Git & Github

Notes open for creative commons use @ developer blog: https://unfoldkyle.com, github: SmilingStallman, *email: kmiskell@protonmail.com*

Learning Resource
Primary: "Pro Git" from official Git site

Intro
-VCS – Version control system. Records changes to file(s) so that specific ver can be recalled later
-Local VCS stores all on local disk. Centralized VCS hosts on central remote server w/ clients only pulling latest snapshot. Both = central point of failure.
-Git = Distributed VCS. Clients mirror full repo (all snapshots), allowing restore remote repo if it dies.

-Git is snapshots, not files that are updated on top of the older files. Compressed database.
-Integrity via checksums created from committed files

-When first set up git, set up --*global user.name & user.email*. Then, check via --list to ensure set.
-Search man for help on command via *git help commandName, or git commandName -h*

Workflow
1) Working Directory – where your local files you work on are stored. Init git here.

2) Staging – the act of setting up the elements in a shot for photo or film. ”index” file that stores info for staging area for next commit. Files added to index will be committed on next commit.

3) Local repo - .git directory, which holds commited snapshots, metadata, project object DB, etc.. When commit occurs, files from staging index pushed to local repo snapshots. Can “checkout” from repo to restore project, push project to remote repo, etc.

Basic Git Commands
1) Initialize local repo - navigate to local project folder, then run *git init*. This creates the .git subdir

2) Add to staging – *git add filename*(s). Note can also use *git add *.filetype* to add all. If file is not added, is said to be “untracked.” Note that any mods made on a file after *add* for that file will NOT be committed on next commit. *add* should be done on file after all changes made and re-added if more changes made before commit.

3) Commit – *git commit -m “Commit message (changes, reason, etc.)”*

2 + 3) Combine commit & stage via *git commit -a* . Make sure files you don’t want commitred are in *.gitignore* file (see below).

-Clone an Existing Repo -*git clone <url or local git folder> <optional-dir-to-clone-to>*

-Check status of staging – *git status*. Lists untracked files, modified files, changes to be committed/ not staged, and branch name/status.

-Ignoring files – add all files wish to ignore in *.gitignore* file created in working dir (ex. *.[oa], *~)
-Glob patterns. start pattern with / to avoid recursion. End pattern with / to specify a dir. # for comment. * zero or more. [abc] any char in brackets. ? Single char. ([0-9]) any character in range.

** nested dirs.

-View differences in working dir vs staged – *git diff*
-View differences in staging area vs last commit – *git diff –staged*

-Remove tracked file from staging (when deleting from WD dir) – *git rm <filename(s)/pattern(s)>*
-Remove tracked file from staging (when keeping in WD dir) – *git rm –cached <filename(s)/pattern(s).* Removes from staging but keeps in .git working tree.

-Renaming files – If move not using get *rm* old file, then *add* again. Also, can use *git mv oldFileName newFileName* command to move add using git.

Adding & Removing File Tracking
-Untracked files – files git is not managing. Add to git tracking via *add* command. Note that after a file is added, after commit, file goes back to being untracked. Need to re-add to git to have changes applied on next commit. Can add all via *add -A*

-ignore files (don't show up in untracked files during *status*) via .gitignore file created in project folder with list of files (one per line) wish to ignore. Can use * to ignore multi (ex. *.jpg)

-To untrack added files *git reset filename* or *git reset* to untrack all

Status & Commits
-*git status* – to inspect contents of working dir and staging area

-Track changes in added file via *git diff filename*

-Commit files via *git commit -m "Commit Message"*
  -Commit message should be in qoutes, < 50 chars, & in present tense (ex. Added x functionality)
-Commit and add all files for commit via *commit -a*
-*git log* – to see list of all previous commits

-When first set up git, set up --global user.name & user.email

Cloning a Repo
-Copies a repo from another source
-*git clone <url> <location>* - can be used to clone a local or remote repo
  - ex. *git clone ./Workspace/Project/myproject/.git*
  - ex. *git clone https://github.com/CoreyMShafer/git_project.git*

Push
-Pushes changes from local commits to remote repository
-First, do all steps to commit to local, then first pull master from remote, to have most recent remote repo via *git pull origin master*, then push to remote via *git push origin master*

-To push branch (see below), run *git branch -u origin branchname*

Fetch
-Fetches changes from remote repo to local repo: *git fetch origin branchname-optional*
-Fetched repos do not immediately overwrite local working space. Merge needs to be done to do this
-Workflow: Fetch to local repo. Check for difference in fetched files compared to working space via

diff HEAD. Merge into working dir.

Pull
-Pulls changes to current branch and integrates into current working dir: *git pull origin master*

Branches
-By default, git only creates the master branch, but other branches can be created via *git branch branchname*. You can then switch branches via *git checkout branchname*. This switches new commits to the new branch.
-List local branches via *git branch -a*

-Allows more organized unit testing, release control, etc.

Merge
-Merges files from branch x into master branch
-ex. create test-branch. Edit test-branch. Test test-branch. All good. Merge into master. Push master onto remote repo.

Git Logs
-View commit history – *git log* – displays commit hashes, authors, dates, commit notes, etc.

-*git log* common options

- -*p* (*--patch*) option to show diffs in each commit and limit num of commits shown via *-p #* option (ex. *git log -p -2)*
- *--stat* – show abbreviated log with # files changed & # of changes per file
- *--pretty=online* – one commit per line, w/ hash num & commit msg
- *--pretty=format <format here>* - ex *–pretty=format %h - %an, %ar : %s"*
  - output = ca82a6d - Scott Chacon, 6 years ago : changed the version number
  - see 'pretty=format options.pdf' for all options
- *--since=<date/period>* - ex. *--since=2.weeks* ex. *since="2018-10-01"*
- *--before=<date/period>*
- *--author=<authorname>*
- *--committer=<commiterName>*
- *--grep=<string>* - only show commits containing grep string
- *-S* – only show commits adding or removing code matching the stringe
- See other common *log* options in 'Common Git log options.pdf'or via *git help log*

-Ex. *git log --pretty="%h - %s" --author='Junio C Hamano' --since="2008-10-01" *
--before="2008-11-01" --no-merges – t/

Add Remote Origin
git remote add origin ssh://login@IP/path/to/repository
git remote add origin http://IP/path/to/repository

-To add github repo as remote:
git remote add origin https://github.com/username/repo.git
git remote add origin git@github.com:username/repo.git

Changing Remote Origin
-Often used to change default from SSH to/from HTTPS
git remote set-url origin https://github.com/username/repo.git

git remote set-url origin git@github.com:username/repo.git

Switching Github Users From Local
-Even if pushing with SSH key registered on *UserX* Github account, will still push based on local git project identity, or local global id if no unique project id set

-To ensure proper user is pushing set up global *user.name & user.email* and possible or unique project values before pushing to Github first time