­­Notes for Git

2024/12/9 Potumas Liu

(git-bash.exe to open git bash via cmd)

* Version Control System (Record code change and history): trak history, work together
  + Centralized: Connect via internet
  + Distributed: Have local instance
* CMD: Fastest, Easiest (Vscode: gitlens, or git website, GitKraken, Sourcetree, but GUI have limitations, and may not always be available)
* Process Using Git
  + Download Git from: <https://git-scm.com/>
  + Configurations (google git config)
    - git config --global user.name “<username>”
    - git config --global user.email “<useremail>”
    - git config --global core.editor “code --wait"
    - git config --global -e (edit the configuration file by default editor)
    - git config --global core.autocrlf true (Windows: true, Liunx: input) (git EOF config)
    - git config --help (go to a website for help)
    - git config -h (a brief discription of git config command)
* Getting Started
  + (pretty: windows: posh-git)
  + Go to a project directory
  + git init (initialize a file ‘.git’ for storing history. DON’T TOUCH IT!!)
  + ls -a (see everything, including the hiding ones)
  + work on project 🡪 staging area 🡪 commit 🡪 store in repository
  + echo hello > file1.txt
  + echo hello > file2.txt
  + git status (check for untracked files)
  + git add file1.txt file2.txt (git add . adds everything) (update the staging area)
  + echo world >> file1.txt (changes things in file1.txt)
  + git commit -m “<commitMSG>” (store the snapshot: ID, message, time, author, complete snapshot) (can quickly get back to previous stage)
  + git commit (open the editor and allow you to edit commiting messages)
  + Commit when you reach a state, and separate problems that have been resolved. Make commit messages meaningful.
  + It is unnecasary to stage code before commiting. (only when you know WTF u r doing)
    - Git commit -am “<msg>”
  + Rm file2.txt
  + git ls-files (show files in the staging area)
  + git add file2.txt (file2.txt is removed, but still type ‘add’)
  + git rm file2.txt (removes files from working dir and stging area once)
  + mv file1.txt main.js (copy a file or rename a file)
    - if run ‘git status’, we get 2 detects: delete file1.txt, and newfile main.js
    - git add file1.txt, git add main.js (a must to track main.js)
    - git rm main.js file1.js (does 2 things at once)
* We don’t want to commit log files
  + Mkdir logs, echo hello > logs/dev.log (logs/ for the directory)
  + A special file in git is ‘.gitignore’
  + Echo logs/> .gitignore
  + Code .gitignore (to edit the file, can also use patterns, e.g. \*.log)
  + Git add .gitignore
  + Git commit
  + YOU MUST put a file into .gitignore before putting it into the staging area, otherwise git will keep track of it, even if you add the file into .gitignore afterwards.
  + To remove sth. From git staging area, ‘git rm --cached -r <dir>/’
  + Github/gitignore for some templates for different programming languages.
* Simpler git status information
  + Git status -s
    - Red: not in the staging area, Green: in the staging area (a snapshot). You HAVE TO add each time a file is modified.
    - M: modified
    - ??: newly created file
    - A: added
* Reivew your code before commiging
  + Git diff --staged (to see the modifications between staging area and repository)
    - $ git diff --staged
    - diff --git a/file1.js b/file1.js (compare a with b)
    - index b2b9cc9..17f9c9c 100644
    - --- a/file1.js (---: old copy)
    - +++ b/file1.js (+++:new copy)
    - @@ -1,2 +1,4 @@ (-: old copy, +: new copy)
    - hello
    - test
    - +sky (green: newlines)
    - +ocean
    - diff --git a/file2.js b/file2.js
    - new file mode 100644
  + git diff (difference between working dir and staging area)
    - (git status -s, all green if nothing is changed without tracking)
    - (+: new copy, -: old copy, green: added, red: removed)
* Comparing codes by vsCode
  + Git config --global diff.tool vscode (vscode is just a name)
  + Git config --global difftool.vscode.cmd “code --wait --diff $LOCAL $REMOTE”
    - Cmd to wait until editor is closed
    - Diff: compare differences
    - MAY have to manually setup ‘$LOCAL $REMOTE’
  + Git difftool (view the difference by vscode)
* History of commit
  + Git log
  + Git log --oneline (short summary)
  + Git log --oneline --reverse