

19110107 황은영

Git 과제

[2강. 연습문제]

아래의 과정을 캡처하여 GitLab에 업로드 하시오.

1. 깃을 사용하기 위해 여러분의 사용자 이름과 이메일을 넣어서 계정 초기화

```
MINGW64/c/Users/사용자/schoolgit/hello-git
examine the history and state (see also: git help revisions)
bisect      Use binary search to find the commit that introduced a bug
diff        Show changes between commits, commit and working tree, etc
grep        Print lines matching a pattern
log          Show commit logs
show        Show various types of objects
status      Show the working tree status

grow, mark and tweak your common history
branch       List, create, or delete branches
commit       Record changes to the repository
merge        Join two or more development histories together
rebase       Reapply commits on top of another base tip
reset        Reset current HEAD to the specified state
switch       Switch branches
tag          Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
fetch        Download objects and refs from another repository
pull         Fetch from and integrate with another repository or a local branch
push         Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git config --global user.name "gitnickname"
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git config --global user.email "popshopoping@gmail.com"
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ |
```

2. 깃 초기화 명령

git init

```
Reinitialized existing Git repository in C:/Users/사용자/schoolgit/hello-git/
Date: Tue Feb 21 07:04:16 2023 +0900
message 1
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ pwd
/c/Users/사용자/schoolgit/hello-git
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ cd hello-git
bash: cd: hello-git: No such file or directory
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ pwd
/c/Users/사용자/schoolgit/hello-git
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git init
Reinitialized existing Git repository in C:/Users/사용자/schoolgit/he
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ |
```

3. 깃 상태를 확인하는 명령

git log git satus

```
사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git init
Reinitialized existing Git repository in C:/Users/사 용 자 /schoolgit/hello-git/

사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ pwd
/c/Users/사 용 자 /schoolgit/hello-git

사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git log
commit 3a69459f232efffaeea54932d24751cf367cb087c (HEAD -> master)
Author: gitnickname <popshopoping@gmail.com>
Date: Tue Feb 21 07:08:55 2023 +0900

    message 2

commit 7269d4c842fcfe8420342b1bbe9e5116c89e2498
Author: gitnickname <popshopoping@gmail.com>
Date: Tue Feb 21 07:04:16 2023 +0900

    message 1

사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$
```

4. hello.txt를 스테이징

git add(에이징) git add hello.txt

```
사 용 자 @DESKTOP-Q3B6S8B MINGW64
$ vi hello.txt

사 용 자 @DESKTOP-Q3B6S8B MINGW64
$ |
```

5. hello.txt를 커밋

git commit -m 초기 스테이징, 커밋 후엔 git commit -am로 에이징과 커밋 사용가능

```
fatal: pathspec 'hello2.txt' did not match

사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ vi hello.txt

사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git commit -m "커밋 세 키 기"
```

6. 커밋 한 결과 확인

git status git log

```
사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git status
On branch master
nothing to commit, working tree clean

사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git log
commit 6918dfa3bbbd812676576418f26265b45b4e65b2
Author: gitnickname <popshopoping@gmail.com>
Date: Tue Feb 21 07:08:55 2023 +0900

    message 커밋 수정

commit 7269d4c842fcfe8420342b1bbe9e5116c89e2498
Author: gitnickname <popshopoping@gmail.com>
Date: Tue Feb 21 07:04:16 2023 +0900

    message 1

사 용 자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$
```

3강. 아래의 동작을 수행하고 캡처하여 GitLab에 제출하시오.

1. 커밋 한 소스와 최근까지 작업한 소스 사이의 차이를 확인하는 명령
git diff

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit
$ git diff
warning: in the working copy of 'hello.txt'
the time Git touches it
diff --git a/hello.txt b/hello.txt
index 1191247..b6bd080 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,2 @@
 1
-2
+two
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit
```

2. 스테이징 되지 않은 파일을 이전 상태로 되돌리는 명령(파일명은 hello.txt)
git checkout (버전2까지) git restore -- hello.txt

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git
$ git restore
fatal: you must specify path(s) to restore

사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git
$ git restore -- hello.txt

사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git
$ git status
```

3. 스테이징이 완료된 파일을 언스테이징 상태로 되돌리는 명령
git reset(버전 2.23이전) 이후 restore명령어 써도됨
git restore --staged <파일명>

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (
$ git restore --staged hello2.txt

사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be comm
  (use "git restore <file>..." to discard changes in w
        modified:   hello2.txt

no changes added to commit (use "git add" and/or "git
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (
$ |
```

4. 커밋이 완료된 파일을 언스테이징 상태로 되돌리는 명령

`git reset <브랜치명>^`

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git
$ git reset HEAD^
Unstaged changes after reset:
M   hello2.txt

사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git
$ git log
commit 02721a8dfacea1cbf938ea94c7348ab7ce69a037e (HEAD)
Author: gitnickname <popshopoping@gmail.com>
Date:   Tue Feb 21 07:43:37 2023 +0900

    message 3 modified

commit 6918dfa3bbbd812676576418f26265b45b4e65b2
Author: gitnickname <popshopoping@gmail.com>
Date:   Tue Feb 21 07:08:55 2023 +0900

    message 커밋 수정

commit 7269d4c842fcfe8420342b1bbe9e5116c89e2498
Author: gitnickname <popshopoping@gmail.com>
Date:   Tue Feb 21 07:04:16 2023 +0900

    message 1
```

5. 특정 커밋의 버전으로 돌아가는 명령(임의로 해시값을 “a1b2”로 함)

`git reset --hard 특정해쉬값`

```
$ git reset --hard 6918dfa3bbbd812676576418f26265b45b4e65b2
HEAD is now at 6918dfa message 커밋 수정

사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git (master)
$ git status
On branch master
nothing to commit, working tree clean
```

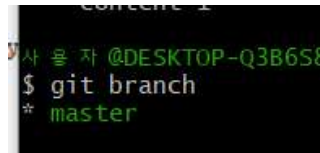
`git commit --amend ==커밋수정`

브랜치 생성 및 정보 관리

[4강. 아래의 동작을 수행하고 캡처하여 GitLab에 제출하시오.]

1. 어떤 브랜치에서 작업중인지 알아보는 명령

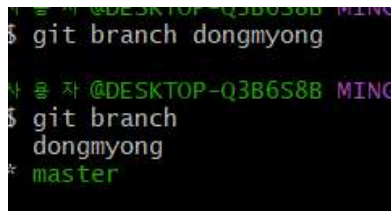
git branch



```
content 1
사용자 @DESKTOP-Q3B6S8B
$ git branch
* master
```

2. 브랜치를 추가하는 명령(추가할 브랜치명은 dongmyong)

git branch dongmyong



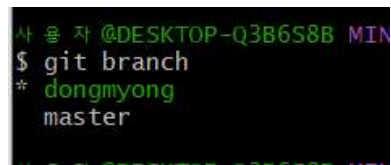
```
사용자 @DESKTOP-Q3B6S8B MING
$ git branch dongmyong

사용자 @DESKTOP-Q3B6S8B MING
$ git branch
dongmyong
* master
```

3. dongmyong 브랜치로 변경하는 명령

checkout은 쓰지말자

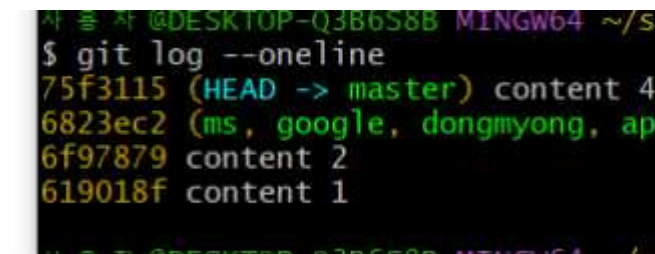
git switch 브랜치명



```
사용자 @DESKTOP-Q3B6S8B MING
$ git switch dongmyong
* dongmyong
master
```

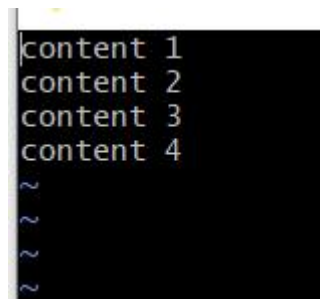
4. 커밋 로그를 간략하게 보는 명령

git log --oneline



```
사용자 @DESKTOP-Q3B6S8B MINGW64 ~/s
$ git log --oneline
75f3115 (HEAD -> master) content 4
6823ec2 (ms, google, dongmyong, ap
6f97879 content 2
619018f content 1
```

5. 수정한 전체 파일을 스테이지에 올리는 명령



```
content 1
content 2
content 3
content 4
~
~
~
~
~
```

5강. 아래의 동작을 수행하고 캡처하여 GitLab에 제출하시오.

1. 커밋 로그 명령에서 브랜치 사이의 차이점을 알아보는 명령

```
$ git log --oneline --branches
e68f3e2 (HEAD -> master, dongmyong) new content 3
3f5cde4 (o2) o2 work 2
0268540 master work 2
3256285 work 1
```

사용자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git

2. 1번 명령어를 수행할 때 시각적으로 보여주기 위해 그래프를 표현하도록 하는 옵션 명령어

```
사용자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git
$ git merge <dongmyong>
bash: syntax error near une
```

3. dongmyong 브랜치를 현재 master 브랜치로 병합하는 명령

```
사용자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git
$ git merge <dongmyong>
bash: syntax error near une
```

4. git에서 소스 사이에 충돌이 발생할 경우 해결하는 방법

6강. 아래의 동작을 수행하고 캡처하여 GitLab에 제출하시오.

1. 브랜치 사이에 커밋을 비교하는 명령(master 기준에서 dongmyong 브랜치와의 비교)

git diff

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/h
$ git diff

사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/h
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  dongmyong/

nothing added to commit but untracked files present
```

2. 가장 최근의 커밋을 되돌리는 명령

git revert

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/st (master)
$ git revert
usage: git revert [--[no-]edit] [-n] [-m <parent-name>] [-s]
       or: git revert (--continue | --skip | --abort | --quit)
       --quit          end revert or cherry-pick operation
       --continue      resume revert or cherry-pick operation
```

3. 현재 작업중인 소스 파일들의 커밋을 뒤로 미루기 위한 명령

```
사용자@DESKTOP-Q3B6S8B MINGW64
$ git merge o2
Merge made by the 'ort' strategy
 o2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 o2.txt
```

4. 따로 보관해 두었던 작업 소스들을 가져오는 명령

```
사용자@DESKTOP-Q3B6S8B MINGW64
$ git merge <dongmyong>
bash: syntax error near unexpected token '<dongmyong>'
```

7강. 아래의 동작을 수행하고 캡처하여 GitLab에 제출하시오.

1. 원격 저장소랑 로컬 저장소를 연결하는 명령

git remote add origin <github주소> / git push

```
사용자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git/loc-git (master)
$ git remote add origin https://github.com/youngogo123/remote-git.git

사용자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-git/loc-git (master)
$ |
```

2. 연결이 제대로 되어있는지 확인 하는 명령

git push -u origin(처음) /git push

```
사용자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/hello-
$ git push -u origin main
fatal: User canceled device code authentication
error: unable to read askpass response from 'C:/P
e'
Username for 'https://github.com':
```

3. 원격 저장소에서 커밋 완료된 최신 버전을 로컬로 당겨서 가져오는 명령

git pull origin main

```
사용자 @DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/he
$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), 688 bytes | 22
From https://github.com/youngogo123/remote-gi
* branch          main          -> FETCH_HEAD
```


8강. 아래의 동작을 수행하고 캡처하여 GitLab에 제출하시오.

1. 원격 저장소를 myhome 지역 저장소로 복제

git clone 주소 파일이름

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit
$ git clone https://github.com/youngogo123/remote-git.git git-home
Cloning into 'git-home'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 1), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (1/1), done.

사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit
$ ls
git-home/  hello-git/  st/  test/
```

2. 자신의 소스를 커밋하기 전에 항상 먼저 수행해야 할 명령

git pull (최근의 커밋이 있는지 확인)

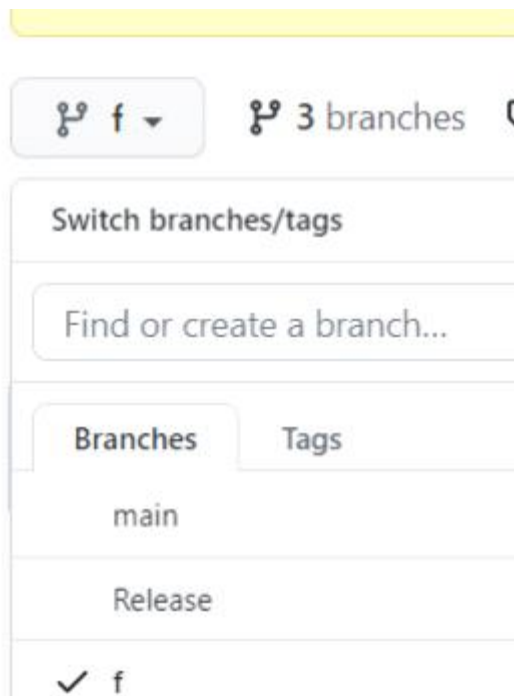
```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/git-home
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 278 bytes | 6.00 KiB/s
```

3. 'Release' 브랜치를 만들고 원격 저장소에 푸시하는 명령

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit
$ git branch Release

사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit
$ git branch
Release
* main
```

```
사용자@DESKTOP-Q3B6S8B MINGW64 ~/schoolgit/git-home (Release)
$ git push origin Release
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 270 bytes | 135.00 KiB/s, done.
```



4. 'Release' 브랜치의 내용이 전혀 문제가 없다고 판단되면 이 내용을 'main' 브랜치에 병합할 때 제약 사항

9강. 아래의 동작을 수행하고 캡처하여 GitLab에 제출하시오.

1. 여러분만의 깃헙 원격 저장소를 생성한 후 개인 프로파일을 작성하고 특정 repository 안에 readme.md 파일을 생성

추가문제

1. 아래 화면은 사용자가 d:폴더에 위치해 있을 때 사용자 id를 id01@tu.ac.kr로 정하고

project01 폴더 안에서 git으로 초기화 한 후 a.txt 파일을 스테이징하여 “init” 메시지로 커밋을 한 결과이다. 명령어를 순서대로 아래 공백 안에 나열하세요. (단, a.txt 파일을 새로 만들거나 아니면 복사해서 폴더 안에 만들어 두었다고 가정함)

```
mkdir project01
vi a.txt
git add .
git commit -m "init"
git log
```

2. main 브랜치를 만들고 main 브랜치로 전환한 다음 master 브랜치를 삭제하세요

```
git branch main
git switch main
git branch -d master
```

3. test 브랜치를 만든 후 test 브랜치로 변경한 다음 c.txt 파일을 추가했습니다. test 브랜치에서 스테이징과 커밋까지 마쳤고, c.txt파일이 어떠한 문제도 없다고 판단되어 main 브랜치에 이 c.txt파일을 병합하는 명령어를 작성하세요. 즉, main 브랜치에 a.txt, b.txt, c.txt 3개의 파일이 존재하도록 병합하세요.(단, 현 브랜치 위치는 main 브랜치입니다)

```
git merge <test>
```

4. 이 명령어로 병합이 잘 되었다면 test 브랜치와 main 브랜치는 각각 3개의 파일이 존재하게 됩니다. test 브랜치 명을 deploy 브랜치 명으로 바꿔보세요

```
git mbranch -m deploy
```

5. github.com 에 “id01” 이라는 계정이 생성되어 있고 이 계정 아래에 “git_study”라는 리포지토리를 만들어 놓 상태를 가정하면 이 레포지토리에 원격 연결하는 명령어를 작성하세요(단, 리포지토리는 https://github.com/id01/git_study.git)

```
git remote add origin <https://github.com/id01/git\_study.git>
```

6. 현 디렉토리 안에 있는 두 a.txt, b.txt, c.txt가 그림처럼 리포지토리에 올라갈 수 있도록 푸시하는 명령어를 작성하세요

```
git branch -M main    (main으로 브랜치 바꿈)  
git push -u origin    (a.txt)  
git push
```

7. HEAD를 B위치로 옮겨서 C와 D를 삭제하고 되돌아가는 명령어를 적어보세요

```
git switch a0fvf8
```

9. 학교에서 작업을 다 마치고 집에 돌아온 후 집 컴퓨터에서 방금 학교에서 작업했던 소스들을 복사하여 리포지토리를 연동하고자 합니다. https://github.com/id01/git_study.git 이 리포지토리를 로컬 디렉토리 E:\Workspace\ 디렉토리 안에 복제하는 명령어를 작성하세요(단, 현 디렉토리는 E:\Workspace입니다)

```
git clone https://github.com/id01/git\_study.git E:\Workspace\
```