Git

- 분산 버전 관리 시스템
- 버전을 통해 코드를 관리하는 도구
    - 코드 관리도구
    - 코드 협업도구
    - 코드 배포도구

SCM & VCS

- SCM(Source Cod Management): (소스) 코드 관리 도구
- VCS(Version Control System): 버전 관리 도구
- DVCS(Distributed Version Control System): 분산 버전 관리 도구

Git 명령어

> 프로젝트(폴더) 기반 코드 관리 도구
>
> `git [명령어]`

(1) `git init`

Git으로 코드 관리 시작

- 코드 관리 단위(기준): 폴더
- `(master)` : ~~현재 branch명~~
- `.git` 폴더 생성: Git이 관리와 관련된 파일(mkdir [폴더명])

```
$ cd ~ #언제나 home에서 시작. 옆에 master상태 있으면 안됨!
$ rmdir intro/
$ git initgit
Initialized empty Git repository in C:/Users/user/intro/.git/
```

- git으로 프로젝트 관리를 중단: `.git` 디렉토리 삭제

```
$ rm -r .git
```

(2) `git staus`

Git의 현재 상태 출력(Git에게 현재 상태를 물어봄)

```
$ git status
On branch master  #master라는 branch에 있음

No commits yet    #아직 commit 없음

nothing to commit (create/copy files and use "git add" to track)   #commit 할 것이 없음
```

- `touch a.txt` 파일 추가 후, `git status`

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a.txt
   #추적되지 않은 파일이 있음(commit하려면 add해야함) (파일명)
nothing added to commit but untracked files present (use "git add
" to track)   #commit 하기 위한 파일이 추가되지 않았음. 그러나 추적되지 않은 파일이 존재함
```

- `git add a.txt` 직후, `git status`

```
$ git status
On branch master

No commits yet

Changes to be committed:      #commit 할 변경들이 있음 (파일명)
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt
```

- `git commit` 이후,

```
$ git commit
On branch master
nothing to commit, working tree clean   #commit 할 게 없음. 작업 폴더는 깔끔함.
```

- 파일 수정 후,

```
On branch master
Changes not staged for commit:   #commit하기 위해 stage되지 않은 변경 사항이 있어요

  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:    test.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

(3) `git add [파일명/폴더명]`

commit을 위한 stage

- `git add .`: 현재 폴더의 모든 변경 사항 stage('찰칵')

(4) `git rm --cache [파일명/폴더명]`

파일 제거

(5) `git commit -m "커밋 메시지"` / `git config`

> commit == 버전을 생성 == 현재 상태의 스냅샷 촬영

Git 설정 관련 명령어

- 최초 commit 실행 시

```
$ git commit -m "First commit"   #작자 미상
Author identity unknown

*** Please tell me who you are.   #당신이 누군지 알려주세요.

Run                               #아래의 명령어를 실행하세요.

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'user@SolenovoE14
.(none)')
```

- `git config --global user.email [나의 이메일]` : (global(전역으로) user의) 이메일 설정
- `git config --global user.email`: 설정값 확인
- `git config --global user.name [나의 이름]`: 이름 설정
- `git config` 설정 후(`vim` 에디터 창),

```
# Please enter the commit message for your changes. (#변경사항에 대한 commit
message 입력)
# Lines starting with '#' will be ignored, and an empty message aborts the
commit.
   (#로 시작하는 라인은 무시됩니다. 아무것도 없는 메시지는 종료됩니다.)

# On brach master
#
# Initial commit
#
# Changes to be committed:
#       new file:   test.txt
```

(6) `git log`

현재까지의 commit를 출력

```
$ git commit -m "First commit"
[master (root-commit) 8238f3d] First commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a.txt
```

```
$ git log
commit 8238f3db64a45016f998067e7146bf7aae5adccc (HEAD -> master)
Author: SOL <hphk.datasoling@gmail.com>    #작성자
Date:   Thu Jul 13 15:49:45 2023 +0900     #날짜

    First commit                           #커밋메시지
```

- 파일 하나 더 추가해보자.

```
$ touch b.txt
$ ls
a.txt  b.txt
$ git commit -m "ADD b.txt"
[master 9489a21] ADD b.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 b.txt
$ git log
commit 9489a21d65b199b32ba0deca76c6a988b4735a7d (HEAD -> master)
Author: SOL <hphk.datasoling@gmail.com>
Date:   Thu Jul 13 15:54:53 2023 +0900

    ADD b.txt

commit 8238f3db64a45016f998067e7146bf7aae5adccc
Author: SOL <hphk.datasoling@gmail.com>
Date:   Thu Jul 13 15:49:45 2023 +0900

    First commit
```

- `git log --oneline`: 한줄로 출력

```
$ git log --oneline
9489a21 (HEAD -> master) ADD b.txt
8238f3d First commit
```

(7) `rm -r .git`

master 상태에서 일반 계정으로 상태 변경

백업하기

```
1. 원격 저장소 생성
1. remote 정보 입력
1. git push
```

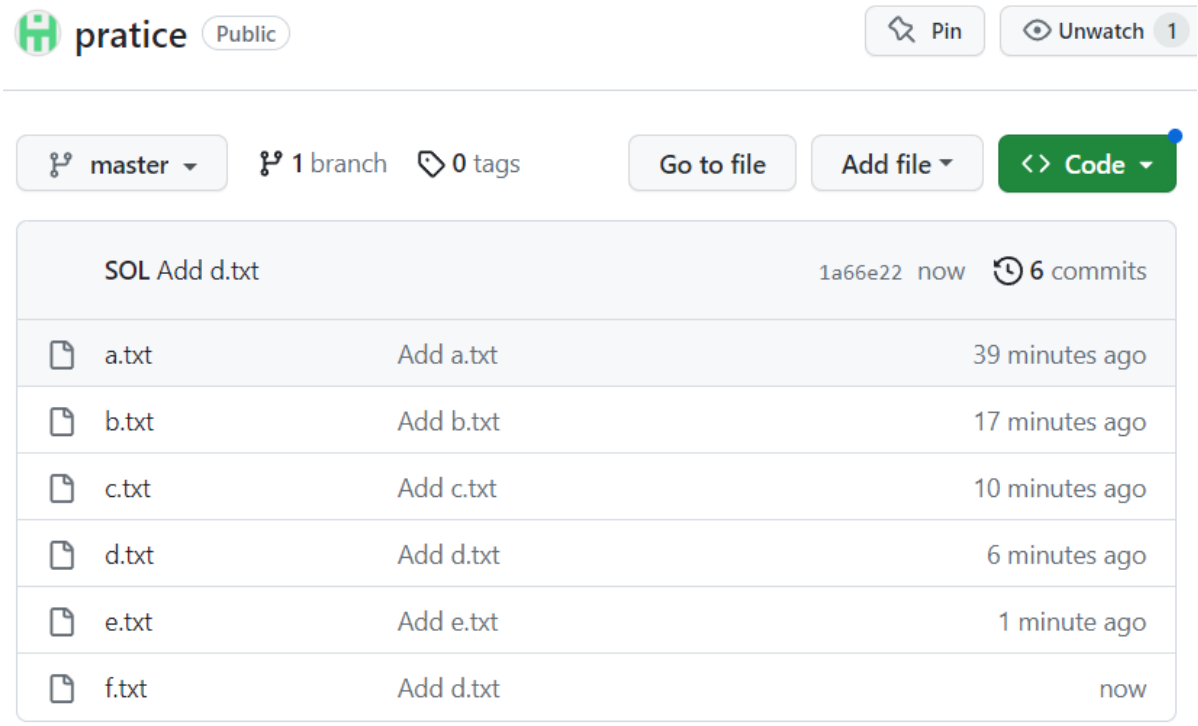
(1) `git remote`

- `git remote add [저장소 이름] [저장소 주소]` : git remote add origin https://github.com/hkeryfonttlxisdrlw/basic_git
  - git에게 원격저장소(remote) 추가(add)를 명령
- 저장소 이름: `origin` (보통 첫 이름은 origin으로 많이 함)
- 저장소 주소: https://github.com/hkeryfonttlxisdrlw/basic_git

```
$ git remote   #아무말없으면 원격저장소 없는 것

$ git remote add origin https://github.com/SSolLEE/intro.git

$ git remote
origin

$ git remote -v   #상세출력(주소 확인)
origin  https://github.com/SSolLEE/intro.git (fetch)
origin  https://github.com/SSolLEE/intro.git (push)

$ git push origin master   #원격 저장소에 코드 백업 ##git push [저장소별명][branch이름]
info: please complete authentication in your browser... #로그인된 github과 연결
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 6 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 419 bytes | 139.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SSolLEE/intro.git
 * [new branch]      master -> master

#위 remote -v 명령어에서 push에 해당하는 주소로 들어가면 코드가 github에 생긴 것을 확인할 수 있음
```

(2) 한 번 한 후, 파일 추가하기

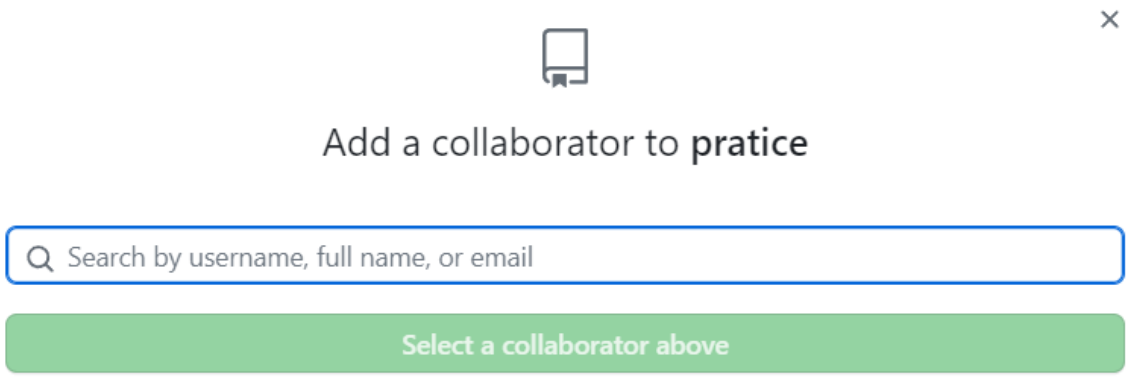
```
$ touch d.txt
#$ git status
$ git add d.txt
#$ git status
$ git commit -m "Add d.txt"
#$ git status
#$ git log               # #표시한 것들은 습관적으로 확인해보자. 중간점검
$ git push origin master
```



협업하기

(1) 협업 멤버 추가하기 -작업 폴더 수정 권한 부여

- 상단 바에서 `Setting` - 왼쪽 카테고리에서 `Collaborators`
- 하단에 `Add People` 클릭



- `Select` 클릭

- `Pending Invite` 링크 복사하여 협업 멤버에게 전달

(2) 추가 된 협업 멤버 작업

- 이메일 또는 수신함 확인하여 `Accept`
  - 협업이 가능해졌다!
- `git clone [팀장주소] [지정폴더명(내맘대로)]`: 원격 저장소 코드 다운로드

```
$ cd ~  #home에서 작업시작
$ git clone https://github.com/SSolLEE/pratice collab

#위의 작업대로 파일 추가 작업 똑같이 수행(touch, add, commit, push)
```

- 추가된 멤버는 /collab에서 작업한다.
  - `git init`을 하지 않아도 ~/collab 뒤에 `(master-branch이름)`이 형성되어 있다.

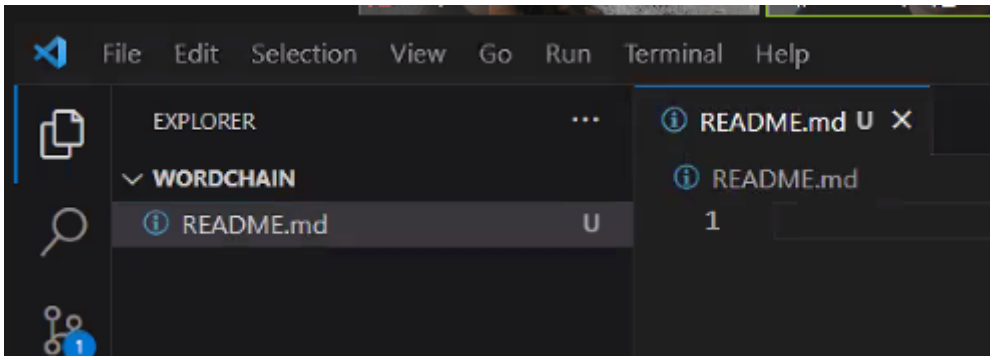
```
#다른 사람이 만든 파일 당겨오기

$ cd practice/    #git작업 폴더에서 하는 것(mater 표시 확인!)
$ ls
a.txt  c.txt  e.txt  g.txt  i.txt  k.txt
b.txt  d.txt  f.txt  h.txt  j.txt
$ git pull origin master #pull : github의 practice폴더에 있는 파일들 당겨오기
$ ls
a.txt  c.txt  e.txt  g.txt  i.txt  k.txt
b.txt  d.txt  f.txt  h.txt  j.txt  l.txt

#그 다음 이어서 파일 만들기 - (touch, add, commit, push)
```

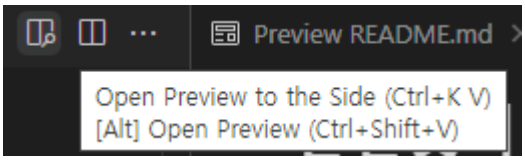
(3) 끝말 잇기(한 파일 내에서 협업하기)

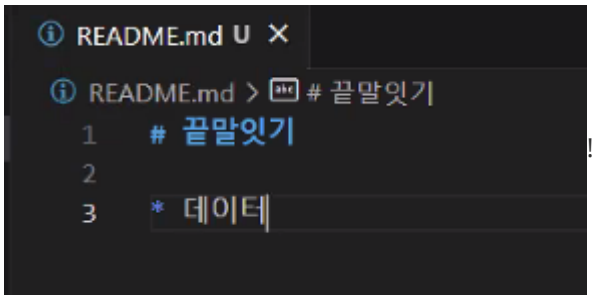
- 팀장

```
$ cd
$ mkdir wordchain
$ cd wordchain

$ git init
$ code . #VS code 열기 명령어: code , 현재 폴더 : .

#이 방법으로 안열리면 VS code 열어서 해당 폴더 열기
```

- VS code
  - `new file` : READ.md

    - 

    - 오른쪽 상단 `네모네모 문서에 돋보기모양` 누르면 preview 확인 가능
    - 

    - 

    ![image-20230714162717475] (C:\Users\user\AppData
    - 저장하기
- 하단 `TERMINAL` (/wordchain (master) 경로에서 시행하는 것 확인)

```
$ git status
$ git add README.md    #두글자 정도 치고 tab하면 자동완성
$ git commit -m "데이터"
$ git log
```

- githnb 올리기
    - repository 'wordchain' 생성
    - 파일 올리기

```
$ git remote add origin [url]
$ git remote -v
$ git push origin master
```

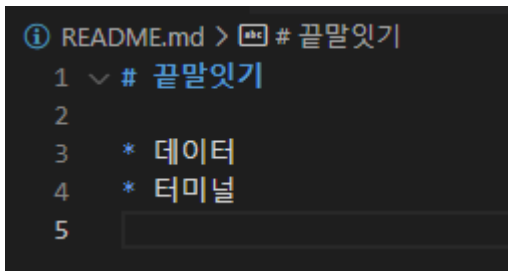
- 팀원 차례

```
$ git clone https://github.com/@@@@@/wordchain wordchain   #복제
$ cd /c/Users/user/wordchain   #폴더로 이동
$ code .   #VS code로 이동. 자동으로 열림
```

- VS code

- 왼쪽에 README.md 있음. 클릭하면 팀장이 쓴 단어가 보임
- 아래에 이어서 단어를 쓴 후 `File-Save`
- `Terminal-new Terminal(bash 형식인 것 확인)`

```
user@&&&&& MINGW64 ~/wordchain (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

user@&&&&& MINGW64 ~/wordchain (master)
$ git add README.md

user@&&&&& MINGW64 ~/wordchain (master)
$ git commit -m "터미널"
[master 33fc0f4] 터미널
 1 file changed, 2 insertions(+), 1 deletion(-)

user@&&&&& ~/wordchain (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mizima1015/wordchain
   ed27837..33fc0f4  master -> master
```



-
- github에 push가 된 것 확인

- 그다음 사람은 terminal에서 바로 git pull 해서 이어서 같은 작업 반복

```
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 271 bytes | 7.00 KiB/s, done.
From https://github.com/mizima1015/wordchain
 * branch            master     -> FETCH_HEAD
   739e0e6..b4c433f  master     -> origin/master
```

```
Updating 739e0e6..b4c433f
Fast-forward
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

$ git add README.md

$ git commit -m "이름표"
[master 88451b1] 이름표
 1 file changed, 2 insertions(+), 1 deletion(-)

$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/@@@@@/wordchain
   b4c433f..88451b1  master -> master
```