# GIT intro

January 10, 2018

## 1 Why is GIT so Badass?

1. Share - let teammates see your code easily
2. Backup - don't worry about losing hours of work if you push your code regularly
3. Restore points - Take snapshots of your work often, so you can restore any point quickly
4. Diffs - select two points in history, and see what changes were made

#### 1.0.1 Also:

- Cherry pick parts of changes
- Managing several different features (code changes) at the same time
- Rebase your somewhat stale code on top of new code from your peers
- This is a VERY popular tool for source code management in the industry. Maybe the most pervasive tool of all?
- Inspect all changes you've made, and quickly discard changes you no longer need with the RESET command
- (this list goes on forever)

## 2 What is GIT?

1. The name "git" was given by Linus Torvalds when he wrote the very first version. He described the tool as "the stupid content tracker"
2. GIT is a content versioning system, that tracks deltas (changes) in files
3. When you setup GIT, you are enabling tracking for changes in a specific folder on your computer. This is your LOCAL repository
4. You manage sharing code by PUSHing code to a REMOTE repo, usually housed on a stable production server/file share

## 3 How do I start using it?

1. Download a GIT tool. I recommend Tower. GitExtensions (pc) and SourceTree (mac) are other options. GitHub Desktop is crap (unless you want to never get good at git).

2. Hammer into your mind that we will be coordinating 3 repositories for this class:

    1. Udacity Remote Repo
    2. Cohort's Remote Repo

3. Your personal Local Repo

3. Clone the Udacity Deep Learning Repo to your computer. Cloning is grabbing the latest code from Udacity, and setting up your personal LOCAL repo

    - https://github.com/udacity/deep-learning
    - Name the Udacity repo: 'udacity' in remote repository settings

4. Add our cohort's repo as another remote respository for the Deep Learning Repo

    - In repository settings, add the following remote repo: https://github.com/heyhaydude/udacity-deep-learning
    - Name the team repo: 'cohort'

5. From the commit where you see remote "cohort/master" branch, create a new branch with your name (e.g. 'matt')

    - This creates a local branch for you to make your changes on
    - When you do work, make sure you are CHECKED OUT on your branch. In general you won't have to worry about this because you won't be checking out other branches.

6. Commit often

    - Commit changes whenever something meaningful in your coding effort has happened. This allows you to recover when you make mistakes more easily
    - Commits only happen locally, and do not affect the REMOTEs

7. Pushing to the COHORT REMOTE repo

    - Always PUSH just your one local branch to the COHORT REMOTE repo.
    - You should NEVER be pushing to the UDACITY REMOTE repo (unless you want all Udacity students to see your code)
        - If you're nervous about this, you can disable the UDACITY REMOTE

8. Fetching others' changes

    - There is a command to FETCH code from all of your REMOTES (UDACITY and COHORT). This is how you can retrieve new code from your peers