

Git (1)

발표자: 양유진

링크: <https://youtu.be/CKX8bT-GLZE>

Git 이란? – 용어 / 구조 / Git 기반 저장소 서비스

Git 사용법 – (1)~(6)

Git 이란?



“프로젝트의 소스 코드 버전 관리를 위해 사용하는 프로그램”

- 분산 버전 관리 시스템
- 버전관리, 백업, 협업에 주로 사용됨

Git 이란? - Git 용어

브랜치(Branch)

- : 특정 commit을 가리키는 일종의 포인터
- branch에 담긴 commit 중 가장 마지막 commit을 가리킴

헤드(HEAD)

- : 현재 브랜치가 가리키는 포인터

인덱스(Index)

- 바로 다음에 commit할 snapshot
- commit 명령어 실행했을 때, git이 처리할 것들이 있는 공간

Git 이란? - Git 구조



Git 이란? - Git 기반 저장소 서비스(Remote repository)

1. GitHub

- Public(공개) 무료, private(비공개) 유료
- 오픈소스 프로젝트 저장소로 적합



2. GitLab

- Private(비공개) 무료
- 개인 프로젝트 저장소로 적합



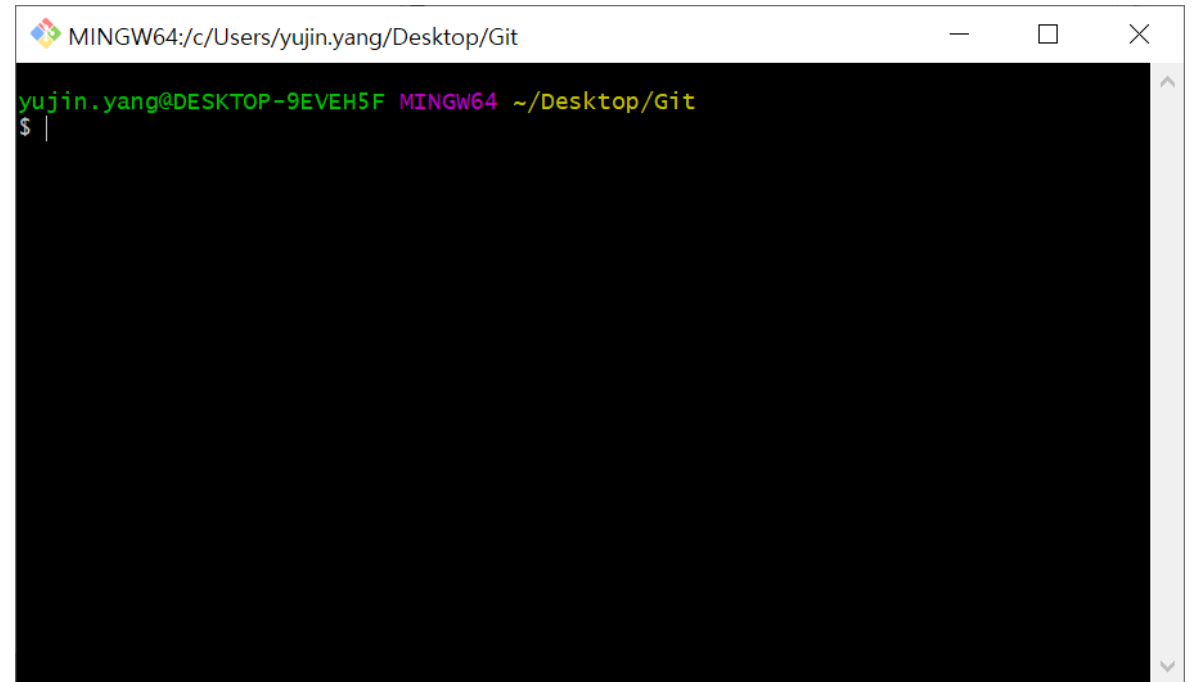
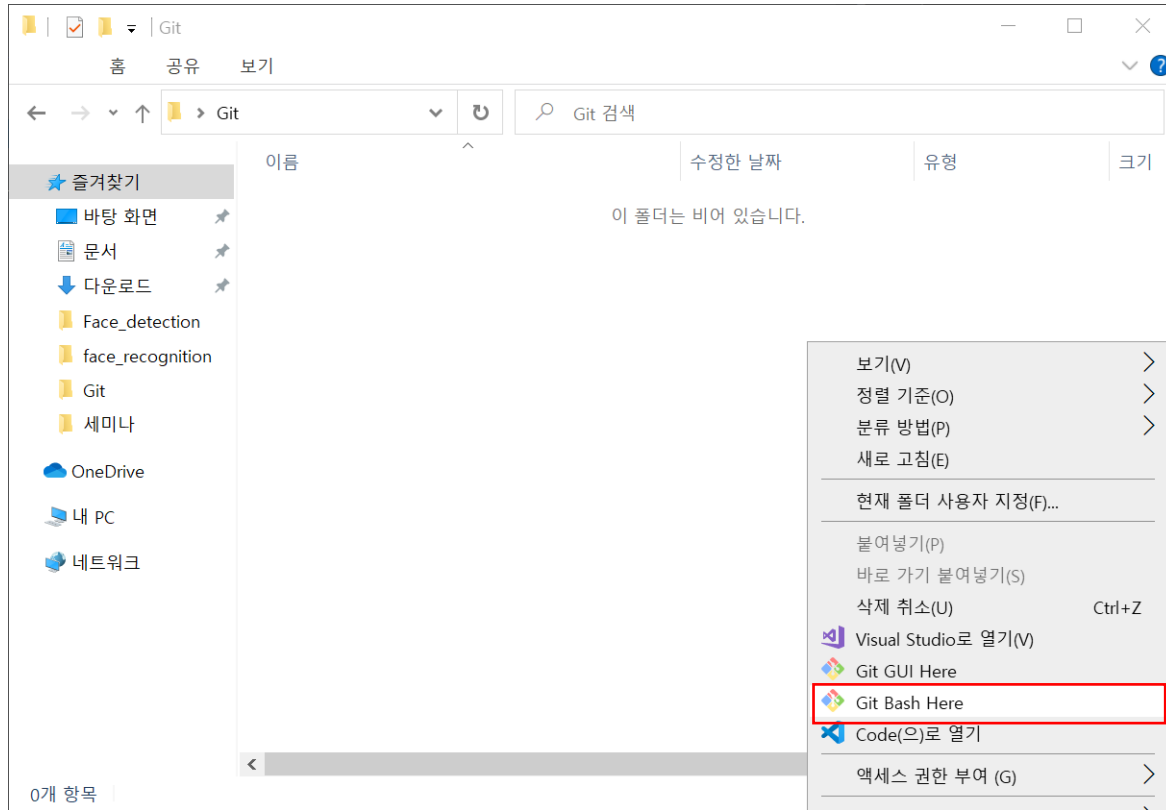
Git 사용법(1) Git 설치

1. Windows 설치방법

2. MacOS 설치방법

- Git Bash
- Terminal
- Visual Studio Code

Git 사용법(1) Git Bash 실행



Git 사용법(2) 최초 설정 - 사용자설정

`git config --global user.name "[사용자 이름]"`

:commit 할 때 사용되는 사용자 이름 정보 등록

`git config --global user.email "[이메일 주소]"`

:commit 할 때 사용되는 사용자 이름 정보 등록

```
yujin.yang@DESKTOP-9EVEH5F MINGW64 ~/Desktop/Git
$ git config --global user.name "YangYu34"

yujin.yang@DESKTOP-9EVEH5F MINGW64 ~/Desktop/Git
$ git config --global user.name "yujin.yang34@gmail.com"

yujin.yang@DESKTOP-9EVEH5F MINGW64 ~/Desktop/Git
$
```

`mkdir [디렉토리명]`

:디렉토리 생성

`cd [./디렉토리명]`

:생성한 디렉토리로 이동

`git config --list`

:설정된 정보 확인 가능

```
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=YangYu34
user.email=yujin.yang34@gmail.com
```

Git 사용법(2) 최초 설정 - 저장소 설정(+ “.git” 설명)

git init

- 해당 폴더(현재 위치)를 local repository(git 저장소)로 만들어줌.

- .git이라는 숨김파일 생성됨

```
$ git init
Initialized empty Git repository in C:/Users/yujin.yang/Desktop/Git/test_source_code/.git/
```

.git/index(파일)

- staging area에 올릴 때, index에도 기록이 됨
- index 파일은 cat 명령어로 읽으면 모두 깨져서 나옴

→ git ls-files -stage || git ls-files -s: index

: index 파일 출력 명령어

.git/objects(디렉토리)

- Git은 모든 파일을 object로 관리함
- 새로운 commit할 때마다, 새로운 object 생성
- 이런 방식이 특정 시점의 사진을 찍는 것과 비슷하다고 하여 snapshot 저장방식이라고 부름.

Git 사용법(2) 최초 설정 - 저장소 설정(+ “.git” 설명)

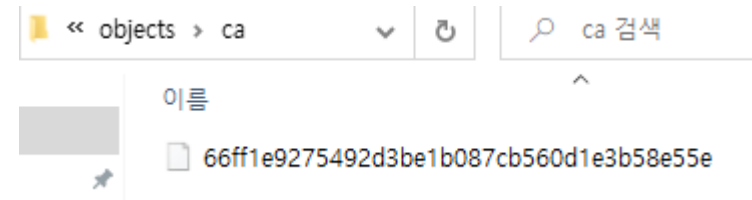
git log

:commit history 확인

```
$ git log
commit ca66ff1e9275492d3be1b087cb560d1e3b58e55e (HEAD -> master, origin/master, origin/HEAD)
```

commit hash ID(SHA-1)

= ./git/objects에서 “폴더이름+파일이름”



.git/refs/heads/master

:commit hash ID 정보가 저장되는 곳

git cat-file -t [commit hash ID]

:해당 object 파일의 type 출력

```
$ git cat-file -t ca66ff1e9275492d3be1b087cb560d1e3b58e55e
commit
```

Git 사용법(2) 최초 설정 - 저장소 설정(+ “.git” 설명)

git cat-file -p [commit hash ID]

: 해당 object 파일 정보 정갈하게 출력

```
$ git cat-file -p ca66ff1e9275492d3be1b087cb560d1e3b58e55e
tree 92f2d4479ef9035777a1d96fa1336e248729fd62
parent d708189ca47ad47ba3b560e6290f2f94a909872b
author YangYu34 <yujin.yang34@gmail.com> 1626604485 +0900
committer YangYu34 <yujin.yang34@gmail.com> 1626604485 +0900
ADD Face_detection(knn) source code
```

git cat-file -t [tree commit hash ID]

: 새로 저장한 파일 내용 저장하고 있는 blob object SHA-1 값 출력

```
$ git cat-file -t 92f2d4479ef9035777a1d96fa1336e248729fd62
100644 blob a7bb0341b751d9071ae49f9b0745c333a1ed9eab [python]find_count_to_turn
040000 tree d4d81c74de41ed484b6df97e7f0b766db7159be5 __pycache__
100644 blob bf7f65e226146015ca0731162824c57c20f89ed6 face_detection_cli.py
100644 blob 218a0fb15a4608f75c2aeeb6cf36465dadd78a13 face_recognition_cli.py
100644 blob 3724326fbd62ff0deddb8a04a494ba02f7894baf face_recognition_knn.py
100644 blob 4c396fe95022a048eca4476daf660f9fb3cc1e41 find_faces_in_picture.py
040000 tree 07c7a78e141761ab815d5b8353f7c9fef9f14bb4 knn_examples
100644 blob 8a628376109193e6ae6851c12f0cea50d248f7eb test.txt
100644 blob deac5c7574e0482f8e31299c8ca4305b1cebef92 trained_knn_model.clf
```

```
$ git cat-file -p 8a628376109193e6ae6851c12f0cea50d248f7eb
This is a commit test text file.
```

test.txt 파일 내용 출력

Git 사용법(2) 최초 설정 - 원격 저장소와 연결

```
yujin.yang@DESKTOP-9EVEH5F MINGW64 ~/Desktop/Git (master)
$ git clone https://github.com/YangYu34/test_source_code.git
Cloning into 'test_source_code'...
warning: You appear to have cloned an empty repository.
```

git clone [git repository주소]

:특정 원격 저장소와 로컬 PC의 저장소를 연결하고 데이터를 복사하여 가져옴.

원격저장소 이름

git remote add origin [git repository URL]

:특정 원격 저장소와 로컬 PC의 저장소 연결.

git remote -v

:연결된 원격 저장소 url 확인

```
$ git remote -v
origin https://github.com/YangYu34/test_source_code.git (fetch)
origin https://github.com/YangYu34/test_source_code.git (push)
```

git remote set-url origin [변경할 git repository URL]

:origin에 설정된 remote url 변경

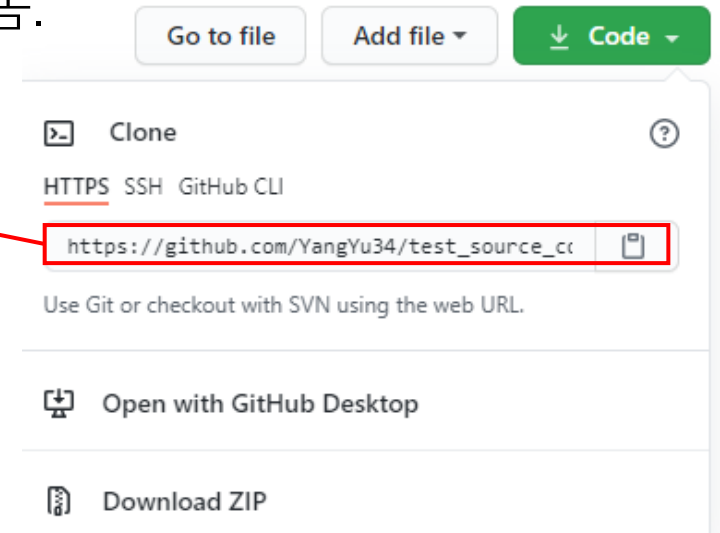
git remote rm origin

:원격저장소 삭제

```
$ git remote add origin https://github.com/YangYu34/test_source_code.git
error: remote origin already exists.
```

git remote rename [변경 전 저장소 이름] [변경 후 저장소 이름]

:원격저장소 이름 변경



Git 사용법(3) staging area로 파일 올리기

git status

: repository 상태 확인

- staging area로 추가할 만한 사항이 있는지 확인해줌.

```
$ git status
On branch master

No commits yet

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

nothing to commit / working directory clean

: 폴더에 commit할 만한 파일 없음

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  find_faces_in_picture.py

nothing added to commit but untracked files present (use "git add" to track)
```

untracked (빨간색)

: staging area에 없는 파일이 working directory에 생긴 경우

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new file:   find_faces_in_picture.py
```

tracked (초록색)

: add명령어를 통해 파일이 staging area로 옮겨짐

Git 사용법(3) staging area로 파일 올리기

git add .

: untracked 파일 전부 working directory → staging area 추가

git add [파일명]

: 지정한 파일만 working directory → staging area 추가

```
$ git add .  
warning: LF will be replaced by CRLF in face_detection_cli.py.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in face_recognition_cli.py.  
The file will have its original line endings in your working directory
```

유닉스 시스템: 개행문자로 LF(Line Feed) 사용

윈도우: 개행문자로 CRLF(Carriage Return과 Line Feed) 사용

→ Git이 둘 중 어느 방법을 선택할지 혼란이 와서 발생하는 에러

git config --global core.autocrlf true

LF: (\0x0a, \n) 새로운 행 추가
CR: (\0x0d, \r) 시작위치로 복귀
CRLF는 LF보다 1Byte더 크게 적용됨

Git 사용법(3) staging area로 파일 올리기 - 작업 되돌리기


git reset HEAD [파일이름]

: tracked된 상태를 untracked상태로 바꿀 수 있음

```
$ git reset HEAD reset.txt
```

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

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   reset.txt
```



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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    reset.txt
```

[옵션] - 범위의 차이

--soft: local repository 이전으로 되돌림

--mixed (default): staging area 이전으로 되돌림

--hard : working directory 이전으로 되돌림

git reflog

: 명령어로 삭제된 commit id 확인

git reflog --hard [commit Hash ID]

: commit 복구

Git 사용법(4) commit하기(local repository로 올리기)

`git commit -m "[commit message]"`

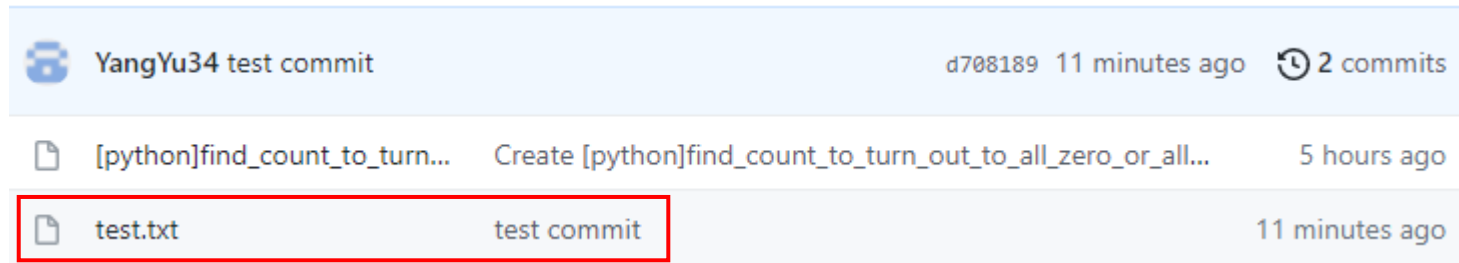
: staging area에 있는 정보 commit하여 local repository에 변경사항 적용

```
$ git commit -m "ADD Face_detection(knn) source code"
[master ca66ff1] ADD Face_detection(knn) source code
18 files changed, 443 insertions(+)
create mode 100644 __pycache__/__init__.cpython-39.pyc
create mode 100644 __pycache__/api.cpython-39.pyc
create mode 100644 __pycache__/face_recognition_cli.cpython-39.pyc
create mode 100644 face_detection_cli.py
create mode 100644 face_recognition_cli.py
create mode 100644 face_recognition_knn.py
create mode 100644 find_faces_in_picture.py
create mode 100644 knn_examples/fonts/SeoulNamsanM.ttf
create mode 100644 "knn_examples/test/C\354\226\270\354\226\264\354\240\225\354\204\235_\354\240\225\354\230\244\355\221\234.pdf"
create mode 100644 knn_examples/test/L and other people.jpg
create mode 100644 knn_examples/test/L.jpg
create mode 100644 "knn_examples/test/\352\263\240\354\225\204\353\235\274.jpg"
create mode 100644 "knn_examples/test/\354\234\240\354\236\254\354\204\235.jpg"
create mode 100644 knn_examples/train/L/L.jpg
create mode 100644 knn_examples/train/L/L0.jpg
create mode 100644 knn_examples/train/L/L1.jpg
create mode 100644 "knn_examples/train/\352\263\240\354\225\204\353\235\274/\352\263\240\354\225\204\353\235\2741.jpg"
create mode 100644 trained_knn_model.clf
```

Git 사용법(5) push하기(remote repository에 반영하기)

git push origin **[branch명]**

: remote repository(origin)에 파일 업로드



```
$ git push
To https://github.com/YangYu34/test_source_code.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/YangYu34/test_source_code.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

git pull origin **[branch명]**

: remote repository(origin)에 업데이트된 내용 받아와 동기화

Git 사용법(6) branch 생성, 삭제

git checkout -b [branch명]

: 새로운 branch 생성

- 생성과 동시에 새로 생성된 branch로 변경됨

git branch -a

: 생성된 branch 확인

- 초록색: 현재 local에서 사용중인 branch

- 흰색: local에 생성된 branch

- 빨간색: 원격저장소에 생성된 branch

```
$ git branch -a
* master
remotes/origin/HEAD -> origin/master
remotes/origin/main
remotes/origin/master
```

git branch -d [branch명]

: local repository에서 branch 삭제

```
$ git branch -d main
Deleted branch main (was ca66ff1).
```

```
$ git branch -a
* master
remotes/origin/HEAD -> origin/master
remotes/origin/master
```

git checkout [branch명]

: 해당 branch로 이동

git push origin --delete [branch명]

: remote repository에서 branch 삭제

```
$ git push origin --delete main
To https://github.com/YangYu34/test_source_code.git
- [deleted]          main
```

```
$ git branch -a
* master
remotes/origin/HEAD -> origin/master
remotes/origin/master
```

감사합니다