Git, Github, and Github Pages

Setting up an identity

1. check if a command is set up
    - `git config user.name`
    - `git config user.email`
2. set global values
    - `git config --global user.name "First last"`
    - `git config --global user.email janedoe@example.com`
3. List all global configuration settings
    - `git config --list`

Initializing a new directory

Different from cloning a repo. This will create a local repo.

steps

1. Navigate to project directory
2. Confirm that you are not in a git repository
    - `git status`
3. Initialize an empty repo. Make sure you are in the root folder of the project
    - `git init`
4. stage project files. Do this before you commit
    - `git add --all`
5. commit changes
    - `git commit -m "initial commit"`
6. check the status of the branch
    - `git status`

Create and sync a remote repo

Remote repos are associate with a local repo. The default remote repository URL.

Prerequisites

- git installed and configured
- git hosting account created

- existing local repo exists (with no linked remotes)

steps

1. Make sure you're in the root directory of the repo that you want to push
2. Ensure working tree is clean with `git status`
3. Check if any remotes are already attached to the local repo. This command should return nothing.
    - `git remote -v`
4. Create an empty remote repo on GitHub
5. Name repo
6. Add a description (optional)
7. Choose public (if you want to publish to pages)
8. **DO NOT** check any boxes under "initialize this repo"
9. Submit
10. Connect local to remote repo
    - `git remote add origin <remote-repo-url>`
    - `git branch -M main`
    - `git push -u origin main`
11. Refresh page

Publish to Github Pages

Prerequisites

- A remote repo with an index.html in the root directory
- To publish, your your repo must be named <username>.github.io

Enable Github Pages

1. Go settings tab of the repo
2. go to github pages
3. select source as main
4. save changes
5. after a few minutes website should be live at <username>.github.io/reponame

Git Command Notes

- Cannot clone a repo into a repo. This is why trying to clone a github account doesn't work.
- If a repo is stored too far up the directory tree, git will try to keep track of everything in that tree.

git add

Add file contents to the index

-A --all .

all. Update index not only where the working tree has a file matching but where the index already has an entry. adds, modifies, and removes index entries to match the working tree.

-f

Allow added otherwise ignored files

--ignore errors

If some files couldn't be added because of errors indexing them, do not abort the operation **and** continue to add others.

-n

Don't actually add the files, just show if they exist and/or will be ignored

— refresh

Don't add the files, but only refresh their stat() information in the index.

-u

update index just where it has an entry matching

-v

show output

git checkout

Switch branches or restore working file tree.

- checkout will update HEAD to set the specified branch as the current branch `git checkout [branch]`

  -b

  Creates a new branch as if git-branch was called and then checksout to it

  -B

  Like -b but if the branch already exists it resets it to the start point. It is like running git branch -f.

  -m

  when switching branches, if local modifications to one or more files are different between current branch and the branch to which you are switching, the command refuses to switch branches in order to preserve your modifications.

  - helpful for stopping merge conflicts while hopping between Branches.

-t

when creating a new branc, set up "upstream" configuration.

git commit

Record changes to the repository. - TODO learn advanced features later.

-a

automatically stages files that have been modified and deleted

-m

use the message that follows the flag within "" as the commit message

git diff

Shows changes between commits, commit and working tree. TODO find some tutorials on how to do this in specific contexts.

`git diff [option] [--] [path]`

view changes that you have made relative to the index(staging area for next commit)

`git diff [options] --no-index [--] <path> <path>`

compare two given paths on a filesystem. --no-index can be ommitted when the command is being ran in a working tree controlled by git and at least one path pointing outside the working tree.

--color-words

highlight changes by tokenizing added and removed linkes by whitespace

git log

List commits that are reachable by following the parent. Links from the given commits but excludes commits that are reachable with ^ infront of them.

example

```
git log foo bar ^bar
```

- This will list all commits that are reachable from foo and bar but not ^bar.

git ls-tree <branchName>

list files in a branch

git merge

Join two or more development histories together. Incorporates changes from named commits since

the time their histories diverged from the current branch into the current branch.

--abort
: will abort the merge process and try to reconstruct the pre-merge state.

git push

update remote refs with local refs.

--all
: push all branches

--delete
: all listed refs are deleted from the remote repository

--dry-run
: Do everything except actually send the updates

-v
: run verbosely

git rm

Remove a file from the working tree. It will also remove it from the system that it is on.

-f
: override the up-to-date check

-r
: Allow recursive removal when a leading directory is given.

git status

shows the working tree status

-b
: show the branch and tracking info

-b
: give output in short-format

-u
: show untracked files. Options are **no normal all**

-v
: shows textural changes committed (like git diff --cached) as well as names of files