Introduction

Git is a free and open source **version control system** that tracks and manages changes to collections of information, such as a development project.

Github.com is a website where repositories are hosted online. You'll need to sign up for an account to get started.

Git is preinstalled on Mac and Linux systems. You can check the **version** with 'git --version' in terminal.

Terminology

Directory == Folder

CLI (command line interface) == Terminal/Command Line

Repository == Project/Location of the Project

Remote == a common repo on GH

Forking == producing a personal copy of someone else's project

Pull Request == submiting changes to a remote project maintainer for review before changes are implemented, or merged.

HEAD == current working directory, change with git checkout

Basics

git clone <repo link>
Pull repo from GitHub

git init (<dir>)
Start a repo locally.

Create an empty repo on Github, then:
git remote add origin <empty repo url>
Create a new connection to a remote

git status
Show files that are modified but have not been commited

git add <file/dir> [or . for all]
Stage changes for the next commit by adding new file/dir to track.

git commit -m "<msg>" ("<descript>")

git commit -am "<msg>"
Add and commit at the same time for modified (not new) files

git push origin <branch>
origin: location of git repository
master: the branch to push to

git push **-u** origin master
git push **--set-upstream** origin master
-u: upstream, in the future, origin master can be ommited

Git pull origin master

git show <commit>
Metadata and content changes of the specified commit

SSH-Keys

Connect your local machine to your GitHub account. Follow instructions at:
https://docs.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

ssh-keygen -t rsa -b 4096 -C "email"
Creates a new ssh key. Two files will be generated. The .pub extension file is the public key.

Pbcopy < ~/.ssh/id_rsa.pub
Copy public key to clipboard

Go to Github.com > Settings > SSH and GPG keys > New SSH key >
Paste public key into the 'Key' field

eval "$(ssh-agent -s)"
Start the ssh-agent

vim ~/.ssh/config
if file doesn't not exist:
touch ~/.ssh/config

Copy and paste into file:

```
Host *
  AddKeysToAgent yes
  UseKeychain yes
  IdentityFile ~/.ssh/id_rsa
```

Automatically load keys into the ssh-agent and store passphrases in your keychain

ssh-add -K ~/.ssh/id_rsa
Add SSH private key to the ssh-agent

Branches

git branch
View branches and current branch.
q to quit

git checkout (-b <branch name>)
Switch between branches
-b create new branch

git diff (<branch1> (<branch2>))
Compares two version of the code
q to quit

git merge <branch>
Merging locally to master isn't regular practice, instead, push from the 2ndary branch & submit PR
However, as you are working, use merge to pull others' changes down to your local machine often so your version does not fall behind

git fetch <remote> (<branch>)
Fetches all, or a specific <branch>, from the repo.

git branch -d <branch>
Delete branch

Changes

git reset (<file>) ((--hard) <commit hash>)
Undo a staging (e.g. add)
--hard: completely remove changes

git reset HEAD~1
Undo last stage & commit (the change will still be there)
HEAD: last commit
(Point HEAD to 1 step further)

git revert <commit>
Create new commit and undo all of the changes made in
<commit>, then apply it to the current branch.

git clean -n
Shows which files would be removed from working directory.
git clean -f
Executes the clean.

Log

git log
Show commit hashes
Space bar to scroll down

git reflog
Show a log of changes to the local repository's HEAD.
Add --relative-date flag to show date info or --all to show all refs

git log -- <file>
Only display commits that have the specified file

git log --follow <file>
Lists version history for a file, including renames

git log -<limit>
show number of commits

git log --oneline
Condense each commit to a single line.

git log -p
Display the full diff of each commit.

git log --stat
Include which files were altered and the relative number of
lines that were added or deleted from each of them.

git log --author="<pattern>"
git log --grep="<pattern>"
Search by pattern

git log <since>..<until>
Show commits that occur between <since> and <until>. Args can be a commit ID, branch name, HEAD, or any other kind of revision reference.

git log --graph --decorate
--graph flag draws a text based graph of commits on left side of commit msgs. --decorate adds names of branches or tags of commits shown.

Config

git config --global user.name <name>

git config --global user.email <email>

git config --global user.alias. <alias> <git command>

git config --system core.editor <editor>