Commands

- git log
    - Shows commit history
    - git log --graph --oneline <branch_name1> <branch_name2>
        - Visual representation of the commit history of <branch_name2> in relation to <branch_name1>
- git checkout <commitID>
    - Reverts git to the commit with <commitID>
    - git checkout master
        - Returns to the master branch
    - git checkout <branch_name>
        - Switches to branch <branch_name>
    - git checkout -b <branch_name>
        - Creates a new branch with name <branch_name> and switches to that branch
- git diff <commitID1> <commitID2>
    - Shows the differences between files in <commitID1> and <commitID2>
    - commit1 taken as the original file and commit2 taken as new file
    - git diff
        - Compares difference in files between staging area and working directory
    - git diff --staged
        - Difference between most recent commit and staging area
- git status
    - Shows which files are in the staging area, which files are untracked
- git add
    - Adds files to the staging area
- git commit
    - Commits the files in the staging area to the git log
    - Each commit should only contain 1 logical change
    - git commit --amend
        - Allows you to edit the commit message for the most recent commit
- git branch <branch_name>
        - Creates a new branch with <branch_name>
    - git branch
        - Shows the branches within the git repo
        - Branch name with * is the current branch
    - git branch -d <branch_name>
        - Deletes the label for the branch with name <branch_name>

        - Typically used after branches have been merged
- git show <commitID>
    - Shows the differences between files in <commitID> and its parent commit
        - Useful for when you have merged files and don't know if the previous commit in the log is the parent
- git merge <branch_name1> <branch_name2>
    - Merges <branch_name1> and <branch_name2> into one branch that can access commits from either branch
        - In the git log, the commits will show up in chronological order
    - Will also automatically merge <branch_name1> into the current branch
    - git merge --abort
        - Restores files to their state before merging
        - Use when there are merge conflicts
- git clone <repo_URL>
    - Creates a clone of the repository with <repo_URL> that has all of the metadata for the repo not just the files from the most recent commit
    - Can't clone a repository from local -> GitHub
        - See git push
    - Can't clone a repository from GitHub -> GitHub
        - See Forking
- git remote
    - View all current remotes for the local repository
    - git remote add <remote_name> <repo_URL>
        - Creates a new remote with <remote_name> to repository with <repo_URL>
        - Default name for <remote_name> is origin
            - Origin used for forked repo
            - Upstream used for original repo
        - Use HTML URL for <repo_URL> NOT SSH
    - git remote -v
        - Same as git remote but more information is outputted
- git push <remote_name> <branch_name>
    - Pushes the branch with <branch_name> from the local repository to the remote with <remote_name>
    - Maintains the <branch_name> in the remote repository
- git pull <remote_name> <branch_name>
    - Pulls the branch with <branch_name> from the remote with <remote_name> to currently checked out branch on the local repository

    - git pull <remote_name> <branch_name> = git fetch <remote_name> + git merge <branch_name> <remote_name>/<branch_name>
- git fetch <remote_name>
    - Updates <remote_name>/<branch_name> with the contents of the GitHub <branch_name>
- git init
    - Initializes a new Git repository in your current directory
- git reset
    - Removes the files added to the staging directory

Useful Bits

- Head
    - Current commit
- Resolving merge conflicts
    - Open the file where the merge conflict exists
    - Determine what the conflict is
        - Resolve the conflict by fixing code
    - Add and commit the changed file
- Merge conflict outline (ex. merge <master> into <easy-mode> branch)
    - <<<<< Head
        - Code from <easy-mode> branch
    - ||||||| merged common ancestors
        - Original version of the modified code
    - =====
        - Code from <master> branch
    - >>>>> <master>
- Remote (remote repository)
    - Stores the location of a repository that you want to send and receive new commits from
    - ex. A GitHub repository
- Forking
    - Make a copy of someone's GitHub repository directly on the GitHub servers
    - Don't have to clone onto local repo then push to own new GitHub repo
- Pull Request
    - One contributor indicating to the rest that they have a commit that they want reviewed before merging into a certain branch on GitHub (usually master)
    - Essentially a merge request

VIM

- Default text editor for Git Bash
- Normal mode
    - What VIM starts out in
    - Can’t immediately edit text
- i (type)
    - Insert mode
    - Can edit text
    - ESC (press)
        - Returns to Normal mode
- :w + RETURN
    - Saves edits made
- :q + RETURN
    - Quits VIM to return to terminal
- :x + RETURN
    - Combines :w and :q to save and quit VIM