# W2D1

## Terminal Tips and Tricks and Intro to Git

---

## Lecture Objectives

\* Become more comfortable with your Terminal Environment

\* Learn what Git is and how to track/submit projects with Git

---

## Lecture Outline

\* Helpful commands

\* Aliases & Functions

\* Customization

- `bash`

- `zsh`

\* Git/Github

\* Demos along the way

---

## Helpful Commands

\* `cd`: change directory

- `cd <path-to-directory>`

\* `ls`: list

- can add `-a` flag to list all files (including hidden ones)

\* `touch`: create new file

\* `mkdir`: create new directory

---

## Helpful Commands cont.

\* `cp`: copy file or folder to new location

- `cp <path-to-file> <path-to-copy-location>`

- add `-rf` flag to recursively copy folder contents

\* `mv`: move file or folder to new location

- `mv <path-to-file> <path-to-move-location>`

\* `rm`: remove file or folder

- `rm <path-to-file>`

- add `-rf` flag to recursively remove folder contents

---

## Demo of Helpful Commands

---

## `.bashrc` & `.zshrc`

\* Config for your terminal environment

\* Lives in your root directory

- `~/`

\* Open your .bashrc or .zshrc in VSCode with the following command

- `code ~/.bashrc` or `code ~/.zshrc`

---

## Aliases

\* Shorten common commands so you don't have to type as much

\* Add to `.bashrc` or `.zshrc`

```sh

alias name\_of\_alias='command that you want alias to represent'

```

---

## Helpful Aliases

```sh

alias bi='bundle install'

alias be='bundle exec'

alias ber='bundle exec rspec'

# For WSL users

alias wndw='cd /mnt/c/Users/your-username/'

```

---

## Functions

\* Perform some command that requires a dynamic input

\* Add to `.bashrc` or `zshrc`

```sh

function function\_name() {

# commands you want to execute

# Parameters represented by $1, $2, ...$n

}

```

---

## Example Function

\* This function would take in one argument (a file or directory) and move that file or directory to the root directory

```sh

function mtr() {

mv $1 ~/

}

```

\* `mtr <path-of-file>`

---

## Customize your Prompt

\* Prompt can be customized by changing your `PS1` variable

- lives in your `.bashrc` or `.zshrc`

\* [Special Prompt Characters: Bash]([https://ss64.com/bash/syntax-prompt.html](https://ss64.com/bash/syntax-prompt.html" \t "_blank))

- `\u`: username

- `\w`: Current working directory

\* [Special Prompt Characters: Zsh]([https://www.makeuseof.com/customize-zsh-prompt-macos-terminal/](https://www.makeuseof.com/customize-zsh-prompt-macos-terminal/" \t "_blank))

- `%n`: username

- `%1`: Current working directory

\* Can add emojis

- Mac: `cmd + ctrl + space`

- Windows: `wndw + ;`

---

## Customizing Prompt Colors

\* Prompt color can be customized

\* For Bash use this format:

- `\[\e[COLORm\]`

- Text that you want to be that color goes after the closing bracket

- [Colors for Bash]([https://ss64.com/bash/syntax-prompt.html](https://ss64.com/bash/syntax-prompt.html" \t "_blank))

\* For Zsh use this format:

- `%F{color}%f`

- Text that you want to be that color goes between the closing brace and `%f`

- [Colors for Zsh]([https://scriptingosx.com/2019/07/moving-to-zsh-06-customizing-the-zsh-prompt/](https://scriptingosx.com/2019/07/moving-to-zsh-06-customizing-the-zsh-prompt/" \t "_blank))

---

## Example Prompt

bash

```sh

export PS1="\u \[\e[32m\]\w \[\e[00m\] $ "

```

zsh

```sh

export PS1='%F{green}%n %f %F{199}%1d %f $ '

```

---

## Customizing LS Colors

\* LS Colors are colors of listed (`ls`) files

\* Can be customized by changing `LS\_COLORS`

\* For Bash:

- Add `LS\_COLORS` to `.bashrc` file

- Use this format for setting colors

- `type-of-file=color-type;text-color;background-color`

- Remember to `export LS\_COLORS`

---

\* For Zsh:

- Add `LSCOLORS` to `.zshrc` file

- Use this formate for setting colors

- bcfxcxdxbxegedabagacad

- each pair represents a foreground and background color of a file type

- color code and file type order can be found [here](<https://www.cyberciti.biz/faq/apple-mac-osx-terminal-color-ls-output-option/>)

- Remember to `export LSCOLORS`

\* [How to Change Colors of LS in Bash]([https://linuxhint.com/ls\_colors\_bash/](https://linuxhint.com/ls_colors_bash/" \t "_blank))

\* [How to Change Colors of LS in Bash/Zsh]([https://www.cyberciti.biz/faq/apple-mac-osx-terminal-color-ls-output-option/](https://www.cyberciti.biz/faq/apple-mac-osx-terminal-color-ls-output-option/" \t "_blank))

---

## Common LS Files

\* `fi`: file

\* `di`: directory

\* `ex`: executable

\* `no`: default

\* `\*.extension`: specific extensions

- `\*.mp3`: all `.mp3` files

---

## Useful Bash and Zsh customization links

\* [How to Customize (and colorize) your Bash Prompt]([https://www.howtogeek.com/307701/how-to-customize-and-colorize-your-bash-prompt/](https://www.howtogeek.com/307701/how-to-customize-and-colorize-your-bash-prompt/" \t "_blank))

\* [How to Customize the zsh Prompt in the macOS Terminal]([https://www.makeuseof.com/customize-zsh-prompt-macos-terminal/](https://www.makeuseof.com/customize-zsh-prompt-macos-terminal/" \t "_blank))

---

# Break (5 mins)

---

# W2D1

## Tracking and Submitting Projects with Git

---

## What is Git?

---

## Git

- Version Control System (VCS)

- Keeps track of changes to files over time

- Lets us go back and forth to previous versions of projects

- A 'Git repository' (or 'Git repo') contains files and records of changes

---

## Git vs GitHub

Git:

- Tracks changes to files on your computer

Git\*Hub\*:

- Online cloud storage for Git repositories

- Company that is fully separate from Git itself

---

## Basic Commands

- `git init`

- Creates new Git repository at the current directory

- \*Never\* nest Git repos!

- \*Never\* create a repo at the root directory!

- `git status`

- See changes from last "commit"

- \*Always\* run before `git init` to double-check you're not currently in a Git repo!

---

## Basic Commands Cont. - Creating New Commits

- `git add .`

- Confirm you're in top level of your git repo before running

- Adds all files in the current directory and its subfolders

- Makes files \*ready to commit\* but \*does not commit them\*

- `git commit -m "<message>"`

- Message should communicate what has been changed since last commit

- Creates newly saved version of the project

- Can only commit files that have been `git add`ed

- You will usually use both of these in sequence

- Commit frequently!

---

## Basic Commands Cont. - Pushing to GitHub

- `git push`

- 'Pushes' changes from your local computer to GitHub

- Can do as frequently as you'd like, but only NEED to do at the end of the day

- Push your work for the day before submitting the link to your GitHub Repo

---

## Classwork Repo Setup Part 1 - Local

1. Create new local folder to contain your classwork projects (aa\_classwork)

2. `cd` into that folder and run `git status` to ensure there is not a containing git repo

3. Add a folder for your day's projects (ex: W2D1) and `cd` into that folder

4. Run `git init` if there is no git repo

- Otherwise move the folder out of the containing repo and try step 2 again

5. Add at least 1 skeleton or project file to the directory

- We need files to commit!

6. `git add .` and `git commit -m "First commit"`

- Need to have at least one commit before we add GitHub

7. Rename master branch to `main`

- `git branch -M main`

---

## Classwork Repo Setup Part 2 - GitHub

1. Go to [GitHub.com]([https://github.com/](https://github.com/" \t "_blank))

2. Hit the "+" to make a new repository

3. Name repository `W2D1`

4. Follow GitHub instructions to "push an existing repository"

5. Refresh the GitHub page to confirm your files were pushed

6. Add link for the day's folder to progress tracker at the end of the day

---

## Final reminders:

- \*NEVER\* nest repos!

- Always run `git status` before `git init` to be sure

- `git add` AND `git commit` are needed to save a new commit

- You need at least one file in a repository before you can make the first commit

---

## Thank you!

2

[12:25](https://app-academy.slack.com/archives/C03EGEH0YSG/p1655753117725749)

Here is a resource for the personal access token if Github requires you to set it up: <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>

[12:29](https://app-academy.slack.com/archives/C03EGEH0YSG/p1655753341139439)

To access your .bashrc or .zshrc use `code ~/.bashrc` or `code ~/.zshrc`  
Here are some helpful aliases:

alias bi='bundle install'

alias be='bundle exec'

alias ber='bundle exec rspec'

alias gaa='git add -A .'

alias gb='git branch'

alias gc='git clone'

alias gco='git checkout'

alias gcm='git commit -m'

alias gd='git diff'

alias gdc='git diff --cached'

alias goverwrite='git fetch origin master; git reset --hard FETCH\_HEAD; git clean -df'

alias gpo='git push -u origin master'

alias gpr='git pull --rebase'

alias grh='git reset --hard'

alias gst='git status'

alias rdbc='bundle exec rails db:create'

alias rdbd='bundle exec rails db:drop'

alias rdbm='bundle exec rails db:migrate'

alias rdbs='bundle exec rails db:seed'

alias rr='bundle exec rails routes'

alias rcon='bundle exec rails console'