Introduction to Git

Mark Gross

Slides available →

https://raw.githubusercontent.com/MarkusG/UCI-Slides/master/git.pdf

About Me

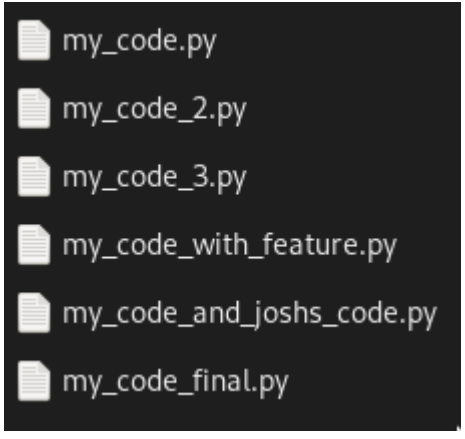
- Mathematics student at Saddleback College
- Software Developer at Roland DGA
- Programming since 2016
- Using Linux since 2018
- In the top 10% of ranked Tetris players worldwide

What is Git?

- **Version control system**
- ***Distributed* version control system**
- **Created by Linus Torvalds in 2005 to develop the Linux kernel**

Why use it?

To avoid this:




```
// this is commented out but we don't want to delete it in case we need it
// later!

//static ht_hash_table* ht_new_sized(const int base_size)
//{
//  ht_hash_table* ht = malloc(sizeof(ht_hash_table));
//  ht->base_size = base_size;
//
//  ht->size = next_prime(ht->base_size); // set size to next prime > base_size
//  ht->count = 0;
//  ht->items = calloc((size_t)ht->size, sizeof(ht_item*)); // zero items
//  return ht;
//}
```

Commits

- **Git tracks the state of your repository via commits**
- **A commit is a snapshot of your code at a given time**

- **Committing a change to git takes 3 steps:**
  - Change the file
  - “Stage” the file in git – `git add`
  - Commit the change – `git commit`

Command Reference

- `git init` – create a new git repository in the current directory
- `git clone <url>` - clone an existing repository from `<url>` into a new directory
- `git add <file>` – add `<file>` to the index (staging area)
- `git rm --cached <file>` - remove `<file>` from the index
- `git commit` – create a commit with all the changes in the staging area
- `git log` – view history

What can we do with history?

- **Reset the project to a previous state**
- **Reset a single file to a previous state**
- **Explore a snapshot of the project at any commit**

Command Reference

- `git checkout` – resets a file to a given state (can also check out commits and branches, more on that later)
- `git reset` – resets the project to a previous state

Branching

- **Branching allows us to keep different tasks separate from each other**
- **Critical to collaboration**

- https://git-school.github.io/visualizing-git/

Merging

- **Merging is how we incorporate changes from a branch into another branch**
- **Merging “replays” the changes made on the divergent branch onto the main branch**
- **This sometimes results in conflicts**

Rebasing

- **Rebasing is the other way we can apply the changes from one branch onto another**
- **Where merging creates a new commit who’s parents are the two branches, rebasing “moves” the commits from one branch onto the other**
- **Interactive rebasing**

Diffs

- **Diffs in general show the *diff*erences between two files**
- **We can use `git diff` to show the differences between two commits**

Remote Repositories

- Git branches can be set up to “track” branches on other repositories
- GitHub
- The “distributed” part of “distributed version control”
- `git fetch`, `git pull` and `git push`