**Git Commands for Terminal**

Git add – add files in your working directory (pc) to git

* “Git add .” – adds all files in the folder (keep space between add and period)
* Git add . <file> – will add just the one file written (keep space between add and period)

Git commit - created a save point to add the current saved items to git

* “Git commit –m “Must add comment for the commit”

Got remote add origin <pathway link to repo is provided by GitHub> – If you already created a local file, create a new repo in GitHub and it will give you a command with link for your terminal. This will link the folder to the empty online repo

Git push –u origin master

**Introduction to Git and Version Control**

1. git --version (tells you what version you have installed)

2. git config --global user.name "Your Name" (configure username)

3. git config --global user.email yourname@example.com (configure email)

4. git config --list (display user and email configurations)

**Building your first repository**

1. git init (initialize git in local directory)

2. git remote add origin git@github.com:Username/Repository-Name (connect your local repository to github)

3. git status (see which files have been changed, and which ones you're adding)

4. git add fileName (track file, stage it for commit)

5. git checkout fileName (discard file changes you don't want to add)

6. git commit -m "Your Message here" (commit your changes, attach a message, create save point)

7. git push origin master (send changes from your computer to GitHub)

8. git reset HEAD fileName (un-stage a file to prevent commiting it)

**First Collaborative Exercise + Basic Workflow**

1. touch .gitignore (create .gitignore file)

2. git pull origin master (pull latest changes from your GitHub repository)

3. git remote -v (view remote connections)

**All about Branching**

1. git branch (view all existing branches on your local repository)

2. git branch branchName (creates a new branch named branchName)

3. git checkout branchName (switch from current branch onto a branch named branchName)

4. git checkout -b branchname (creates a new branch named branchName and switches to it, combination of above two commands)

5. git branch -d branchName (delete a branch named branchName)

6. git merge branchName (pull changes FROM branch named branchName INTO the branch you're currently on)

**Dealing with merge conflicts, remote repositories, and developing as a team.**

1. git remote add repoName https://github.com/Username/Repository-Name.git (allows multiple remote repositories)

2. git fetch repoName branchName (fetch changes from remote repo WITHOUT automatically merging them)

3. git pull repoName branchName (get changes, pull and merge them onto local machine)

4. git remote set-url repoName git@github.com:Username/Repository-Name (allows you to change the url for your GIT to GitHub connection)

5. git remote rm repoName (remove a remote repository)

**When things go wrong, and how to fix them.**

1. git log (view commit history, messages, and SHA ids for each commit. type 'q' to quit)

2. git reset --soft HEAD^ (roll back to previous commit, leave changes staged)

3. git reset --hard HEAD^ (roll back to previous commit, unstage changes)

4. git reset --hard theSHAid (roll back to any commit. WARNING: re-writes git history)

5. git revert --no-commit theSHAid..HEAD (revert from HEAD to any previous commit of your choice, just replace theSHAid with the actual SHA hash from the git log)

6. git revert --continue (continue forward with your revert. Creates revert commit)

7. git pull repoName theSHAid (pull previous commit from remote repository)

8. git commit --amend -m "new commit message" (change your previous commit message)

**Git workflow 2.0, debugging and organizing larger GIT projects**

1. git diff branch1..branch2 (compare differences between two branches)

2. git diff branch1...branch2 (compare branch1 and branch2 against their parent branch)

3. git diff branch1..branch2 >branchlog.txt (compares two branches, creates branchlog.txt if it doesn't already exist, and prints the output of the branch differences to the text file)

4. git diff HEAD (compare current directory to last commit)

5. git push repoName --delete branchName (delete a branch named branchName that exists on your remote repository)

6. git rebase repoName/branchName (merge from repoName branchName into your current branch without creating a merge commit. places your commits at the top in the history line)

7. git rebase -i HEAD~5 (combine previous 5 commits into one commit. WILL OPEN VIM, use esc, i, :w, :q)

**Documenting/tagging your code + GIT commands in depth.**

1. git tag -a v0.0 -m "message" (assign a version number of 0.0 and a message to a commit.)

2. git tag (view existing tags)

3. git show v1.0 (show version 1.0 and associated information)

4. git tag -d v0.0.1 (remove a tag)

5. git --help (brings up help menu and displays common options)

6. git commandName --help (display full manual on usage of specified command. Press q to exit doc)