**Powershell commands**

* pwd
  + Print working directory
* dir
  + Shows all files in folder
* mkdir
  + Make new folder
* mv
  + Move folder
  + mv folderA/myfile.py folderB/
  + moves myfile from folder A to folder B
* rm
  + Remove directory or folder
* cd
  + Is change directory
  + cd .. drops down one level
  + cd takes you back to root
* echo
  + echo “text” > file.txt
    - This creates file.txt and writes “text” in it
    - This also overwrites the file
  + echo “text” > file.txt
    - This adds “text” to file.txt
* type
  + type file.txt shows the content of file.txt
* code
  + code
    - This opens the file in VS code
    - code file.txt would open the file in VS code
* copy
  + copies file
* del
  + deletes file
* notepad
  + notepad file.txt will open file.txt or create the file in notepad if it does not exist
* ren
  + renames files - ren file.txt newname.txt

**Git and GitHub**

* Git username Plan-kton
  + https://github.com/Plan-kton
* Git is a local tool that tracks changes on your computer
  + Git manages your local history
* GitHub is a website that store Git repositories in the cloud and adds collaboration features
  + GitHub is where you can push and pull that history to the cloud

**Git**

* git init
  + this initializes the file. It tells git to start tracking this folder
* git status
  + tells you what files if any have been changed
  + shows modified but not staged files
* git add
  + This stages the file. It tells git to include this file in the next commit
  + git add. for all files in folder or git add ‘filename’
  + these are the files in the folder that you want to push
* git commit -m “message”
  + this save a version with the changes
  + It is a snapshot of the code when you commit
* git diff or git diff --staged
  + shows unstaged changes
  + –- staged shows changes staged for commit
* git log
  + show previous commits, authors, messages and timestamps
* git merge
  + This merges two branches master/main and other branch
  + To use git merge you must first switch back to main/master and then run merge
* git branch
  + lists all of the branches and shows the current branch with an \*
* git branch new\_branch
  + will create a new branch
  + git branch “new\_branch” creates new\_branch branch
* git branch -d new\_branch
  + after merging you can delete the branch -d is for delete
  + git branch -D “new\_branch” force the delete even it it is not merged
    - use this if you are 100% sure you don’t want that branch or it was just a throwaway experiment
* git branch -m master main
  + will rename from master to main
  + the -m is really for renaming. It is moving the label from one label to another
  + git branch -m old\_name new\_name
  + If you are in the branch then you can simply do git branch -m new\_name
* git switch
  + switch branches
  + git switch new\_branch takes you from main to new\_branch
  + -c is for create. git switch -c new\_branch will create new\_branch and switch you to that branch

**Other Git**

* git rm
  + removes tracked file
* git mv
  + renames tracked files
* git config –global user.name
  + This shows what user name is connected to this folder
* git config –global user.email
  + This shows what email is connected to this folder
* git config user.name
  + This checks the local machine
* git config user.email
  + This checks the local machine

**GitHub**

GitHub is the central hub your local git talks to. It is the central hub for all the coders working on your project. It keeps track of changes from each person contributing to the project. See GitHub common practice below.

**Create repository on GitHub**

1. Go to github.com
2. Click new repository in the top right of the screen
3. Fill in:
   1. Repo name
   2. Description
   3. Public or private
4. Leave the “initialize with README” unchecked if you already have local code
5. Click create repository

**Once we have repository build, connect your local project to with GitHub**

1. Git init which tells git to pay attention to this local folder
   1. git init
   2. git add .
   3. git commit -m “initial commit”
   4. git remote add origin [git@github.com:Plan-kton/forecast.git](mailto:git@github.com:Plan-kton/forecast.git)
   5. git remote -v (to confirm connection. The v is for verbose and means give me more details)
   6. git push -u origin main (the -u flag links your local main branch with origin/main. The u is upstream and means you are connecting your local main folder to the origin/main folder upstream)

**GitHub Concepts**

* Branch
  + A separate line of development. Often used for features, fixes, experiments, etc.
  + Allows developers to try different things without corrupting main
* Pull Request
  + A request to merge your branch into another branch (usually main, master, or dev) on GitHub.
  + This allows others working on the project to review code and discuss.
* git fetch
  + downloads branch updates from GitHub but does not merge them
  + compare to git pull. git pull both fetches and merges changes into your local branch
  + fetch is used when trying to debug a GitHub conflict. This is when two people make a change to the same code and GitHub does not know which code to use
* git remote -v
  + A pointer to the GitHub repo. Usually named origin
  + This shows if Git is connected to GitHub
  + This shows the git hub path
    - origin [git@github.com:Plan-kton/cpi\_fah\_fcst.git](mailto:git@github.com:Plan-kton/cpi_fah_fcst.git)
  + -v is for verbose and means provide more detail
* git remote set-url origin [git@github.com:YOUR\_USERNAME/NEW\_REPO\_NAME.git](mailto:git@github.com:YOUR_USERNAME/NEW_REPO_NAME.git)
  + This update your local remote to point to the git hub repository
  + The local remote must be point to an active repository to push
* git clone
  + clones repository
  + Copies a GitHub repo to your local machine
* fork
  + Your own copy of someone else’s repo. Useful for open source
* To authenticate
  + origin [git@github.com](mailto:git@github.com): username/project-name.git
  + origin [git@github.com](mailto:git@github.com): Plan-kton/cpi\_fah\_fcst.git
  + It will return Hi your-username! You've successfully authenticated... if everything lines up as expected
* Connect local Git to GitHub
  + git remote add origin
* Set global git identity to match main account
  + In VS code…
  + git config –global user.name “Your Name”
    - This is Eric Karlson and should not match my git username Plan-kton
  + git config –global user.email “email” (associated with the git hub acct)
    - This should match the github email however

**GitHub Common Practice**

* If multiple people are working on a project, they should not be working on the same piece of code. Git does not know which changes to accept
* Each user should create a branch that relates to what they are doing
  + Then when complete, this branch should be reviewed to make sure nothing conflicts
  + Then when approved, changes can be merged with main
* Before changes are merged with main, the user should pull first to make sure they have the most recent version of main. Then push