**Git Cheat Sheet w/ Trung Du**

**Setup** (name & email attach to commits)

Tip: Do this on GitHub Desktop

$ git --version

$ git config --global user.name “Trung Du”

$ git config --global user.email “duprogramllc@gmail.com”

**Start a Project** (create local repo by set current <directory> as git repo)

$ echo “# test-repo” > README.md

$ git init <directory>

or Download remote repo to local repo

$ git clone <url**.git>** (via SSH)

**Make a Change to Files**

Add file to staging (Stage file) to track

$ git add <file>

Merge Conflict, choose which patch to add

$ git add -p <file>

Stage all changed files and commit

$ git add .

Commit all staged files to git (history)

$ git commit -m “commitMsg” -m “descMsg”

Commit all tracked files to git (history)

$ git commit -am “commit msg”

Unstages file, keeping file chg

$ git reset <file>

Revert everything to last commit

$ git reset --hard

**Undoing last commit**

Move/rename a file & stage move

$ git mv <existingPath> <newPath>

Remove file from working directory & staging area, then stage removal

$ git rm <file>

Remove from staging area only

$ git rm --cached <file>

View previous commit (READ only)

$ git checkout <commitID>

Create a new commit, reverting changes from a specified commit

$ git revert <commitID>

Go back to previous commit & delete all commits ahead of it (revert is safer)

Add --hard flag to also delete workspace chg (BE VERY CAREFUL) $ git reset --hard <id>

$ git reset <commitID/file>

Change last commit message only

$ git commit-amend

Change last commit id only

$ git cherry-pick

**View local repository**

List new/modified files not yet committed

$ git status

Show full change history, see commit IDs

$ git log origin/main..main (main Commit)

List commit history w/ IDs

$ git log --oneline

Show history for file/directory including diffs

$ git log -p <file/directory>

Show changes to unstaged files ***before merge***

Team: rarely use git diff & git merge **(git push)**

$ git diff <newBranch/file> (q to exit)

Show changes to staged files

$ git diff --cached

Show all staged & unstaged file chg

$ git diff HEAD

Show changes btw 2 commit ids

$ git diff <commit1ID><commit2ID>

List change dates & authors for file

$ git blame <file>

Show file changes for commit id and/or file

$ git show <commit> : <file>

**WTF**

Need help

$ git command --help

<https://training.github.com/>

**GitHub vs git**

Commit on GitHub, not need git add (auto)

Team: Easier view git diff on GitHub

Team: Easier to do pull request on GitHub

**Connect local machine -> GitHub via SSH**

**(1 time, rare)**

Tip: dl GitHub Desktop & initialize

Generate key locally

$ ~ ssh-keygen – t rsa -b 4096 -C “duprogramllc@gmail.com”

Path: /Users/trung/.ssh/id\_rsa

Upload id\_rsa.pub

Note: Never share id\_rsa

GitHub > settings > SSH & GPG keys > New SSH key

Adding your SSH key to ssh-agent

Lookup on GitHub Help

Modify: vim ~/.ssh/config

**Forking a GitHub repository (dl remote repo)**

(*Not a collaborator*, dl repo default branch to local repo to access, then open pull request from GitHub), if collaborator use git pull

Note: Useless to fork own repo

GitHub click fork copy to remote GitHub

Clone copy from your GitHub to local repo

$ git clone <url.git> (via SSH)

**Create new repo on command line (rare)**

Step 1: Create new repository

mkdir test-repo2

cd test-repo2

echo “# test-repo2” >> README.md

git init

git add README.md

git commit -m “add README.md”

Step 2: Push repo to GitHub

Note: not work no test-repo2 on GitHub

Note: vs git clone has copy on GitHub

~~$ git push -u origin main~~

Fix: Create test-repo2 on GitHub & copy SSH

Create connection btw local & remote repo

$ git remote add origin git@github.com:trungdullc/test-repo2.git

Note: OK, since local & remote connected

Upstream tracker (-u): future can use git push

$ git push ***-u*** origin main (created link)

Team: pull request on GitHub

**Synchronizing (created local repo from git, but want upload to GitHub)**

Note: Must create remote repo on GitHub 1st

& copy GitHub SSH link (url.git)

Link local repo to remote repo (not created)

$ git remote add origin <url**.git**>

View all remote connections of local repo

$ git remote -v

Remove a connection to local repo

$ git remote remove origin

Rename a connection

$ git remote rename <old> <new>

Fetch all branches from remote (no merge)

$ git fetch origin

Fetch a specific branch

$ get fetch origin <branch>

Fetch latest chg from origin (remote repo) and merge vs git clone

$ git pull -u origin main

Upload to a branch, so can pull request

Upstream: if use -u in future can use git push

$ git push ***-u*** origin <main/newBranch>

Fetch latest chg from origin and rebase/move

$ git pull --rebase origin

Push local chg to origin

Note: To use need set Upstream first -u

Team: Not use git merge, must check first

$ git push -u origin <newBranch/main>

**Tip: Easier do pull request on GitHub**

**Rename Local Branch**

$ git branch -m main (master->main)

$ git branch -m <oldBranch> <newBranch>

**Rename Remote Branch (GitHub)**

$ git push origin –delete <oldBranch>

$git push -u origin <renameBranch>

**Publishing Local Branch to Remote repo**

Note: git can’t create new remote branch

$ git push -u origin <localBranch>

**Pull/Clone Remote Branch to Local repo**

$ git branch --track feature origin/feature

**Branches (main, develop, bug/feature)**

List all local branches

$ git branch

List all branches (local & remote)

$ git branch -av

List all branches

$ git branch -a

List remote branches

$ git branch -r

Create a new branch

$ git branch <newBranch>

***Switch branch*** & update working directory

$ git checkout <branch> git switch <br>

Create new branch and switch to it

$ git checkout -b <newBranch>

Delete local merged branch (except active)

$ git branch -d <newBranch>

Delete any localBranch (except active branch)

$ git branch -D <newBranch>

Delete remote Branch from git

$ git push origin --delete <remoteBranch>

Tag current commit (for new version releases)

$ git tag <tagName>



**Merging**

Merge newbranch into main in local repo

--no-ff: for no-fast-forward merge

$ git checkout main or develop

$ git merge <newBranch>

Merge & squash all commits -> 1 new commit

$ git merge --squash a

Undo conflict and start over

$ git merge --abort



**Rebasing** (Note: history changed)

Rebase feature branch onto main (to incorporate new changes made to main in linear fashion)

$ git checkout <feature>

$ git rebase main

Iteratively clean up a branches commits before rebasing onto main

$ git rebase -i main

Iteratively rebase the last 3 commits on current branch

$ git rebase -i Head~3

Undo conflict and start over

$ git rebase –abort

**Merge Conflict (keep up with remote main)**

$ git pull -u origin main

$ git diff main

$ git merge main

$ mate <file> (fix file manually)

Fix: VSC open merge conflict file & edit

**Warning:** Do not use rebase on commit that you’ve already pushed/shared to GitHub remote repository, only for cleaning local commit history before merging

**Basic Concepts**

Hint: Open terminal in VSC

main: default development branch

origin: default upstream repo

HEAD: current branch

HEAD^: parent of HEAD

HEAD~4: great great grandparent of HEAD

**WINDOW vs LINUX commands**

dir ls

dir /a:h ls -la

cls clear

del <file> rm <file>

rmdir /s /q <dir> rmdir <dir>

rd /s /q <dir> rm -rf <dir>

pwd

cd ../..

mkdir <directory>

vim

**Stashing (record current directory temporarily and not commit to git)**

Store modified & staged changes

include untracked files, add -u

include tracked & ignore files, -a

$ git stash

As above, but add a comment

$ git stash save “comment”

Partial stash, stash single file, a collection of files, or individual changes from within files

$ git stash -p

List all stashes

$ git stash list

Re-apply stash w/o deleting it

$ git stash apply

Re-apply stash at index 2, then delete it from stash list.

Omit stash@{n} to pop recent stash

$ git stash pop stash@{2}

Show diff summary of stash1

Pass -p flag to see full diff

$ git stash show stash@{1}

Delete stash at index 1

Omit stash@{n} to delete last stash

$ git stash drop stash@{1}

Delete all stashes

$ git stash clear