
  - Essentially a way to **track**, collaborate and share “repositories” of programs and files
  - Operates through the command line in your terminal
- **Github** is a website that hosts **Git** repositories
  - Can be used to store, share, and host your files for collaborators or users.
  - Opensource and easy to use



Github repositories have a user-friendly interface that allows easy navigation and different developer tabs that helps highlight issues or new pull requests

What is Git?

- Git was designed for **version control.** Meaning that there is a well-established record of changes made to each file.
- Git allows “cloning” entire repositories of files that can run on a local machine quickly and seamlessly.
- Github repos can be specified as public or private and can invite collaborators and adjust permissions
- Github hosts remote repositories that are stored online and can be cloned
- Proper use of Git creates multiple backups and can be worked on offline
- Git/Github is opensource and developed by Linus Torvalds as an initiative to improve collaboration between developers
- Git can manage many files at once

Why not other file managers?

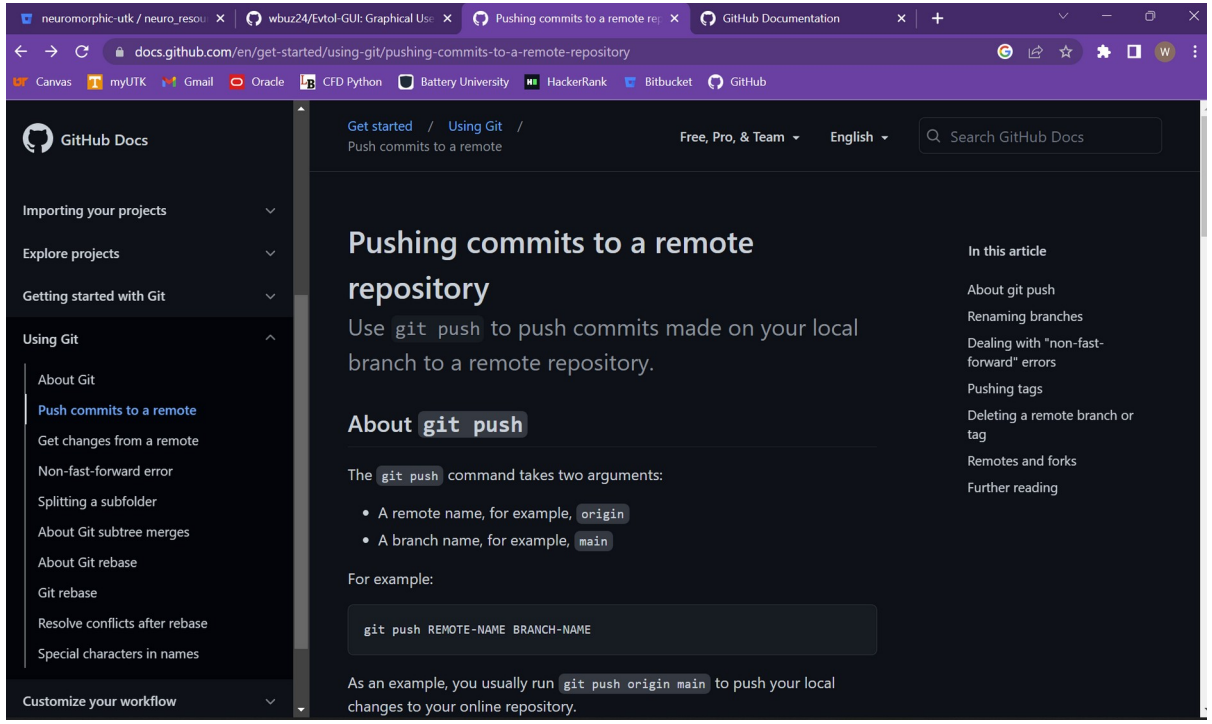
- Other file managers like Google drive allow for effective **file streaming** and **collaboration**
  - Does not allow for different workflows on the same program
  - Ideal for files that do not need to be **compiled**
  - Difficult to share **large** directories
- Git/Github (and other services like it) tracks previous generations of the same file(s)
  - Seamless to revert to previous versions
  - Turns your file directory into a “remote host” for sharing and collaborating on projects
  - Allows for multiple “branches” representing different test versions of the same repo that can be merged or kept separate.
  - Supports “forking” which allows **collaborator** rights on public repositories they did not previously have writing access to.

Installation and Tutorial

Note:

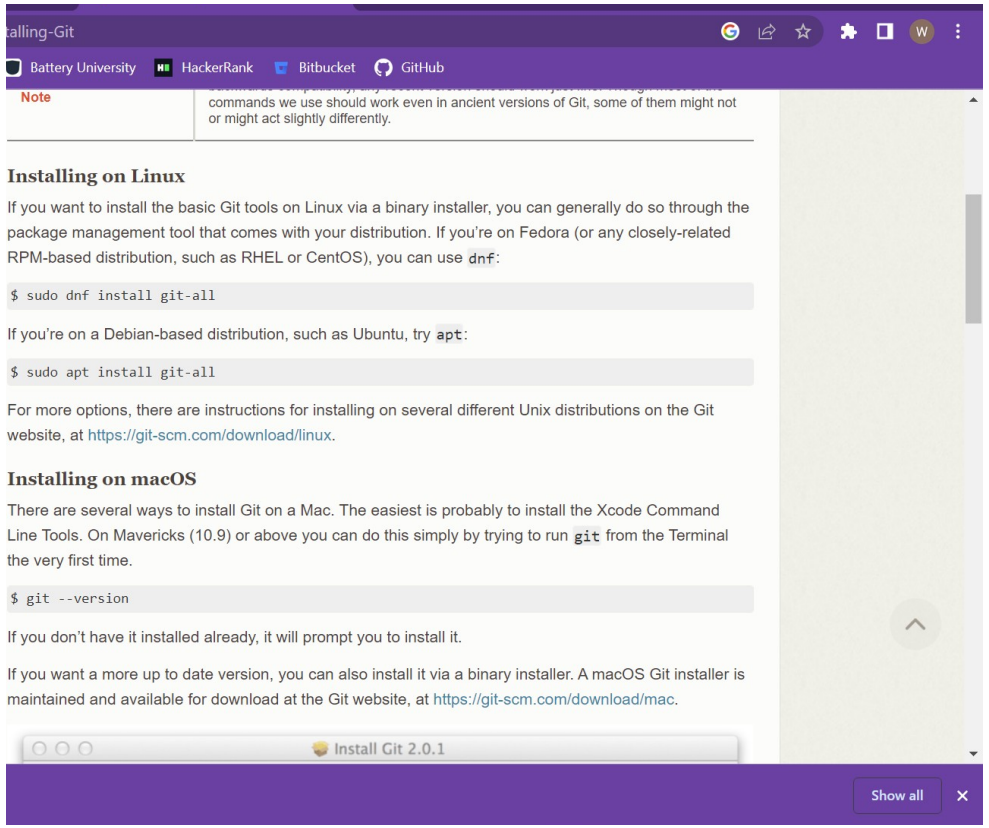
- Git has gone through a number of generations and therefore a few commands have been updated/revised.

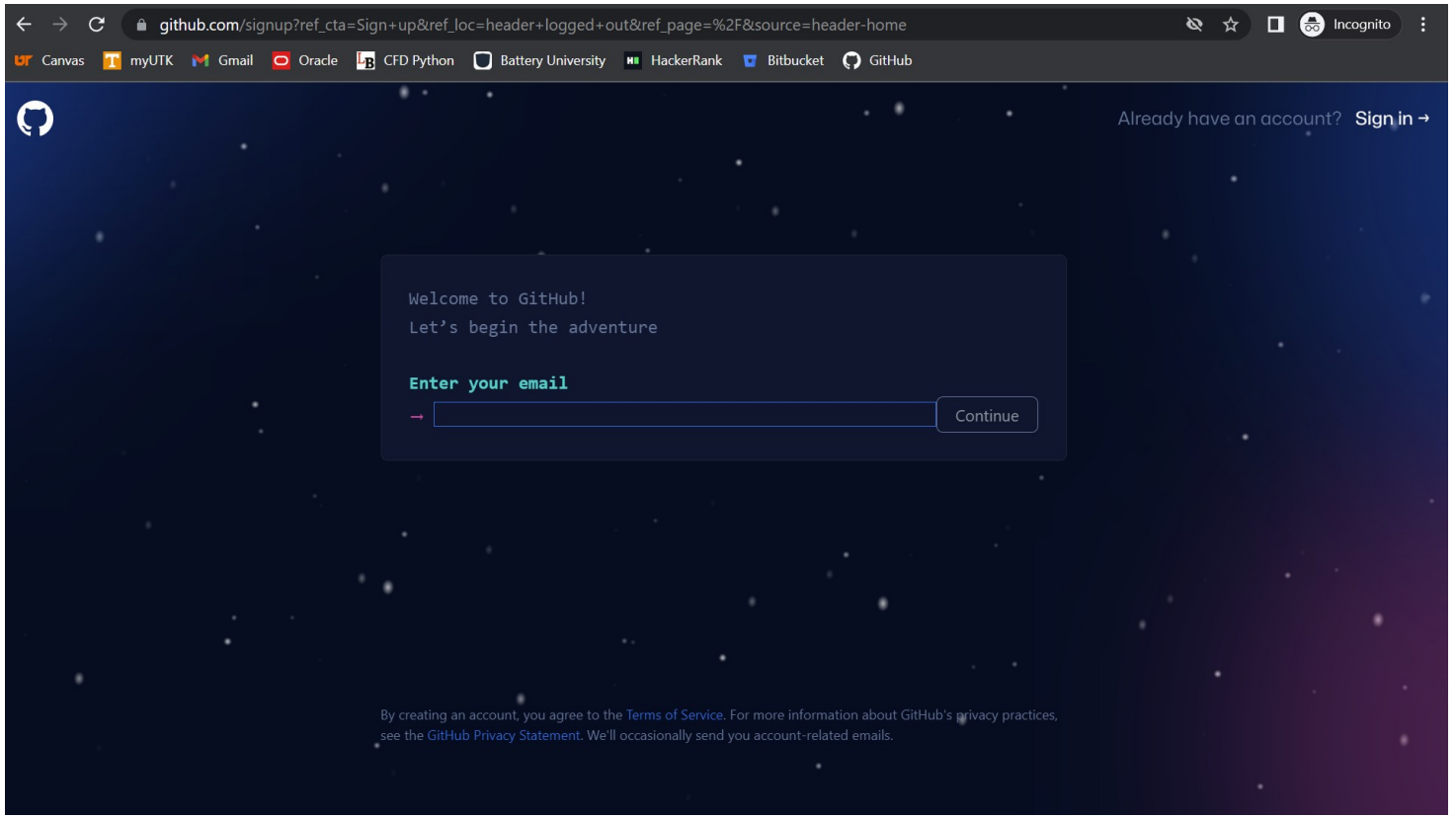
- Github Docs is your go-to resource, including OS-specific documentation.



Getting started with Git

- Create an account on Github using your desired email
- Download Git depending on your OS





Connect your local Git account with Github

- Open your terminal and run the following commands: (“**git - -version**” will check the version installed)

```
PS C:\Users\willb\Documents\Neuromorphic> git config --global user.name "wbuz24"
PS C:\Users\willb\Documents\Neuromorphic> git config --global user.email wbuziak@vols.utk.edu
PS C:\Users\willb\Documents\Neuromorphic>
```

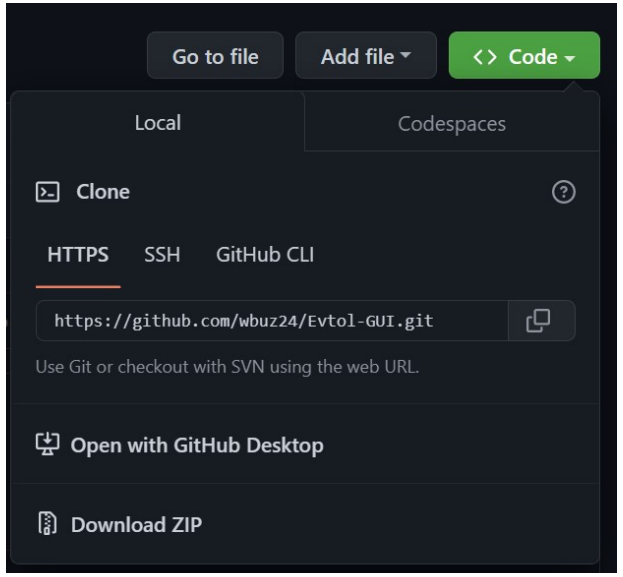
  - Configure your username
  - Configure your school email (use github generated noreply email if preferences are set to private)
    - “--global” flag configures for *every* repository on your computer

- Now, your local machine will identify you when you push commits to the **remote** repository

Clone a repository with **git clone**

- Connecting by SSH keys can be tricky but prevents the need to input your password with each pulling/pushing/cloning.

- The easiest way to clone a remote is by using the Github generated HTTPS URLs
  - Simply copy the URL from the repo on Github
  - “git clone <URL>” **in your terminal**

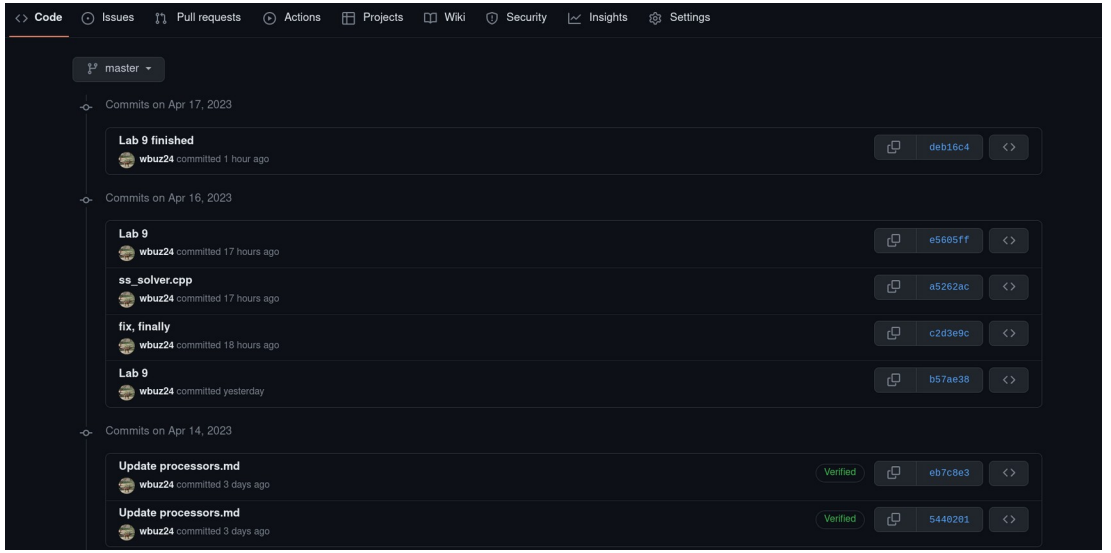


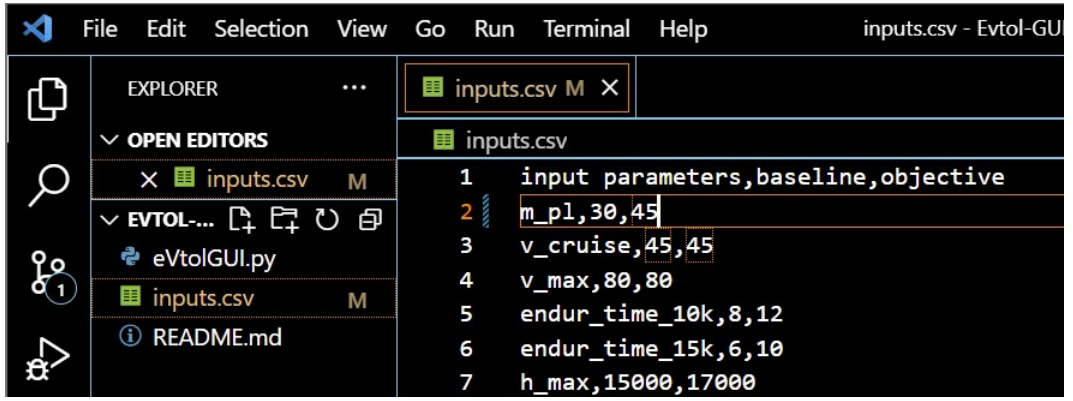

```
PS C:\Users\willb\Documents\Spring 2023\Research> ls


    Directory: C:\Users\willb\Documents\Spring 2023\Research


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/10/2023   8:37 PM                timesheets


PS C:\Users\willb\Documents\Spring 2023\Research> git clone https://github.com/wbuz24/Evtol-GUI.git
Cloning into 'Evtol-GUI'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (30/30), done.
Receiving objects:  25% (8/32)remote: Total 32 (delta 4), reused 0 (delReceiving objects:  37% (12/32)

Receiving objects: 100% (32/32), 15.60 KiB | 5.20 MiB/s, done.
Resolving deltas: 100% (4/4), done.
PS C:\Users\willb\Documents\Spring 2023\Research> ls


    Directory: C:\Users\willb\Documents\Spring 2023\Research


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/10/2023   8:46 PM                Evtol-GUI
d-----         1/10/2023   8:37 PM                timesheets
```

Git tracks your changes

- Now, you have a **local** copy of the repository you just cloned, and Git will monitor your changes

- Git is broken up into **sections** and files can be moved with various commands
  - The basic workflow of Git:
    - **Clone** a repository
    - **Modify** a file(s)
    - **Stage** a file(s)
    - **Commit** the file(s) to be stored in the repo
    - **Push** the branch to Github
    - Initiate a **pull request** on Github
    - Review and **merge** the branches on Github





- Untracked files are not stored in Git

“git status” will report the status of the working directory

- The “staging” area is for files that will be pushed in the commit
  - “git add <file>” **moves a file into the staging area**
  - “git restore --staged <file>” **removes a file from the staging area**
  - ‘git commit -m “<message>”’ **commits the staged area for pushing**
  - “git push <remote> <branch>” **pushes the branch to Github**

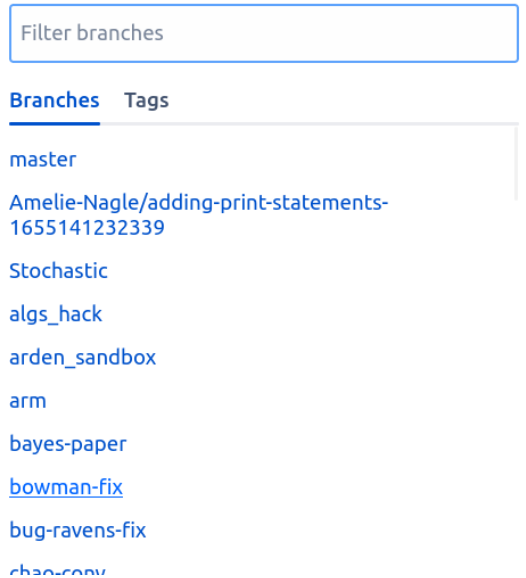
- “git help” will return an **overview of commonly used commands**

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   inputs.csv

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\willb\Documents\Spring 2023\Research\Evtol-GUI> git add inputs.csv
PS C:\Users\willb\Documents\Spring 2023\Research\Evtol-GUI> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   inputs.csv
```

```
These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects
```

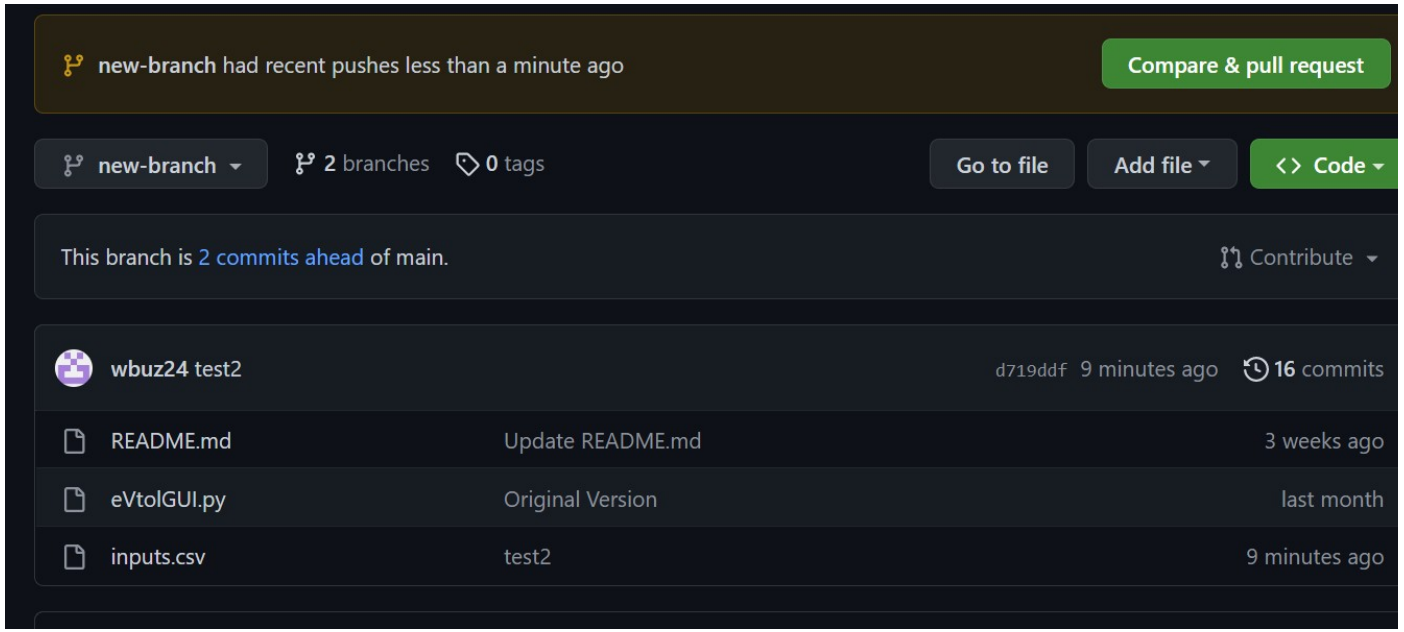
Branches create a safe copy to modify

- Before making major changes, you will want to **create a new branch**
  - **Switch to a new branch** with “git switch -c <branch-name>”
- This allows you to develop in isolation from the **main branch**
  - Branches can be merged to the main branch with a **pull request**
- “Pull requests” are notifications to collaborators that you have made changes to a branch and seek to **merge** it to the **main** branch
  - Collaborators would then **pull** these changes



git pull attempts to update your copy

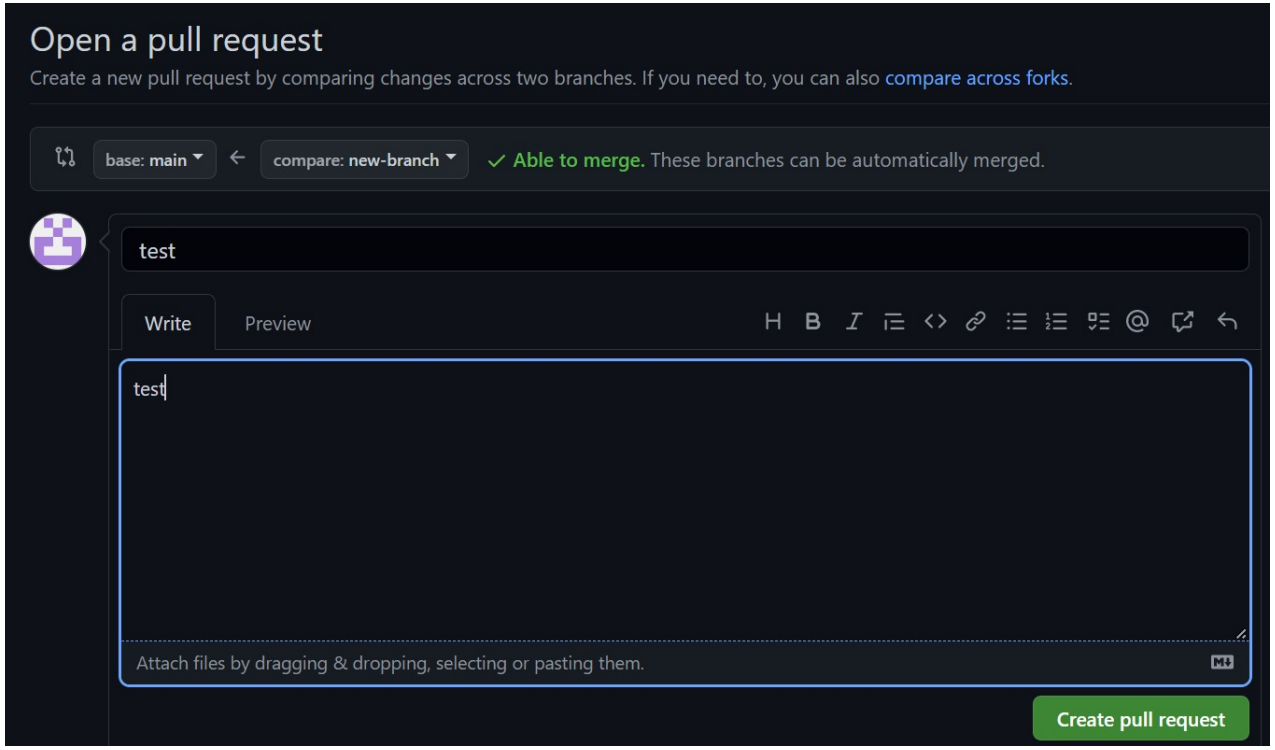
- Most frequently used commands:
  - “git pull <remote> <branch>” **updates Github’s branch copy into your local**
  - “git add <file>” **adds the file to the stage area**
  - “git commit –m “<message>”” **creates a new commit of what is in the stage area with a description**
  - “git push <remote> <branch>” **will push your branch to Github and can be seen online**




```
PS C:\Users\willb\Documents\Spring 2023\Research\Evtol-GUI> git status
On branch new-branch
Your branch is up to date with 'origin/new-branch'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   inputs.csv

PS C:\Users\willb\Documents\Spring 2023\Research\Evtol-GUI> git commit -m "test"
[new-branch 32a1187] test
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\willb\Documents\Spring 2023\Research\Evtol-GUI> git push origin
Enumerating objects: 41, done.
Counting objects: 100% (41/41), done.
Delta compression using up to 8 threads
Compressing objects: 100% (35/35), done.
Writing objects: 100% (41/41), 16.21 KiB | 16.21 MiB/s, done.
Total 41 (delta 10), reused 30 (delta 4), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
remote:
remote: Create a pull request for 'new-branch' on GitHub by visiting:
remote:      https://github.com/wbuz24/Evtol-GUI/pull/new/new-branch
remote:
To https://github.com/wbuz24/Evtol-GUI.git
 * [new branch]      new-branch -> new-branch
PS C:\Users\willb\Documents\Spring 2023\Research\Evtol-GUI>
```

Pull Requests notify collaborators for review

- After you have pushed your new branch to Github, you can initiate a pull request to review the code and merge with the main branch
  - Pull requests can be sent to individual collaborators



Merge conflict mitigation

- Merge conflicts are bound to happen when collaborating with other machines.
  - Can be reduced with proper Git usage

Pull before you modify to mitigate conflicts

- When a collaborator pushes a branch to Github and **merges** the changes with the **main** branch, use “git pull” to update your local copy of the main branch.
  - It is important that you **pull** new commits of the main branch **before** you push your branch to Github, **so you merge your collaborators work**

- "git pull" should be used when the master branch has **updated** to avoid **conflicts**, **non-fast forward** and other Git errors

  - When in doubt, check your:
    - Directory
    - Git Status
    - Branch
    - Commit status
    - Branch Head

How do I solve a merge conflict? → What are you trying to do?

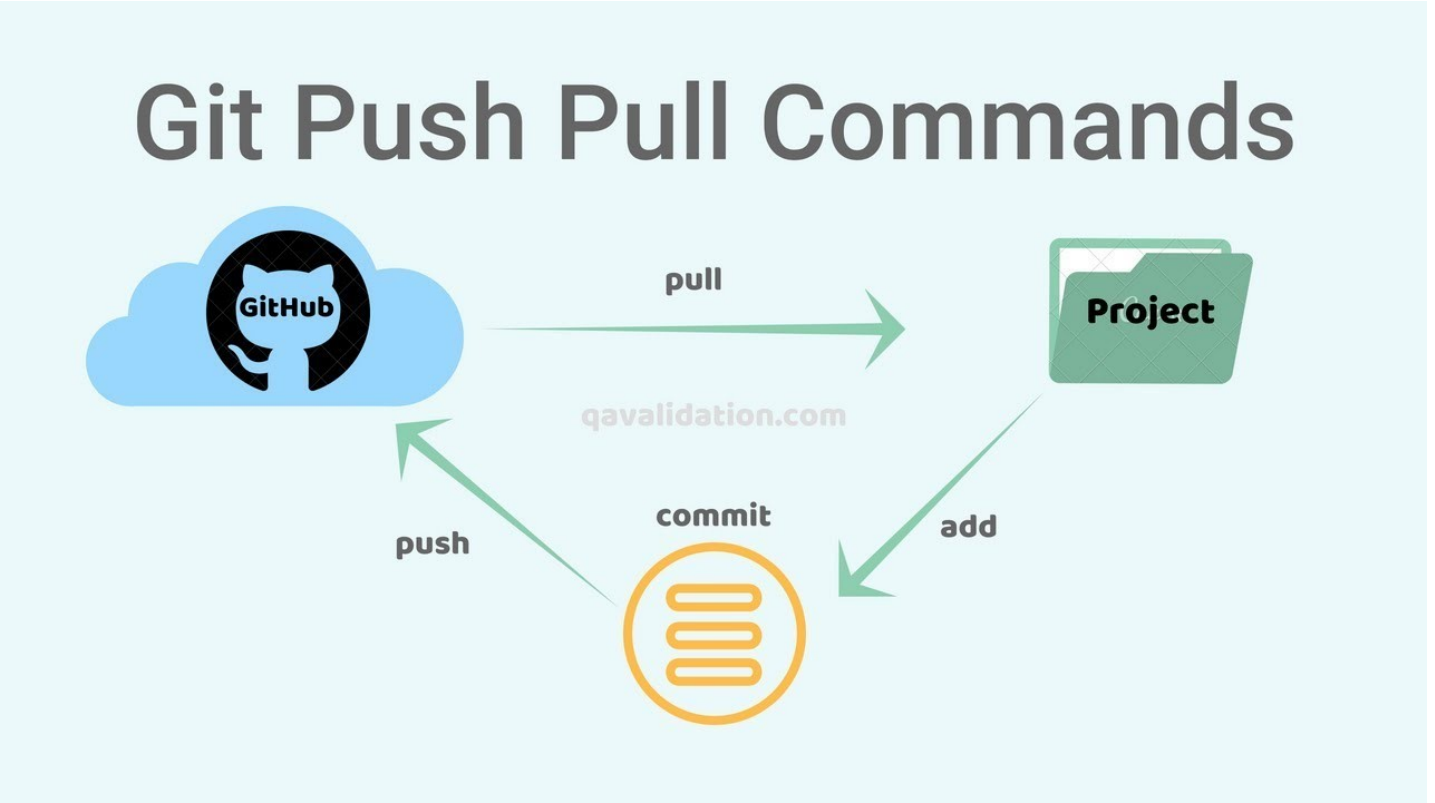
- Often, you can pull a branch into a new branch (isolated from what you are currently working on, to worm your way out of your mess and start fresh)
    - Commit your changes
    - "Checkout" a new branch with your desired branch
        - **git checkout –b <new-branch> <remote/branch>**

- You can cancel a merge during a conflict:
    - **git reset --merge**
    - Or
    - **git reset --hard HEAD** (will reset your branch head)

Staying up to date mitigates conflicts

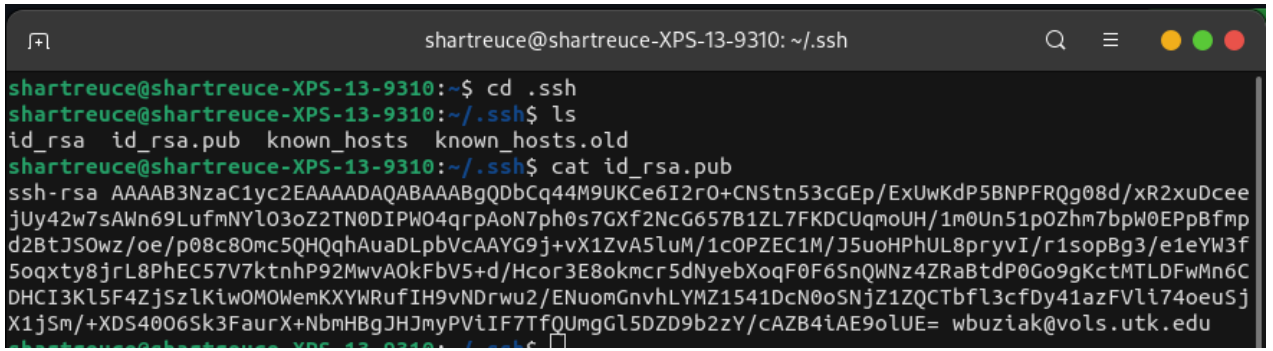
- Once your branch is merged to the main branch, it is safe to delete the branch that you had created.

- If your local repo is not up to date, you may experience a “non-fast-forward” error

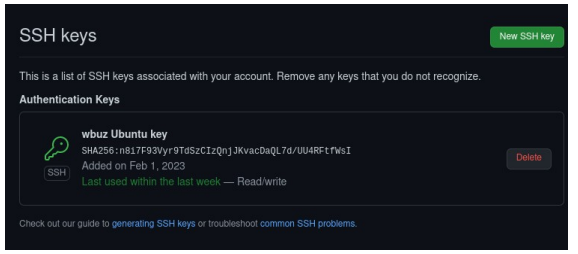


How to generate SSH

- In your terminal:
  - **ssh-keygen –C "<git.email>"**
    - Default settings work (just press enter until it generates)
  - Locate your ".ssh" folder (hidden directory)
    - **cd**
    - **ls –l -a**
  - **cat id_rsa.pub**
    - Copy the key



- On Github:
  - Navigate to Settings > SSH and GPG keys
  - Create a new key and paste your public key.

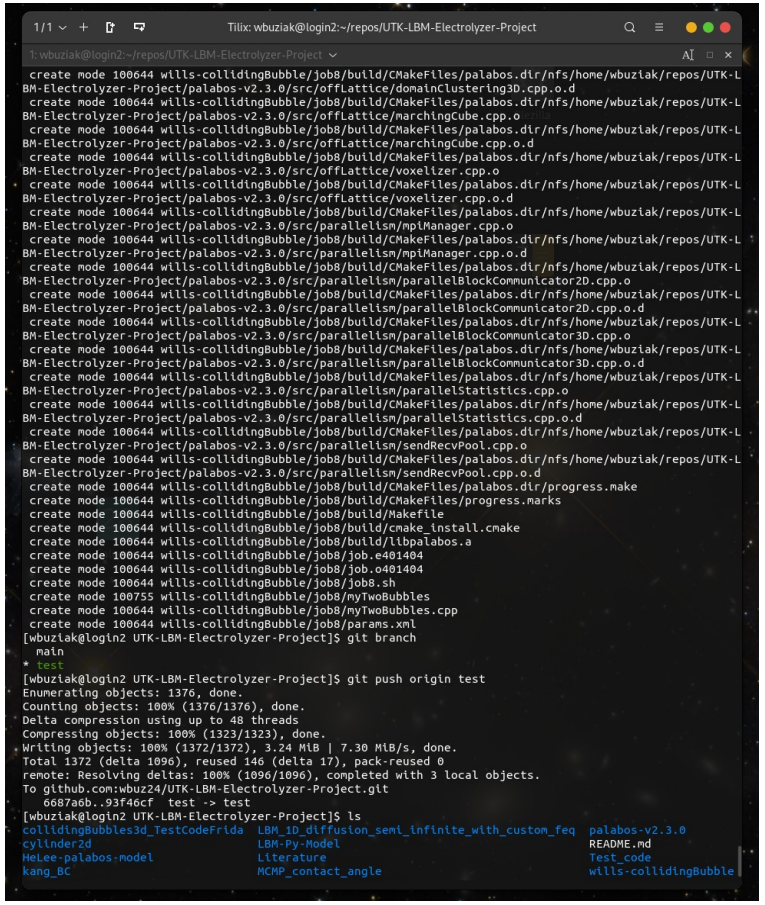


  - Git/Github should now be authenticated and no longer require a password

Structure your repository to keep it Light

- **.gitignore** tells git specific files in the directory to **not** track
  - This is a hidden file (can be found with "ls –l –a")
  - .gitignore files often include files that are only relevant to your local machine (graphs, images, individual log files, etc.)

- **.keep** tells git to track an otherwise empty directory
  - Git does not track empty folders by default
  - .keep files can be used to maintain a structure in a repository

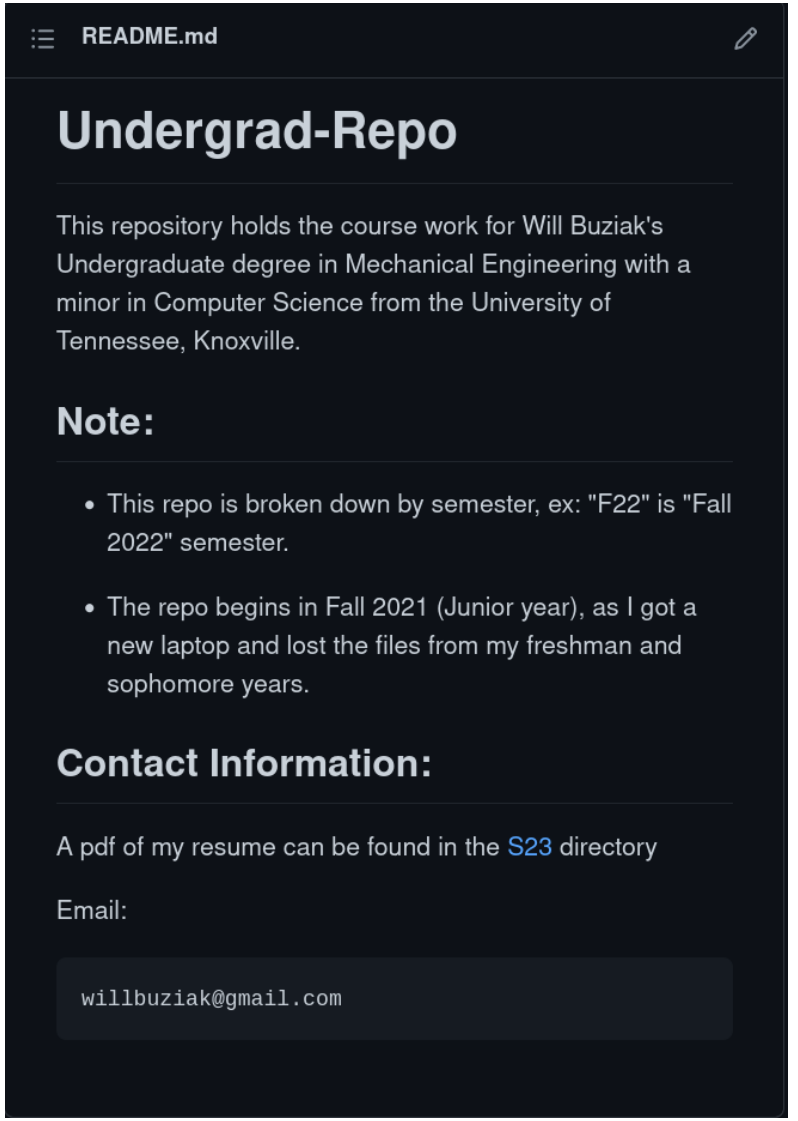


Repositories can quickly fill up with irrelevant files.

Improve your documentation

- Markdown "README.md" files are an excellent format for documenting repositories

- Markdown files support many different formats to highlight different aspects of your code and/or repository
  - Headers
  - Copy fields
  - Hyperlinks
  - Editable directly from Github


README.md

Undergrad-Repo

This repository holds the course work for Will Buziak's Undergraduate degree in Mechanical Engineering with a minor in Computer Science from the University of Tennessee, Knoxville.

Note:

- This repo is broken down by semester, ex: "F22" is "Fall 2022" semester.
- The repo begins in Fall 2021 (Junior year), as I got a new laptop and lost the files from my freshman and sophomore years.

Contact Information:

A pdf of my resume can be found in the S23 directory

Email:

```
willbuziak@gmail.com
```

Other Useful Commands - Naming and navigation

- “git remote -v” will show all the remotes you have locally and their names
  - “git remote” can also be used with:
    - add <name> <URL>
    - rename <old> <new>
    - remove <name>
- “git switch <branch>” switches to a branch
  - “-c” flag creates a new branch
- “git rm” removes files from a working tree
- “git restore” restores a working tree’s files
- **“git help” lists common commands**
- "git branch - a"
  - lists current branch heads
- "git checkout –b <new-branch> <origin/branch>
  - Creates a new branch based on an existing branch
- "git merge"
  - Merges with a given branch
  - "--quit" flag can cancel the current merge during conflicts
- "git config"
  - Configures your system, default is to the working repo
  - "--global" applies the config to the entire machine

References

- **Github Docs**

https://docs.github.com/en

https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

- **Tennlab Neuromorphic Research Git presentation**

https://bitbucket.org/neuromorphic-utk/neuro_resources/src/master/presentations/git_presentation/images/

- **“Push a New Local Branch to a Remote GitHub repo with Git”**

https://www.youtube.com/watch?v=se1WitSPKwc

- **"Difference between git PULL and git FETCH"**

https://www.youtube.com/watch?v=Mdo7hvlUJ-U

- **Connecting to Github with SSH**

https://docs.github.com/en/authentication/connecting-to-github-with-ssh

- **How to Fix Git error: you need to resolve your current index first**

https://unfuddle.com/stack/tips-tricks/git-error-you-need-to-resolve-your-current-index-first/

- **Pull a Remote Branch into a New Branch**

https://coderwall.com/p/_gepwq/pull-a-remote-branch-into-a-new-branch