ROLO - NOMAD CODERS NAPOLITAN

GIT & GIT HUB

용어정리 및 커밋방법 상세설명

작성일 : 2023.05.17

출처 : progit.pdf, 위키백과, Doeveloper님 명강의(5/16)

VERSION CONTROL SYSTEM 의 종류

LOCAL

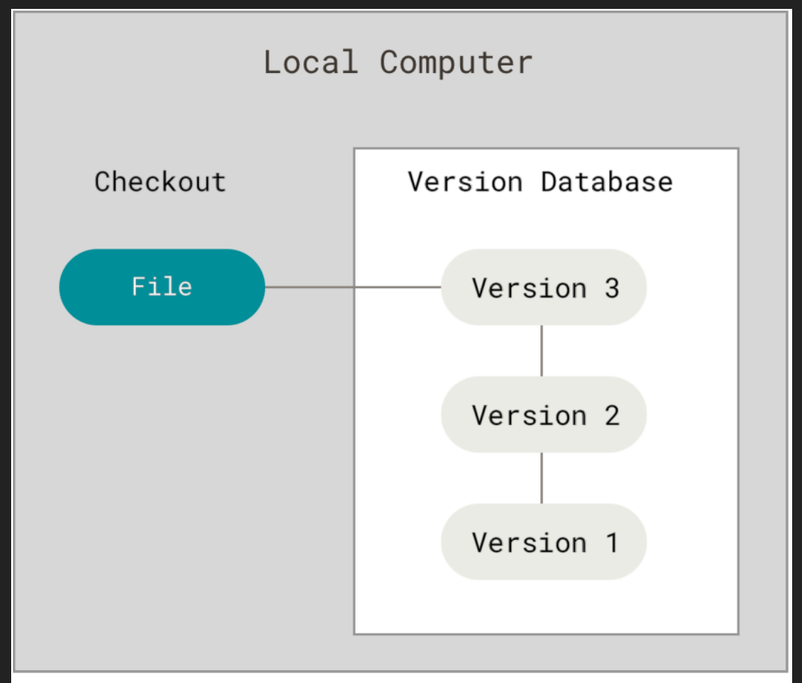


Figure 1. Local version control diagram

CENTRALIZED

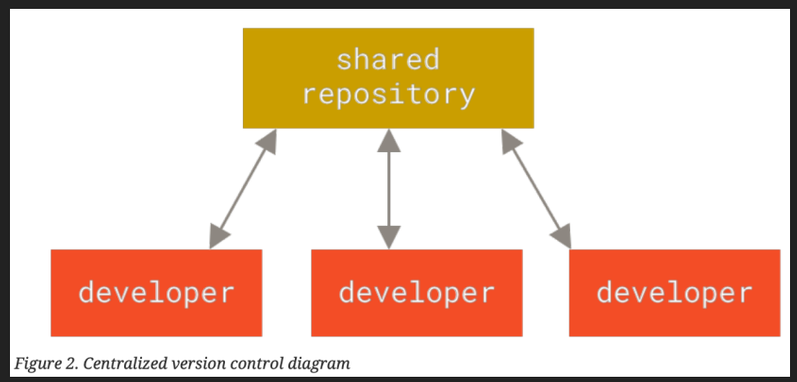


Figure 2. Centralized version control diagram

DISTRIBUTED

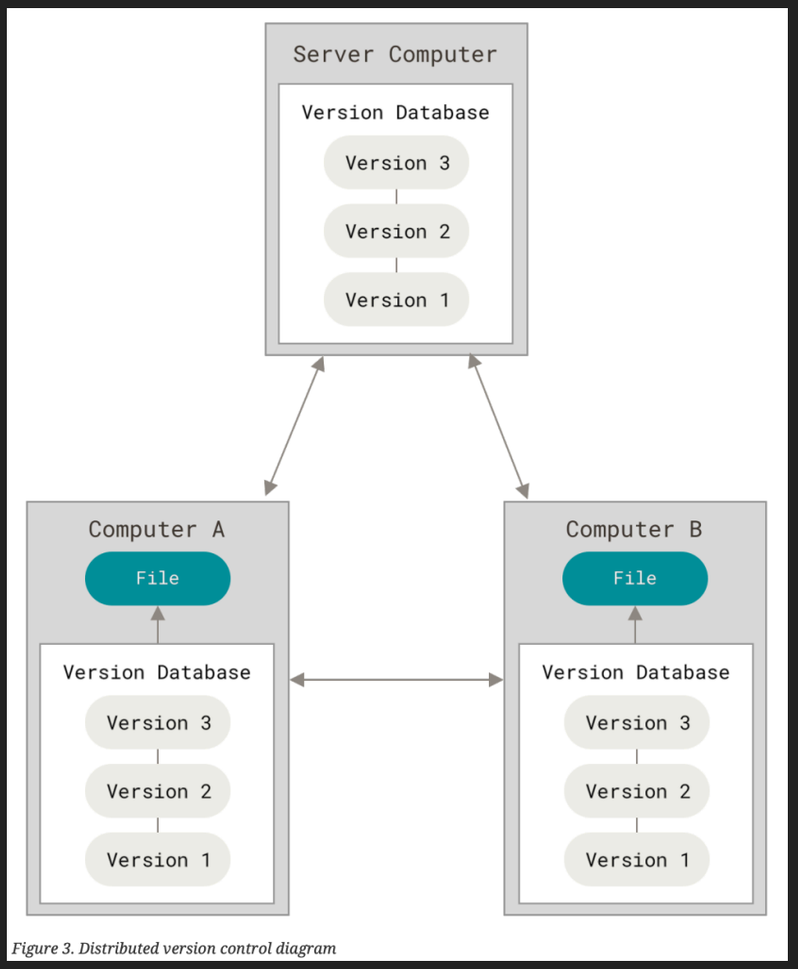


Figure 3. Distributed version control diagram

단점 : 한곳에 저장된 파일이 날아가면, 모든 프로젝트를 망친다

- 마치 파일 버전1, 버전2 저장하듯, 따로 저장을 하되, patch set을 이용
- 대표프로그램 : RCS

- 몇년 간 기준이 되어 이용됨, 1개의 중앙 저장소에 모든 사람이 접속하여 파일을 업데이트함.
- 대표프로그램 : SVN, Perforce

- 최근 가장 핫함. 파일의 마지막 상태만 가져오는 것이 아니라, 모든 히스토리를 가져온다. 클론 레파지토리는 풀백업이다.
- 대표프로그램 : Git, Mercurial, Bazaar or Darcs)

깃의 탄생배경

- 2002년 : 리눅스 커널 프로젝트가 BitKeeper 라는 분산 버전 관리 시스템을 사용함.
- 2005년 : 리눅스 커널 프로젝트가 더이상 BitKeeper를 사용하지 못하게됨. 따라서 리눅스 창시자 리누스 토르발드가 자신만의 버전관리시스템을 개발하게 됨.

- 리누스 토르발드의 새로운 버전관리 시스템의 개발목표
  - 빠른속도
  - 간단한 디자인
  - 수천개의 브랜치를 사용할 수 있는 병렬적인 개발지원
  - 완전히 분산된
  - 리눅스 커널 프로젝트같은 아주 큰 프로젝트를 다룰 수 있도록 (속도와 용량면에서)

깃이 갓인 이유 (깃의 장점)

1. 파일의 저장시점만 저장하는 것이 아니라, 변경사항만 저장한다. 변경된 것이 없으면 덮어쓰기같은 작업을 하지 않는다. 그냥 저장을 안함.(변경사항없음)
2. 저장 할때마다 현재 프로젝트의 파일 전체를 스냅샷을 찍는다고 생각하면된다.
3. 이전버전과 이후버전 비교가 용이하다.
4. 이전버전데이터를 당겨올 (pull) 수 있다.
5. 로컬 컴퓨터에서 작업한다.
    1. 인터넷없는 곳 (비행기, 시골)에서도 커밋을 할 수 있다.
    2. VPN이 없을때도 커밋을 할 수 있다.
    3. 히스토리 확인 속도가 매우 빠르다. (로컬에 모든 프로젝트의 히스토리가 저장되어 있기 때문에)
6. 정보를 잃기가 어렵다.
   모든 정보를 추가로 저장하고 추가 데이터를 집어 넣는 개념이기때문에

깃의 저장방법 및 온전함 (INTEGRITY)

- 체크섬을 이용해 저장
  (체크섬: 나열된 데이터를 더하여 체크섬 숫자를 얻고, 정해진 비트수의 모듈라로 정해진 비트수로 재구성 한다)
- 이로 인해, 깃이 모르게 파일을 저장하고 변경하고 삭제할 수가 없다. 깃이 체크섬 방법으로 저장을 했기때문에, 변경시에 “해독”을 하는 작업이 필요하므로, 깃이 그 변경사항을 지나칠 수가 없다.
- 깃의 메커니즘 : SHA-1 hash
  ->SHA-1 hash의 포맷 (16진수를 이용한 40자리 String)
  24b9da6552252987aa493b52f8696cd6d3b00373

3단계 영역 (AREAS, STATES)

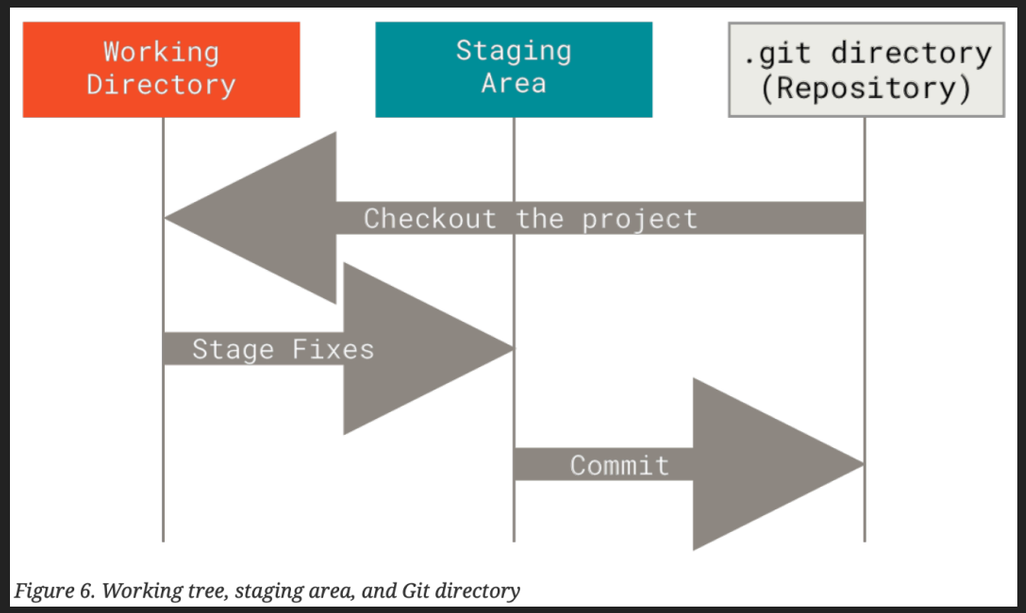
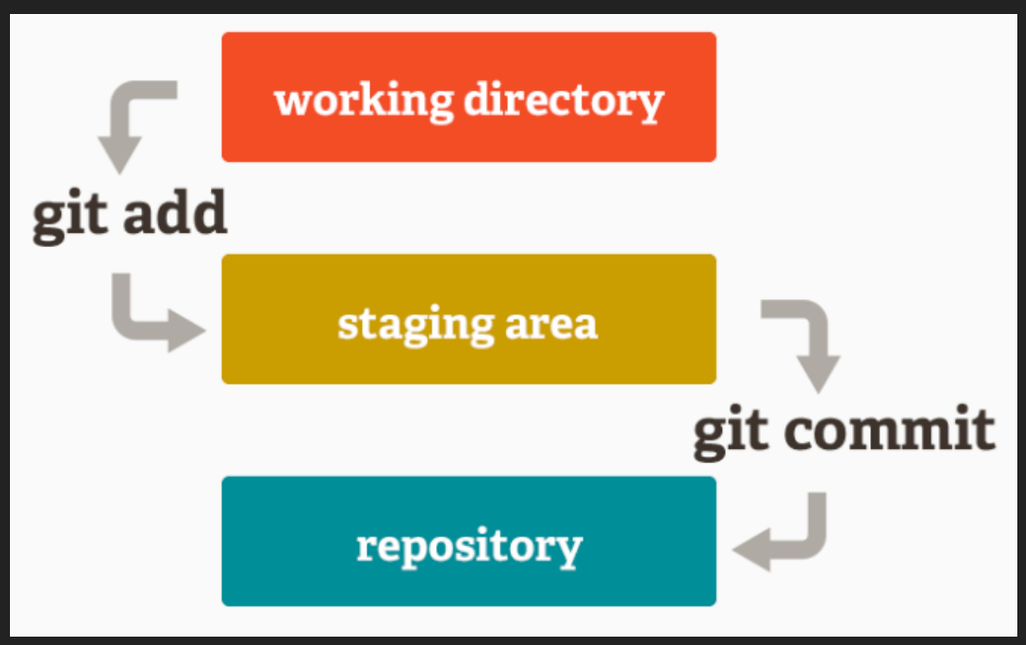


Figure 6. Working tree, staging area, and Git directory



- Working Directory (modified)
  - 파일을 변경했으나, 아직 저장되지 않은 상태
- Staging Area(staged)
  - 파일을 스테이지에 저장한 상태
- .git directory(Repository) (committed)
  - 스테이지의 정보들을 저장소에 올린 상태

GIT & GIT HUB DESKTOP & GIT HUB

SCM tools

GIT https://git-scm.com/

시스템

- 무료 오픈소스 분산 버전 관리 시스템
  - 저장시점을 기준으로 브랜치 생성 및 병합이 용이
  - 분산 <-> 중앙집중,
    중앙집중(centralized) : 다른 사람이 해당 브랜치를 작업하고 있을때, 다른 사람은 락이 걸려서 사용 할 수 없다. (Subversion)
    분산: 언제든지! 작업 할 수 있고, 인터넷이 없어도 작업할 수 있다. (Git)
- 빠른 속도 & 적은 메모리 차지
  - C 언어로 개발됨 (그래서 빠름)
- 2005년 개발 (리누스 토르발스)

GIT HUB DESKTOP
https://desktop.github.com/

GIT GUI

- Git 의 GUI.
- 커맨드로 git을 다루기가 복잡하다보니, 개발된 툴.
- 다른 폴더에 파일 저장하듯이, 편리한 화면에서 작업 할 수 있다.

GIT HUB https://github.com/

웹 서비스

- 루비 온 레일스로 작성된 분산 버전 관리 툴인 깃 저장소 호스팅을 지원하는 웹 서비스
- 오픈소스 소프트웨어 인터넷 호스팅 서비스
- 유상/무상
- 2008년 설립, 2010년 깃허브로 명명
- 2018년 마이크로소프트가 인수
- 본사 : 캘리포니아 샌프란시스코
- Remote repository는 git hub 의 것!!

GIT과 GIT HUB을 적절히 사용하는 순서 (파일을 저장하는순서)

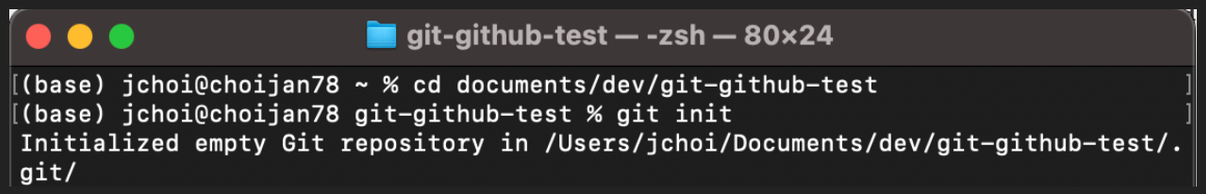
- Git repository 만들기
  - 내 컴퓨터의 폴더를 git repository 로 지정하기
  - Remote repository를 불러오기(cloning)
    - 다른 사람의 remote repository는 권한이 없기 때문에, git hub에서 fork를 하고 forked 된 본인의 remote repository를 불러와야 한다.
- VSC 를 이용해 코드파일 생성 및 저장
- 나의 Remote repository 에 올리기
- 다른 사람의 Remote repository에 pull request 하기. (권한이 없는 3자의 repository)

GIT과 GIT HUB을 조화롭게 사용해야 하는 이유

파일 저장하는 시스템 자체가 GIT 이고, REMOTE REPOSITORY는 GIT HUB이기때문에
클라우드 저장소를 이용하려면 GIT HUB을 이용해야하고, 파일 저장은 GIT으로 해야되기 때문

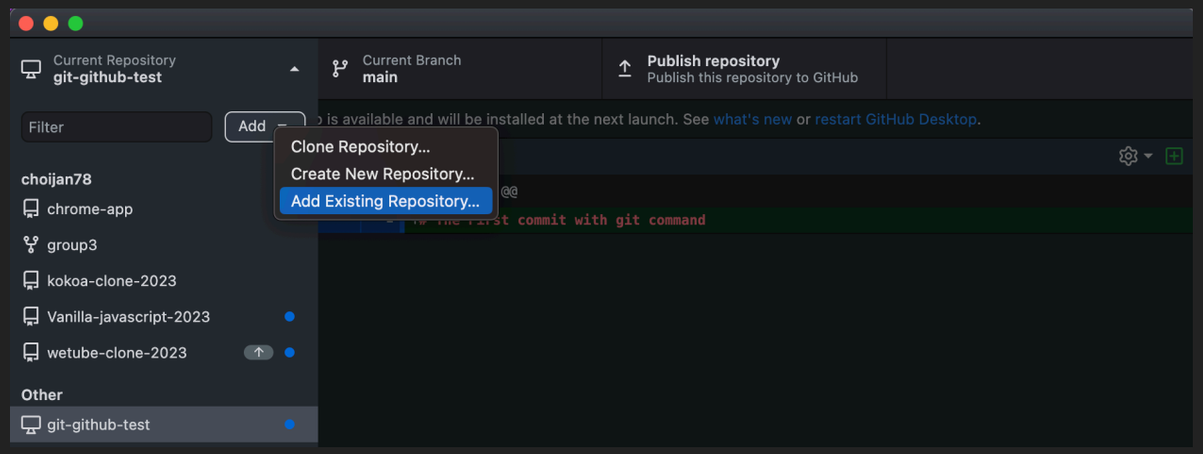
GIT COMMAND

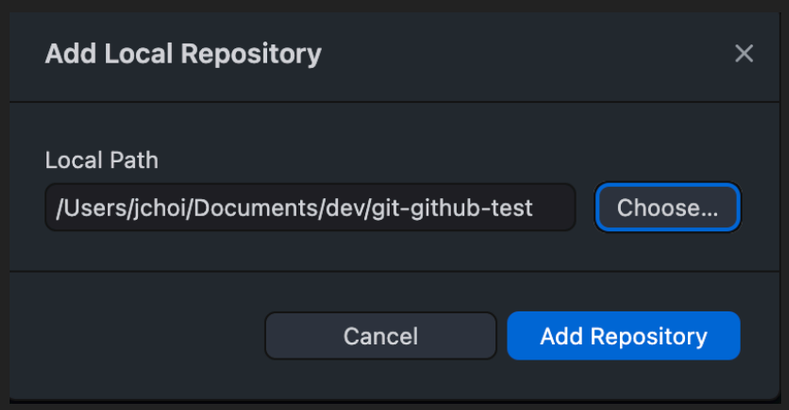
1. 터미널에서 해당 폴더에 접속하기
2. git init 입력 - 끝



GIT HUB DESKTOP

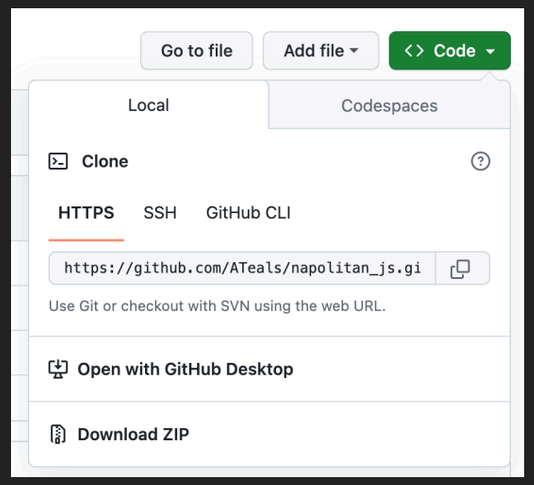
1. Current Repository 메뉴클릭
2. add 클릭
3. add Existing Repository 클릭
4. Local path 지정해주고 add Repository 클릭





GIT COMMAND

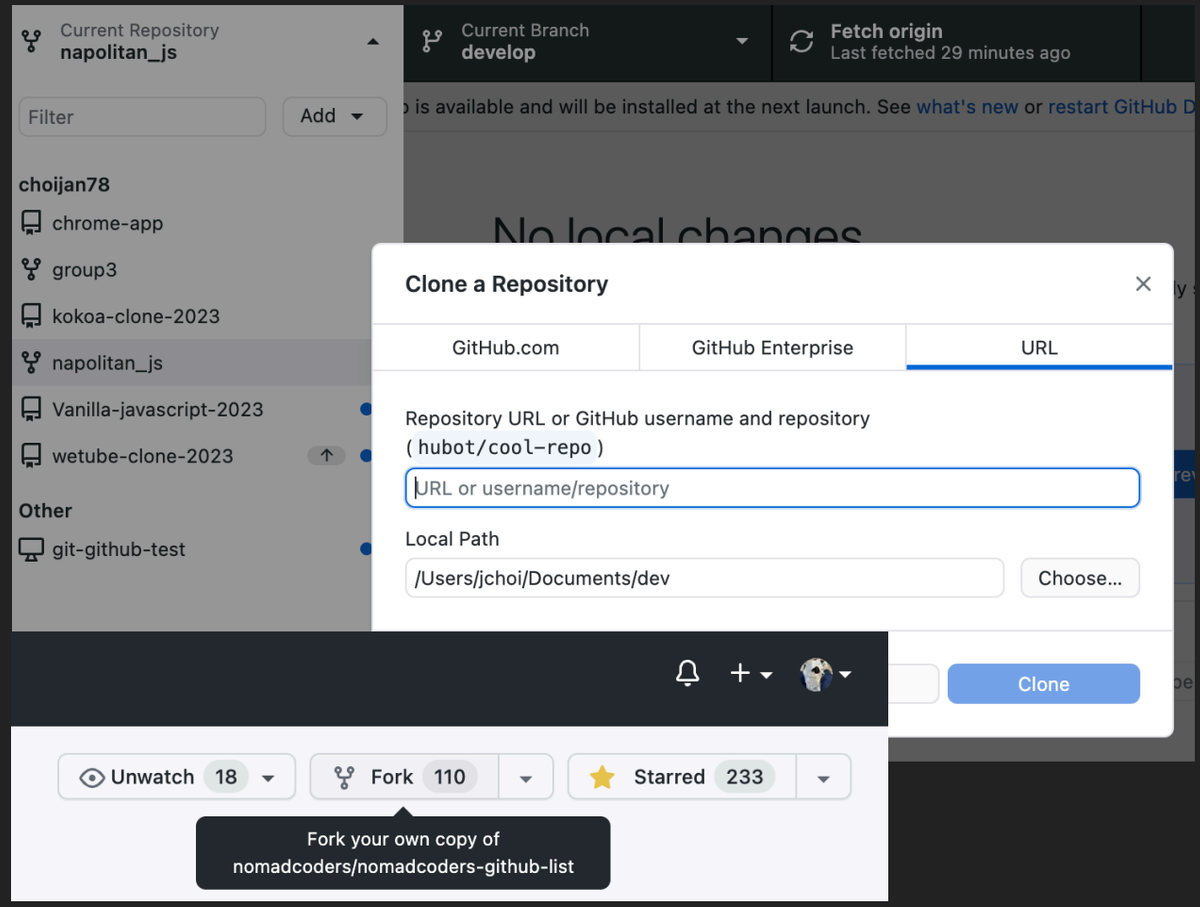
1. 리모트 저장소의 URL 복사
2. 폴더를 생성하고자 하는 곳으로 터미널 이동
3. git clone https://github.com/choijan78/napolitan_js.git
4. Clone한 폴더로 이동




```
(base) jchoi@choijan78 git-github-test % git clone https://github.com/ATeals/nap
olitan_js.git
Cloning into 'napolitan_js'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 24 (delta 3), reused 6 (delta 0), pack-reused 0
Receiving objects: 100% (24/24), 19.06 KiB | 6.35 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

GIT HUB DESKTOP

1. Current Repository 메뉴클릭
2. add 클릭
3. add Existing Repository 클릭
4. Local path 지정해주고 add Repository 클릭



GIT COMMAND

1. git remote
2. code .
3. git branch
4. git checkout develop
5. git status
6. git add .
7. git commit -m ‘commit message’
8. git log
9. git push origin develop

```
(base) jchoi@choijan78 git-github-test % git status
On branch main
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")
(base) jchoi@choijan78 git-github-test % git add .
(base) jchoi@choijan78 git-github-test % git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

GIT HUB DESKTOP

1. 브런치선택
2. vsc 에서 폴더 열고 작업
   작업과 동시에 사이드카에서 changes가 바로 올라옴 (git add가 자동으로 됨)
3. commit massage 제목 적고 커밋버튼클릭
4. Public branch 클릭시 push 됨