# Git Commands

git --version

git clone <https://github.com/udacity/asteroids.git>

git log

git log –stat

git log –graph

git log –graph –decorate –oneline –all

git status

git diff #show changes between “working directory” and “stage area”

git diff – staged #show changes between “stage area” and “HEAD”

git diff UID\_OLD UID\_NEW

git config –global

git checkout b067…

git checkout master # to return to master HEAD

# git checkout –b new\_branch\_name

git reset --hard #use case: an file deleted by accident, which shown in the “stage area”

git gc # If a branch is deleted and leaves some commits unreachable from existing branches, those commits will continue to be accessible by commit id, until Git’s garbage collection runs. This will happen automatically from time to time, unless you actively turn it off. You can also run this process manually with git gc.

git merge BRANCH \_A BRANCH \_B

git show COMMMIT\_ID #show the diff with the parent

git branch –d BRANCH\_B # delete the branch after merge, only the branch label is deleted, commits are accessible through BRANCH\_A

git merge –abort #run after unsuccessful merge, to resolve the errors

scp – stands for “secure copy” and lets you securely copy a directory from one computer to another.

# Git Rules

* One Commit Per Logical Change, Staging Area may be used for this purpose.

# Bash Commandos

ls -l | grep "git\*" #lists contents with git in the name

diff -u game\_old.js game\_new.js

scp – stands for “secure copy” and lets you securely copy a directory from one computer to another.

# Git Technology Common Info

Made by **Linus Torvalds** to speed up, the programming of the **Linux kernel**.

Other types of **version controls** are Concurrent Version System CVS, and Subversion (for both central server needed).

**Octopus** is a strategy Git uses to combine many different versions of code together.

Git 2.9.0 using **MinTTY** default terminal of **MSYS2**

**Symbol links** trouble in Windows? Look at git v2.9.0 Release note.

**HEAD** is what Git calls the **commit** you are on. You can “detach” the HEAD by switching to a previous commit by git checkout snapshot\_uid, return to master by using “git checkout master”.

https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflowq

# Git Concept Map

TYPE-OF: manual saving, dropbox, google docs, wikipedia are

PART-OF: commit is the

OPERATES-ON: git diff operates on commits

# Git Terminology

|  |  |
| --- | --- |
| Version-control | A folder |
| Commit | Check point (represents version of the file), a snapshot |
| Branch | A label |
| Merge | Combine |
|  |  |

# Cource Reflections

You do not lose your previos work with the version control.

Manual Commits. Give you more control on what en when is saved. By commenting every logical commit, you create handy overview.

08/07/2016 14:00

Multi-File commits by git are seen as dependencies. One file version control is ok for any small tasks. But big projects requere more comlicated workflow with possibility to breack it up in pieces definde a bunch of the steps, to fix my mistackes by going forward and backward. Kind of get everything in the right way before showing to other people. It has steep learning curv but it's woth it because power is so great, and once you do learn it, once you past the curve you can really do a lot. And it's things that you couldn't do without learned that much.

What do you think are the pros and cons of manually chosing when to create a commit, like you do in Git, vs having versions autamaticaly saved, like Google Docs dose?

08/07/2016 14:00

How can you use the commands git log and git diff to view the history of files?

## How might using version control make you more confident to make changes that could break something?

Lewis Kaneshiro

Udacity Course Developer

Risk taker, no fear to the project, freaking out, try to make it work again, able to rethink the entire code base, rethink parts honestly not approached. In general, exploration, making a program incrementally better, easy, no worries about going back and fix a project that’s broken. Explore deep concepts, architecture.git

## Now How Do You Want to Use Git?

Speed up my programming thoughts.

# Configuration

## Git



# Git Get Gather (Config Information)

git config --get remote.origin.url

git config –l #get full config list

# Tools

### Diagramming Tools

* [**gliffy**](https://www.gliffy.com/)
* [**yUML**](http://yuml.me/diagram/activity/draw)

# Git Remote Password Setup

Setup password ([link](https://help.github.com/articles/caching-your-github-password-in-git/)) commandline helper:

git config --global credential.helper wincred

# Git Remote Push

$ git remote add origin <https://github.com/RelaRef/reflections.git>

$ git remote –v

origin https://github.com/RelaRef/reflections.git (fetch)

origin https://github.com/RelaRef/reflections.git (push)

$ git push origin master

Counting objects: 34, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (33/33), done.

Writing objects: 100% (34/34), 2.29 MiB | 333.00 KiB/s, done.

Total 34 (delta 13), reused 0 (delta 0)

To https://github.com/RelaRef/reflections.git

\* [new branch] master -> master