



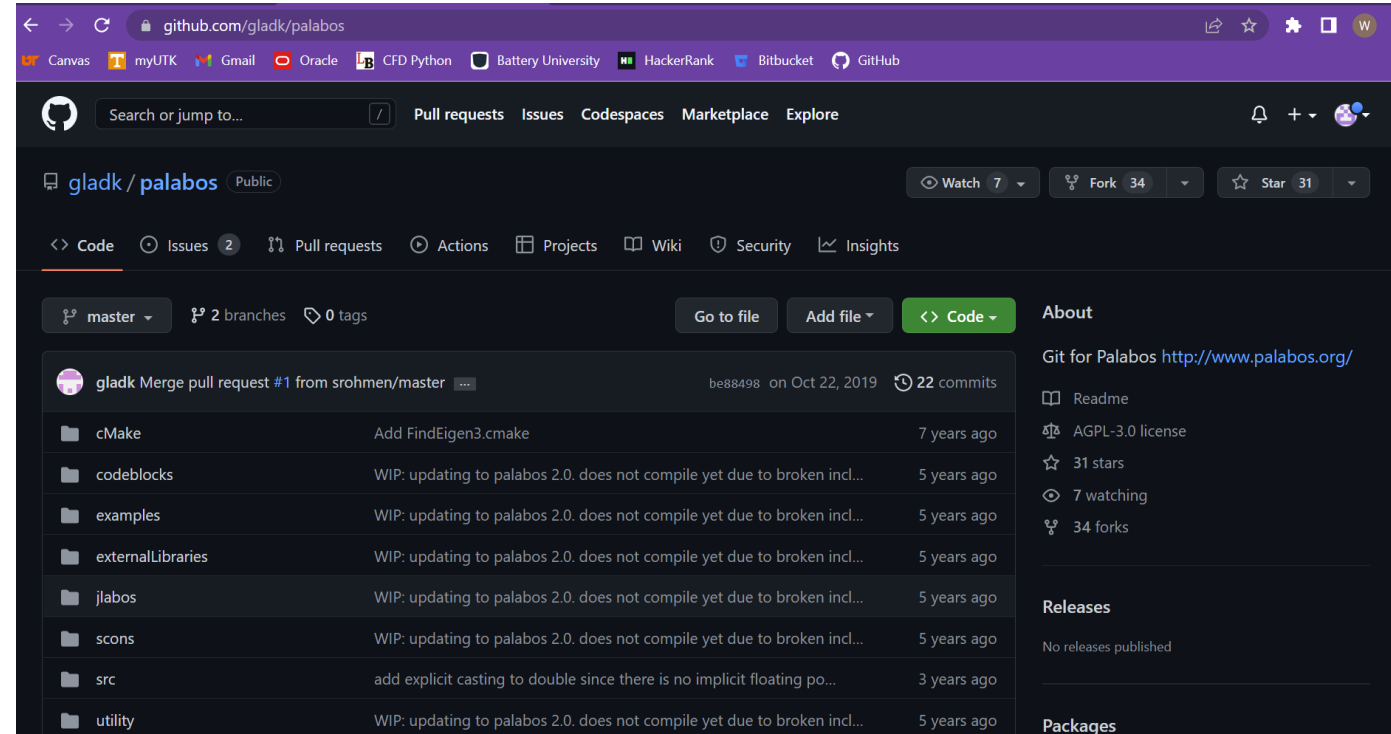
# Git/Github

introduction/tutorial

Will Buziak  
1/11/23

# What is Git/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

# Why Git?

---

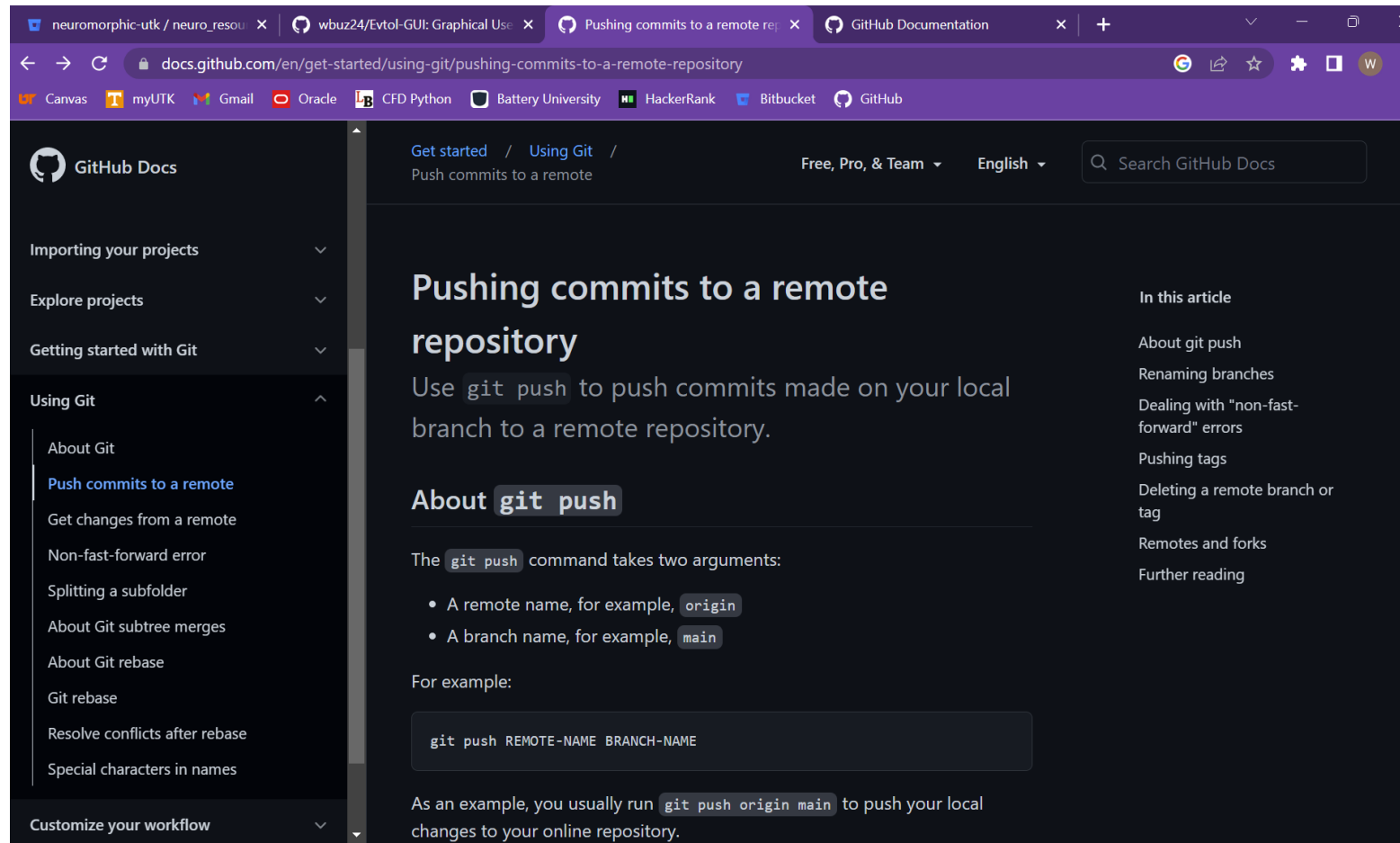
- Git was designed for **version control**. Meaning that there is a well-established record of changes made to each file.
  - Git allows “cloning” entire repositories of files that can run on a local machine quickly and seamlessly.
- Github repos can be specified as public or private and can invite collaborators 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
  - 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
  - Proper Git usage mitigates **overwriting**
  - Allows for multiple “branches” representing different test versions of the same repo that can be merged
  - Supports “forking” which allows **collaborator** rights on public repositories they did not previously have writing access to.

# *Github Docs is your friend*

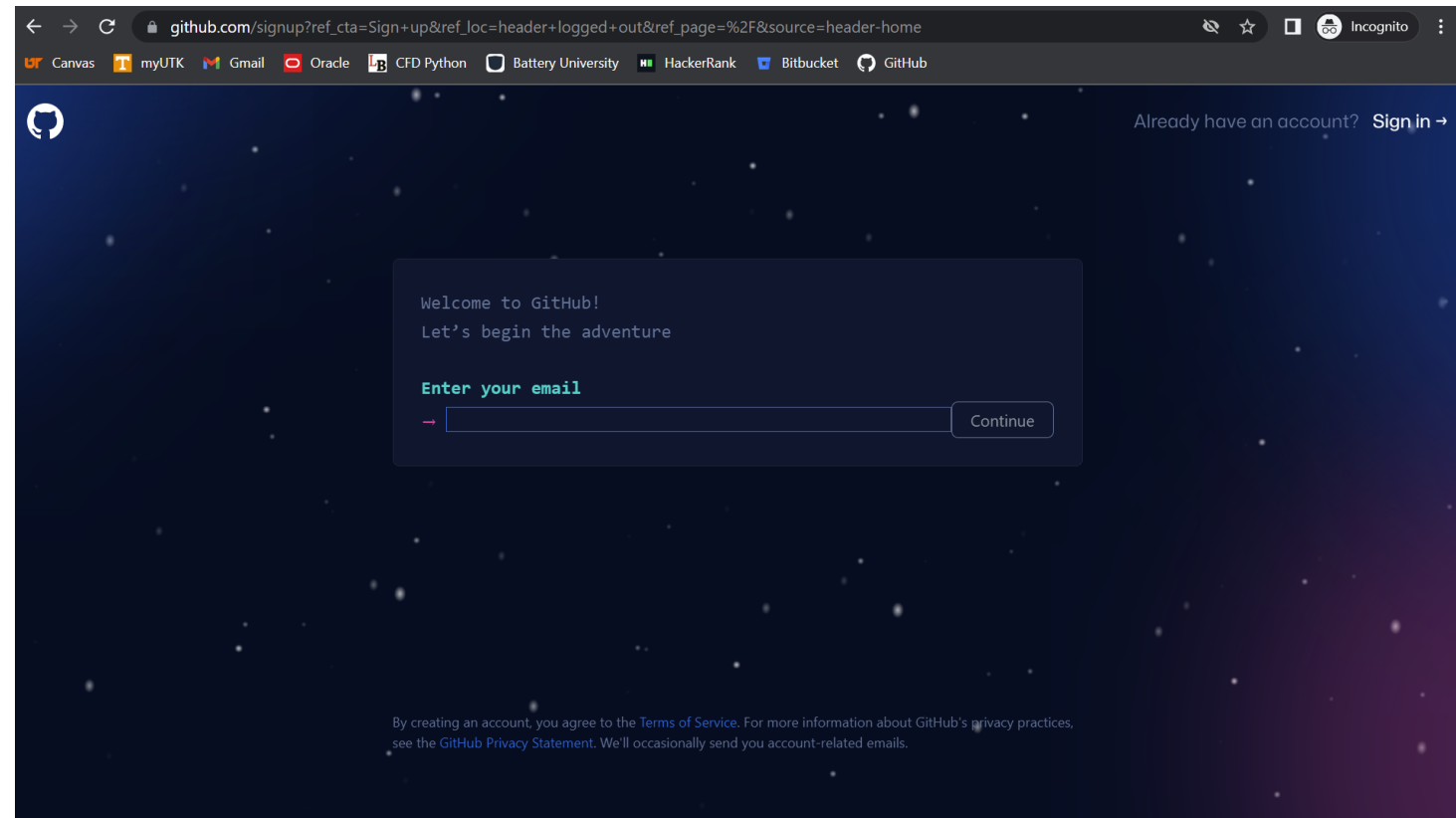
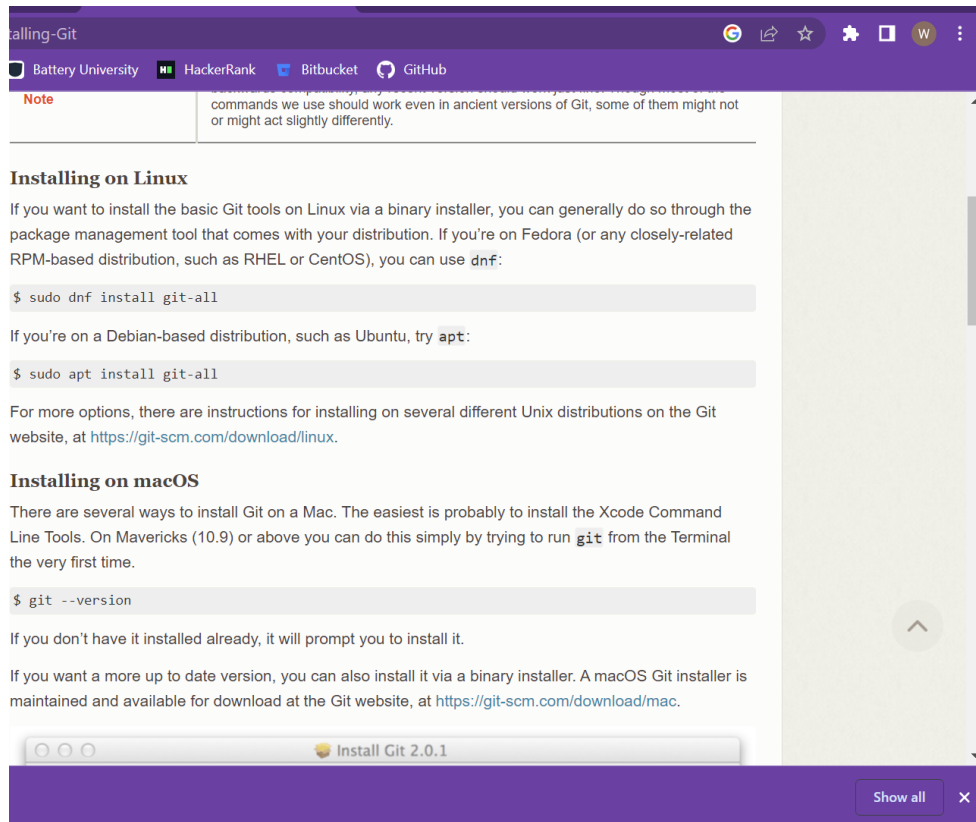


## Note:

Some Git commands perform similar tasks, typically intending to break up the processes of a more confusing command

# Getting started with Git

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



# Connect your local Git account with Github

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

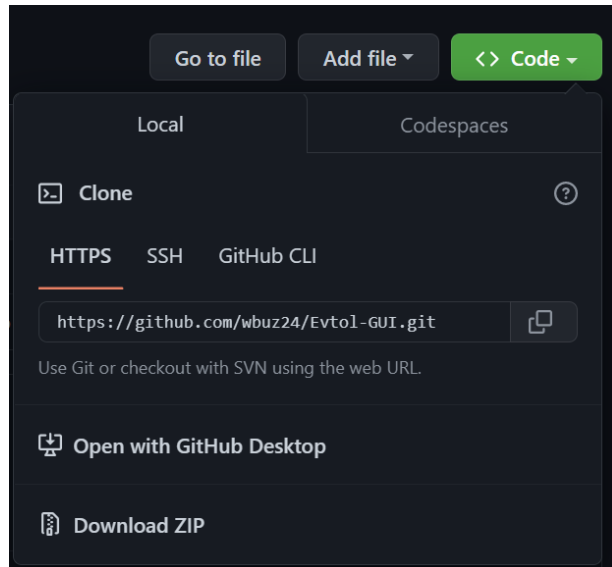
```
PS C:\Users\willb\Documents\Neuromorphic> git config --global user.name "wbuz24"  
PS C:\Users\willb\Documents\Neuromorphic> git config --global user.email wbuziak@vols.utk.edu  
PS C:\Users\willb\Documents\Neuromorphic> 
```

- 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



# Methods of cloning from a remote 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

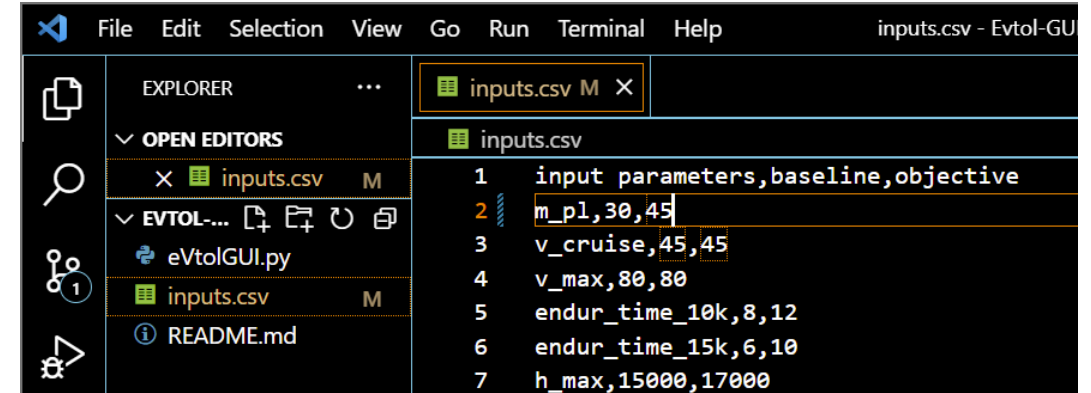
Directory: C:\Users\willb\Documents\Spring 2023\Research

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



The screenshot shows a code editor window titled 'inputs.csv - Evtol-GU'. The editor has a menu bar with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. On the left, there is an 'EXPLORER' sidebar showing a file tree with 'OPEN EDITORS' containing 'inputs.csv M' and 'EVTOL-...' containing 'eVtolGUI.py', 'inputs.csv M', and 'README.md'. The main editor area shows the content of 'inputs.csv' with the following text:

```
1 input parameters,baseline,objective
2 m_p1,30,45
3 v_cruise,45,45
4 v_max,80,80
5 endur_time_10k,8,12
6 endur_time_15k,6,10
7 h_max,15000,17000
```

# “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

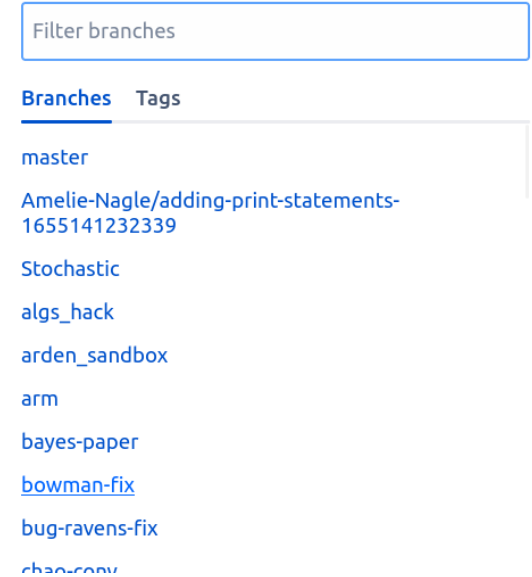
|        |   |
|--------|---|
| branch | List, create, or delete branches                            |
| commit | Record changes to the repository                            |
| merge  | Join two or more development histories together             |
| rebase | Reapply commits on top of another base tip                  |
| 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                   |

# Create a new branch when pushing to Github

- 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 switch -c <branch>” creates a new branch

- After changes to a file are made:
  - “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**
    - Try “git push --set-upstream <remote> <branch>” to set your upstream push to a default branch

The screenshot shows a GitHub repository page for a user named 'wbuz24' on a branch named 'test2'. At the top, a notification bar states 'new-branch had recent pushes less than a minute ago' with a 'Compare & pull request' button. Below this, the repository name 'wbuz24 test2' is shown, along with '2 branches' and '0 tags'. A status bar indicates 'This branch is 2 commits ahead of main.' and a 'Contribute' button. The file list includes 'README.md' (updated 3 weeks ago), 'eVtolGUI.py' (original version, last month), and 'inputs.csv' (test2, 9 minutes ago). A 'Go to file' button and 'Add file' dropdown are also visible.

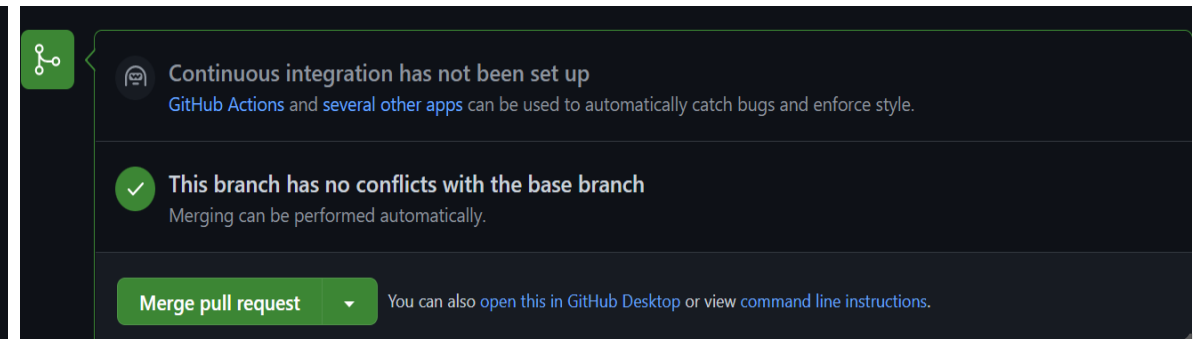
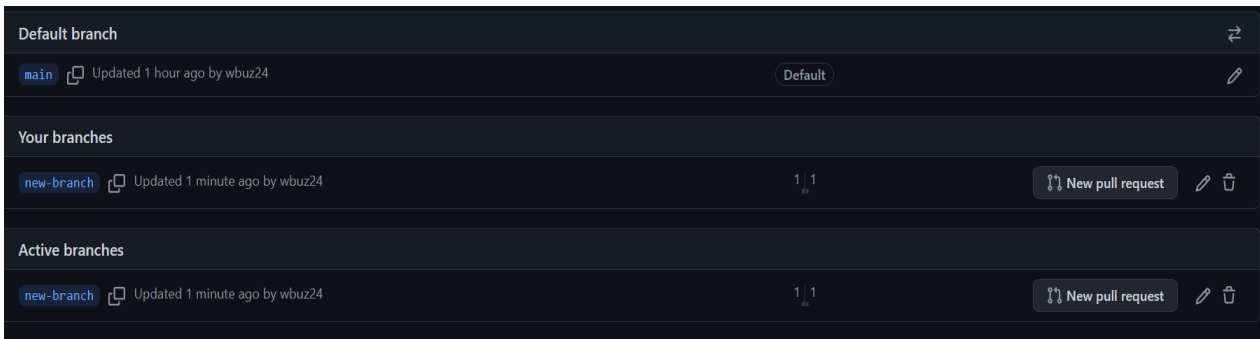
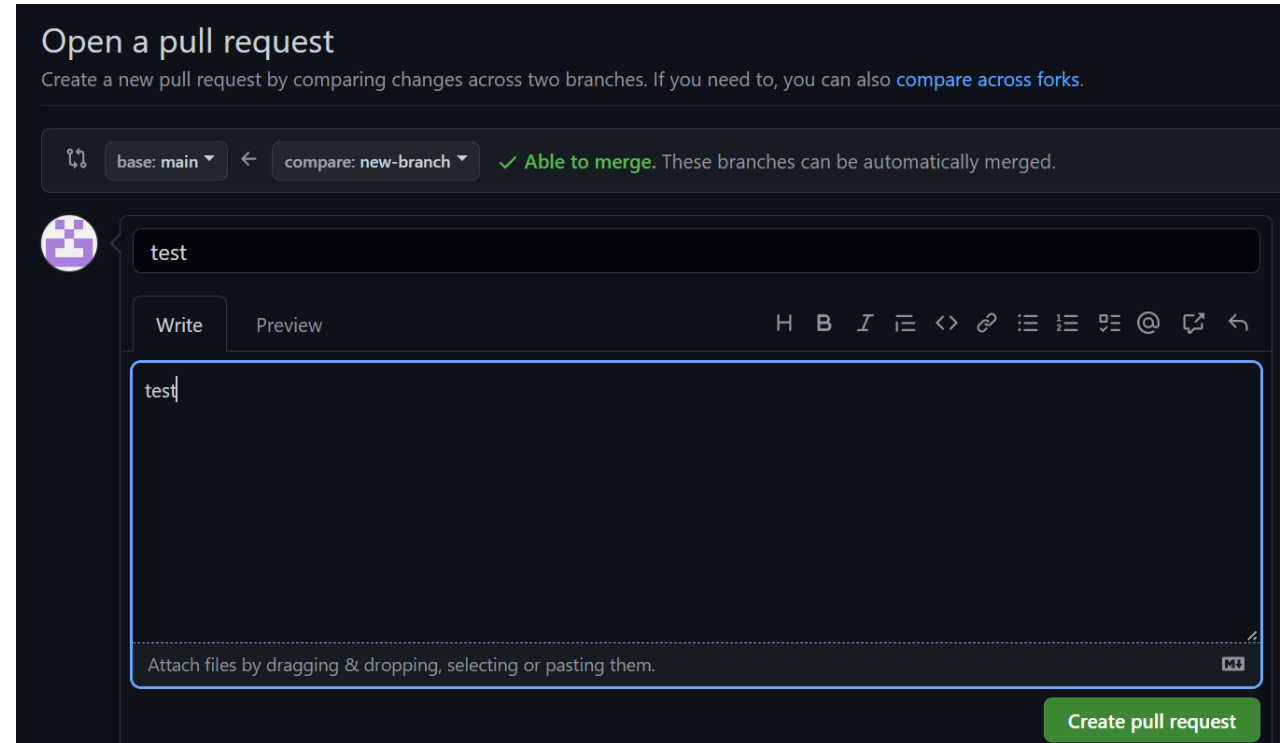
```
PS C:\Users\willb\Documents\Spring 2023\Research\Evtol-GUI> git status
On branch new-branch
Your branch is up to date with 'origin/new-branch'.

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



# 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 main has **updated** to avoid overwriting, non-fast forward and other Git errors
  - When in doubt, check your:
    - Directory
    - Git Status
    - Branch
    - Commit status
    - Branch head

# Commit before you merge to avoid conflicts

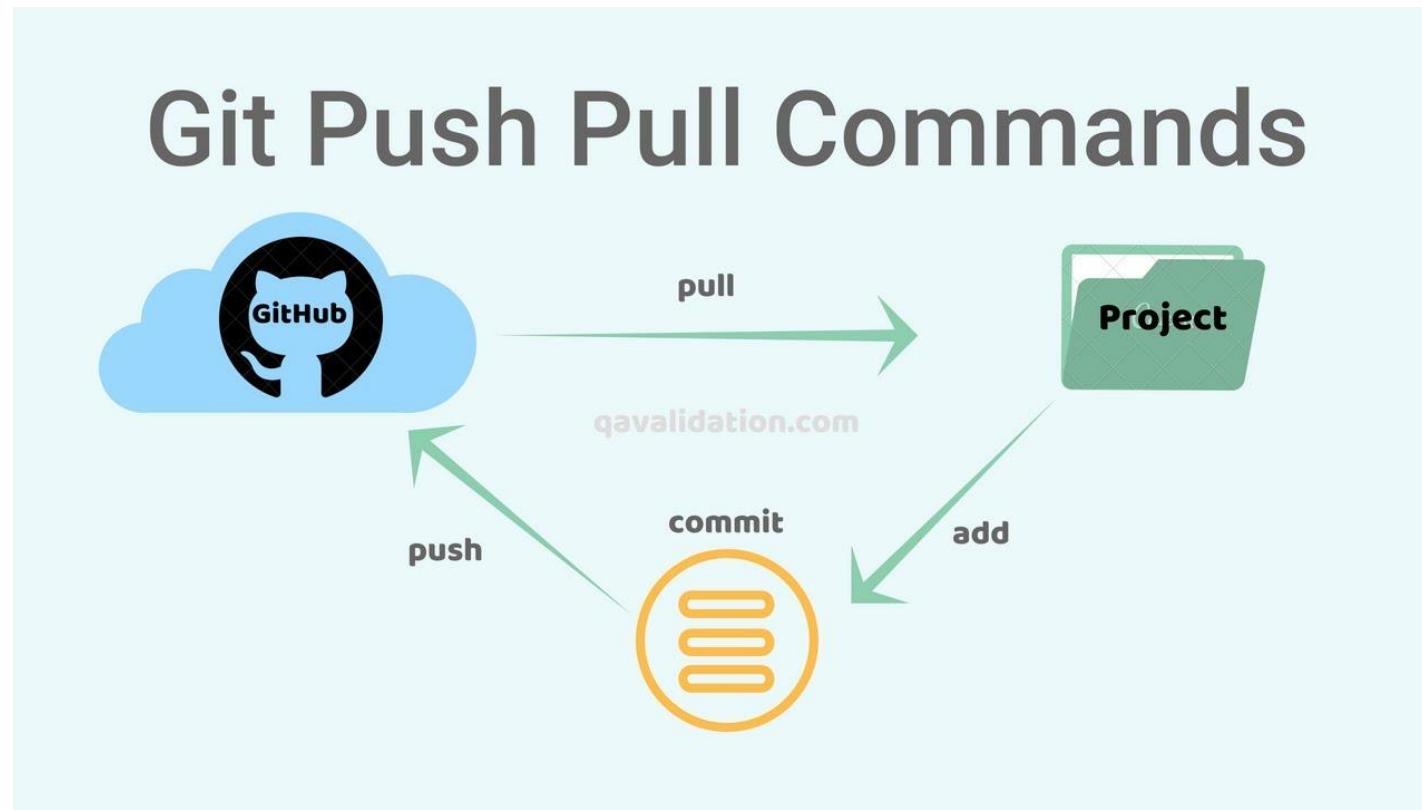
---

- Merge conflicts are inevitable and the quickest solution depends on what you are trying to do.
  - Often, you seek to pull a branch into a new branch (isolated from what you are currently working on)
    - 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)



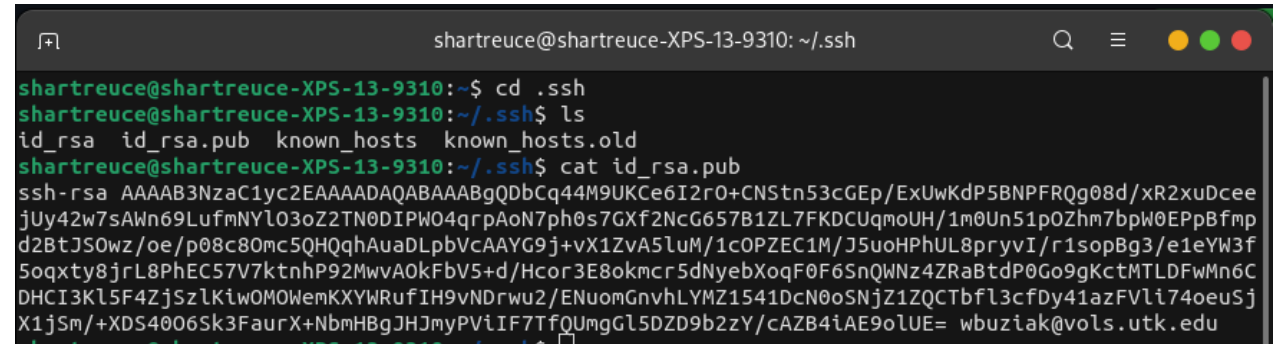
# Proper Git usage prevents overwriting

- 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



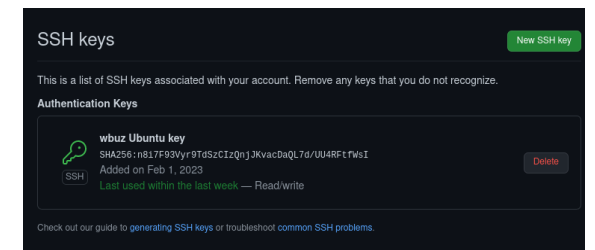
# Generating SSH Keys

- 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



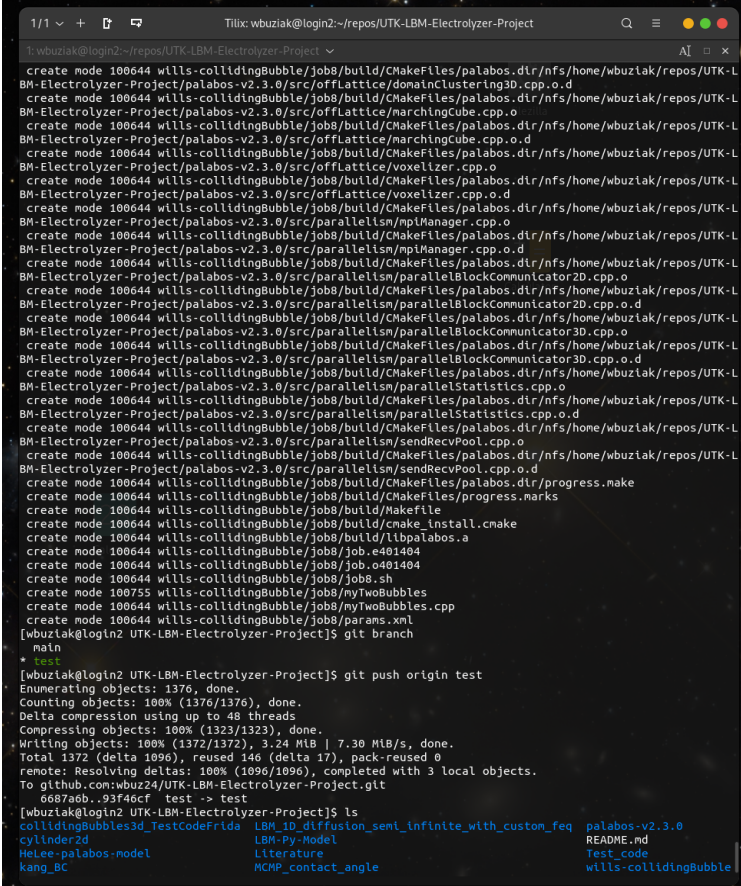
```
shartreuce@shartreuce-XPS-13-9310: ~/.ssh
shartreuce@shartreuce-XPS-13-9310:~$ cd .ssh
shartreuce@shartreuce-XPS-13-9310:~/.ssh$ ls
id_rsa id_rsa.pub known_hosts known_hosts.old
shartreuce@shartreuce-XPS-13-9310:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQBcQ44M9UKCe6I2r0+CNSn53cGEp/ExUwKdP5BNPFRQg08d/xR2xuDCee
jUy42w7sAWn69LufmNYl03oZ2TN0DIPW04qrpAoN7ph0s7GXf2NcG657B1ZL7FKDCUqmoUH/1m0Un51p0Zhm7bpw0EPpBfmp
d2BtJS0wz/oe/p08c80mc5QHqghAuaDLpbVcAAYG9j+vX1ZvA5luM/1cOPZEC1M/J5uoHPHUL8pryvI/r1sopBg3/e1eYW3f
5oqxytj8rL8PhEC57V7ktnhP92MwvA0kFbV5+d/Hcor3E8okmcr5dNyeBxOqF0F6SnQWNz4ZRaBtdP0Go9gKctMTLDFwMn6C
DHCI3Kl5F4ZjszLkiwOMOWemKXYWRufIH9vNDRwu2/ENuomGnvhLYMZ1541DcN0oSNjZ1ZQCTbfl3cfDy41azFVli74oeuSj
X1jSm/+XDS4006Sk3FaurX+NbmHBgJHJmyPVlIF7TfQUmgGL5DZD9b2zY/cAZB4iAE9oLUE= wbuziak@vo1s.utk.edu
```

- 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



# .gitignore & .keep files keep your repo clean

- .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

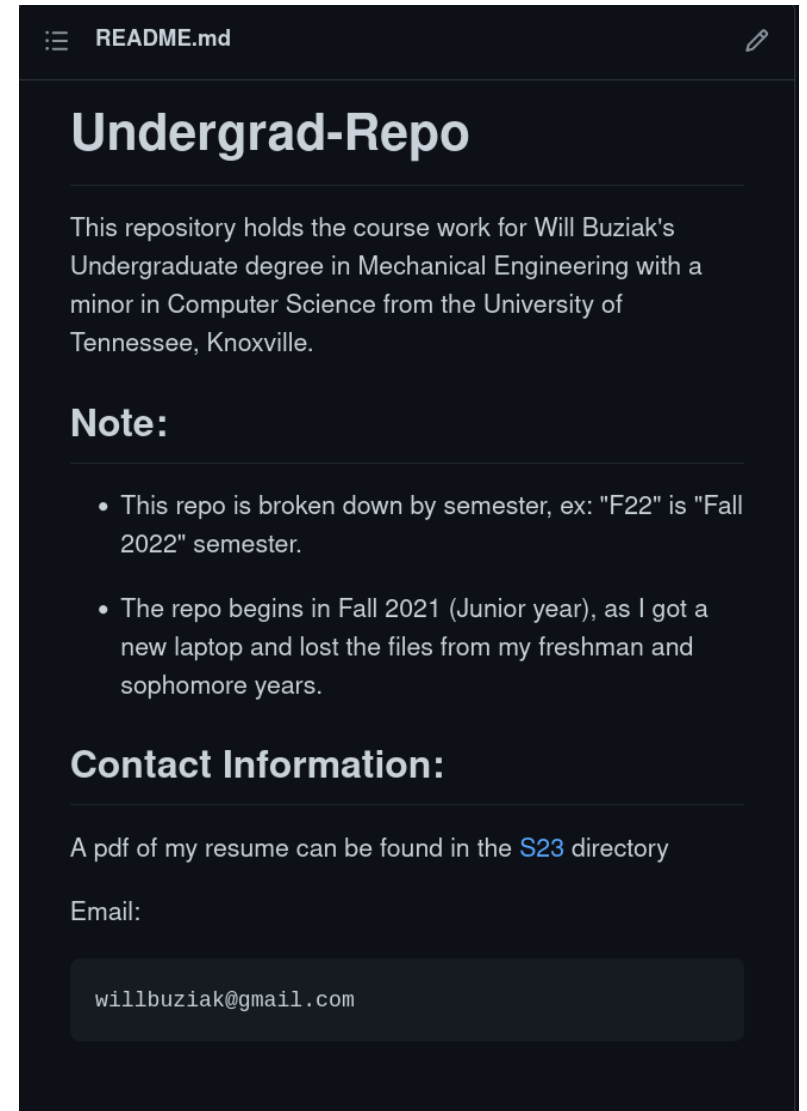


```
1/1 + 0 0 Tili: wbuziak@login2:~/repos/UTK-LBM-Electrolyzer-Project
wbuziak@login2:~/repos/UTK-LBM-Electrolyzer-Project$ git push origin test
Enumerating objects: 1376, done.
Counting objects: 100% (1376/1376), done.
Delta compression using up to 48 threads
Compressing objects: 100% (1323/1323), done.
Writing objects: 100% (1372/1372), 3.24 MiB | 7.30 MiB/s, done.
Total 1372 (delta 1096), reused 146 (delta 17), pack-reused 0
remote: Resolving deltas: 100% (1096/1096), completed with 3 local objects.
To github.com:wbuz24/UTK-LBM-Electrolyzer-Project.git
  6687a5b..23f46cf test -> test
wbuziak@login2:~/repos/UTK-LBM-Electrolyzer-Project$ ls
collidingBubbles3d_TestCodeFrida  LBM_1D_diffusion_semi_infinite_with_custom_fed  palabos-v2.3.0
cylinder2d                        LBM-Py-Model                                     README.md
HeLee-palabos-model               Literature                                         Test_code
kang_BC                           MCMC_contact_angle                             wills-collidingBubble
```

Repositories can quickly fill up with irrelevant files.

# README.md files document your repo

- 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



# 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/](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=Mdo7hvIUJ-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](https://coderwall.com/p/_gepwq/pull-a-remote-branch-into-a-new-branch)