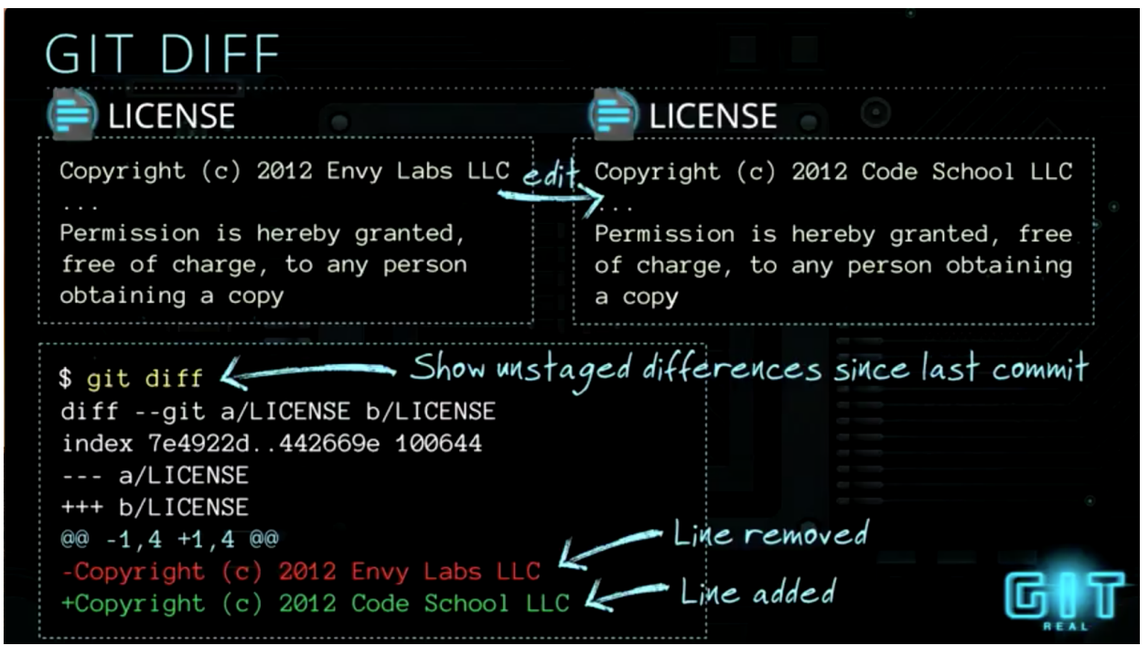
# I. Git Introduction

*Codeschool Notes*

# II. Commits and Resets

## A. Git DIFF

- 1. lets say you made a change to a file but you don't remember what that change was, **git diff** to the rescue!
- 2. **git diff** by itself shows UNSTAGED changes between NOW and your LAST commit
- 3.




  - a) the minus symbol, and a red color means a line was remove
  - b) a plus symbol, and a green color, means a line was added
- 4. once a file has been added to staging, **git diff** won't print out changes on it
  - a) **git diff** is meant to show you changes on stuff you haven't set up for being committed
  - b) to get show changes in staged files use **git diff —staged**

## B. Unstaging Files

- 1. if you've staged something you decide you don't want to commit, use **git reset HEAD YOUR-FILENAME** to unstage
- 2. HEAD refers to the last commit you made

## C. Destroying any changes you made

- 1. lets say you made a bunch of changes and you want to get rid of them
- 2. **git checkout - - MYFILE**
  - a) this checks out the last version of the file git was keeping track of AND OVERWRITES ANY CHANGES YOU MADE

## D. Skip Staging and Commit directly

- 1. **git commit -a -m "some message here"**

- 2. just assumes anything modified will be staged and it then commits it
- 3. note this does not add NEW FILES that you haven't told git to keep track of (you have to use git add at least once on those files)

## E. UNDOING A COMMIT - OR "oops"

### 1. **git reset - -soft HEAD^**

- a) You made a commit, now HEAD points to the last thing you did, but you want to go back to before you commited
- b) the ^ symbol means "go back one", so to go back to your previous commit, you refer to HEAD^
- c) to go back two commit HEAD^^
- d) **- - soft** resets you into staging

### 2. adding stuff to the last commit without going through the **git reset —soft HEAD^** crap

- a) make sure staging is clear
- b) **git add somefile.txt**
- c) **git commit - -amend -m "new message"**
- d) the **- -amend** directive tells git to take whatever you put in staging and shove it onto your last commit

## F. Quick Recap of useful commands

- 1. **git reset - -soft HEAD^** undo last commit and leave it where it was in staging before you last committed
- 2. **git commit - -amend -m "some message"** take whatever you CURRENTLY have in staging, and shove it onto the last commit
- 3. **git reset - -hard HEAD^** undo last commit and ALL CHANGES
- 4. **git reset - -hard HEAD^^** undo last TWO commits and ALL CHANGES

## G. How to share?

### 1. **git remote**

- a) does not take care of access control or hosting

### 2. hosted solutions

- a) github
- b) bitbucket

### 3. roll your own baby hardcore all the way WOOOO!

- a) gitosis
- b) gitorious
- c) lots of drugs (you'll need them)

### 4. assume we are using github

- a) **git remote add origin** https://github.com/Peppy/my-repo-name.git
- b) **add** means **add a new remote**

- c) **origin** is the name you give that remote
- d) the url is something github will tell you, but you can actually guess it from the username and repostiory name

▼ 5. show a list of all repositories

▼ a) **git remote -v**

- (1)


- (2) the “fetch” and “pull” are two different things as far as git is concerned, you can have a name that you only push to, and a url that you fetch stuff from. In this case it’s the same but it doesn’t have to be

▼ 6. pushing to the repostiory

- a) **git push -u repository-name local-branch-to-push**
- b) if you don’t want to write this every time you push checkout the article on https://help.github.com/articles/set-up-git

▼ 7. getting changes from your remote

- a) **git pull**

## ▼ H. Recap of Remote commands

- 1. **git remote add <name of remote> <url address>**

▼ 2. **git remote rm <name of remote>** removes a remote from your local git installation

- a) this does NOT mean that you destroy the remote repository! it means you tell git you don’t want to remember that repository name. imagine you had a bad breakup and destroyed all your girlfriends contact info from your fone, your ex still exists, she’s just not in your contacts
- b) **git push -u <name of repository> <branch you want to push>**

# ▸ III. Cloning And Branching

# ▸ IV. Fixing Upstream Merges