GIT & GITHUB

Lesson 1

1.04. Automatically compare files under OS:

windows: FC; (file compare)

Mac/linux: diff; (difference)

e.g. win: c:\> **fc**  favotite-app-old.html favotite-app-new.html

linux: diff – u favotite-app-old.html favotite-app-new.html

1.20 Installing GIT

**$ git –version** //which version we are using

1.21 Cloning and Exploring

To clone a repository, run git clone followed by a space and the repository URL.

e. g $ **git clone** <https://github.com/udacity/asteroids.git>

$ **git log** will show a list of the recent commits with information about them, including commit IDs.

$ **git diff** followed by two commit IDs will compare the two versions of the code in those commits. Running git diff followed by two commit IDs will compare the two versions of the code in those commits. usually previous ID first, newer ID second

e. g. $ **git diff**  6ab384196a35df76663ffe1184ebd641404ced26 ed98201aa6499901b22c50d72

bc41c9a7acf308f

Entering commit IDs: If it is easier, you may enter the first four or more characters of the commit ID rather than pasting the entire ID.

$ **git config --global color.ui auto** to get colorued output

1.24. Git Errors and Warnings

**Should not be doing an octopus.**  Octopus is a strategy Git uses to combine many different versions of code together. This message can appear if you try to use this strategy in an inappropriate situation.

**You are in 'detached HEAD' state.**  HEAD is what Git calls the commit you are currently on. You can “detach” the HEAD by switching to a previous commit, which we’ll see in the next video. Despite what it sounds like, it’s actually not a bad thing to detach the HEAD. Git just warns you so that you’ll realize you’re doing it.

1.26. Checking Out Old Versions of Code

$ **git checkout** commit ID restoring an earlier version referenced by ID.

in case we are returning the latest we use the latest ID

2.03 Initializing a Repository

in the working directory:$ **git init**

Each Git repository is tied to a specific directory - the directory where you ran git init. Only files from that directory (and subdirectories inside that directory) will be contained in that repository, and you can have different repositories in different directories. Note: it's often the case that a Git repository in some directory will only contain, or track, some of the files in that directory, rather than all of them.

2.04. Examining the New Repository

$ **git status**: List which files are staged, unstaged, and untracked

2.06 Staging area

$ **git add** filename: adding file to Staging area before commit

$ **git reset** filename: removing file from staging area. The file will be removed from the staging area, but it will still be in your working directory.

eg.$ git add lesson\_2\_reflections.txt

$ git reset lesson\_2\_reflections.txt

2.09. Commitig changes

$ **git commit** ; execute, moving changes in the staging area to the repository

Committing all changes: git add, git status, git commit

2.09. git diff revisited

**$ git diff** : show differences between working directory and Staging area

$ **git diff --staged**: show differences between Staging Area and Repository (most recent commit)

$ **git reset - hard**: delete all changes both in Working Directory and in Staging Area

2.14. Making a Branch

$ **git branch**: showing existing branches

$ **git branch** easy-mode: creating a new branch named easy-mode

**$ git checkout easy-mode**: switching to easy-mode branch

2.17. Branches for Collaborations

**$ git log --graph --oneline** master coins: showing graphical representation (commit history) of branches ‘master’ and ‘coins’

2.19 detached HEAD revisited

$ **get checkout –b** new\_branch\_name: in detached HEAD state create a new branch and and it will contain all commit made in the detached HEAD state. equivalent to the sequence of $ git branch new\_branch\_name; $ git checkout new\_branch\_name

2.23. Merging Coins into Master

$ **git gc**: start garbage collection

2.24. Merging on the Command Line

$ git checkout master: you should checkout the branch you want to merge

$ **git merge** master coins: it merges the two branches into current branch. This case it is master.

if you have branch1 checked out, and you run $ git merge branch2 branch3, the merged version will combine branch1 as well as branch2 and branch3.

Restore your files to their state before you started the merge by running **$ git merge --abort**

$ **git show** commit\_id: compare commit with its parent commit

$ **git branch –d** coins: deleting coins branch

2.28. Resolving Merge Conflicts

$ git reset --hard HEAD

$ git config --global merge.conflictstyle diff3

$ git merge master easy-mode

2. 30. Committing the Resolution Solution

$ **git log -n** some\_number: show the latest some\_number commits

$ git log –n 1: show the latest commit