

오픈소스sw의 이해

Lecture #7: git instructions

Software Engineering laboratory
Yeungnam University

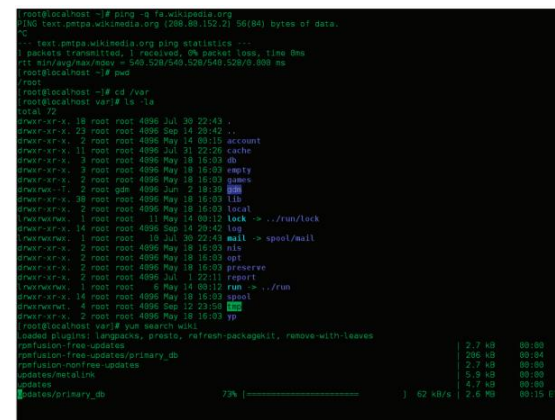


git 명령어로 버전 관리하기

■ git 명령어를 학습해야 하는 이유

- SourceTree vs. git 명령어 활용

- SourceTree를 사용하기도 하지만 SW 개발이 익숙한 개발자들은 대부분 git 명령어를 활용하고자 함.
 - 훨씬 더 빠르고 편리하기 때문임.
 - git 명령어를 한번 손에 익혀두면 명령어 한두 줄로 간단하게 버전을 다룰 수 있음.
 - 이는 버전을 관리할 때마다 일일이 SourceTree를 열고 마우스로 버튼을 클릭하는 것보다 훨씬 빠르고 간편한 방식.
 - 더욱이 어떤 개발 환경에서는 소스트리를 설치할 수 없을 수도 있음.
 - 모든 개발자가 그래픽 환경에서 개발하는 것은 아니기 때문임.



git 명령어로 버전 관리하기

■ git 명령어 학습하기

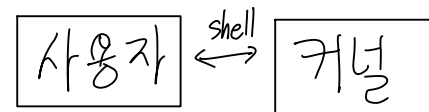
- 우선, SourceTree에서 시각적으로 이해한 형상관리 내용을 충분히 이해하고 있어야 함.
- 다음으로, 명령어의 반복적인 사용
 - 우리가 클릭과 더블클릭을 굳이 하나하나 생각하며 구분하지 않듯이 명령어의 의미를 한번 이해하고 반복해서 사용하다 보면 자연스럽게 체화할 수 있음.
 - 어느 순간부터는 깃 명령어를 칠 때 머리가 아닌 손이 먼저 반응하게 될 것.

git bash 열기

■ git bash shell

- bash shell은 Linux에서 가장 널리 사용되는 셸.

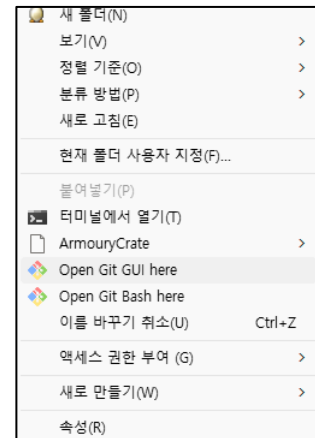
- shell: 사용자와 커널 사이의 매개체 역할을 하는 프로그램 (사용자로부터 명령을 받아서 그것을 프로세싱하기 위해 커널에게 넘겨주는 일을 하는 프로그램).
- bash: bourne Again Shell의 축약어.



■ git bash 열어서 git 명령어 입력 준비

- 특정 폴더에서 마우스 우측버튼 클릭 git bash 열기 or 하단 작업표시줄에서 git bash 입력
 - (예) C:\test (작업 디렉터리를 C:\test로 설정)
 - 해당 경로에서 마우스 오른쪽 버튼을 클릭한 후 Open Git Bash here를 클릭

```
MINGW64:/c/test
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test
$
```



버전 만들기

- git init : 로컬 저장소 생성하기 = clone

- 로컬 저장소를 만들려는 경로에서 git bash 열기

- (예) C:\test (작업 디렉터리를 C:\test로 설정)

- 해당 경로에서 마우스 오른쪽 버튼을 클릭한 후 Open Git Bash here를 클릭

- git init 입력

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test  
$ git init  
Initialized empty Git repository in C:/test/.git/
```

버전 만들기

■ git status : 작업 디렉터리 상태 확인하기 (사용빈도 높음)

● 특정 경로에서 git bash 열기

■ git status 입력

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git status
On branch master

No commits yet

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

■ 해당 폴더에 A가 적힌 a.txt 파일을 만들 경우,

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git status
On branch master : 현재 기본 브랜치

No commits yet : 현재 커밋X

Untracked files: : 기존에 변경사항을 추적하지 않은 파일
  (use "git add <file>..." to include in what will be committed)
    a.txt

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

- On branch master: 현재 기본 브랜치, 즉 master 브랜치에 있다는 의미.
- No commits yet: 현재 아무런 커밋도 하지 않았음을 의미.
- 마지막으로 Untracked files: git이 기존에 변경 사항을 추적하지 않은 대상을 나타냄.
 - 여기에 a.txt가 표시
 - 이는 기존에 버전을 관리한 적 없던 a.txt라는 새로운 파일이 생성되었음을 의미

버전 만들기

■ git add : 스테이지에 올리기

● git add <스테이지에 추가할 대상>

■ (예) git add a.txt

- Add할 파일이 여러 개일 경우, 현재 디렉터리에 있는 모든 변경사항을 한번에 스테이지로 추가
Git add -A : 프로젝트 내 모든 폴더 및 파일들을 stage에 올림.
Git add . : 현재 디렉터리에 있는 파일들을 stage에 올림.

● git status 로 작업 디렉터리 확인

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git add a.txt

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git status
On branch master

No commits yet

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

버전 만들기

- git commit : 스테이지에 올린 변경사항들을 커밋하기

- git commit -m “커밋 메시지 제목” or
- git commit -message “커밋 메시지 제목”

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git commit -m "first commit"
[master (root-commit) 0a295a9] first commit
1 file changed, 1 insertion(+)
create mode 100644 a.txt
```

- git commit
 - commit 메시지 제목 및 본문 내용 입력 가능

버전 만들기

■ git commit 수행시 add 및 commit 동시 수행

- git commit -am “커밋 메시지 제목” or
- git commit -a -m “커밋 메시지 제목” or
- git commit --all --message “커밋 메시지 제목”

■ git log

- 저장소의 커밋 목록 출력
- 커밋 해시, 만든 사람, 커밋이 만들어진 날짜, 커밋 메시지가 출력
- 커밋 해시 우측의 HEAD -> master는 현재 HEAD가 master 브랜치에 있음을 나타냄

(주의)

- git commit -am "커밋 메시지" 명령은 깃이 변경 사항을 추적하는(tracked) 파일에만 사용 가능.
- 다시 말해, 스테이지에 이미 올라와 있거나 한 번이라도 커밋한 적이 있는 파일에만 사용할 수 있음.
- git이 기존에 변경 사항을 추적하지 않은(untracked) 파일은 이 명령어를 사용할 수 없음.

(실습)

■ add 및 commit 동시 수행 예제 실습

- a.txt 파일에 A를 추가



- git status

- a.txt 파일이 추가되었음을 확인

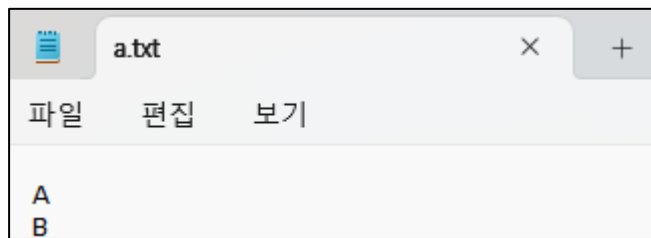
- git add a.txt *스테이지 등록*

- git commit -m "first commit" *커밋*

(실습)

■ add 및 commit 동시 수행 예제 실습

- a.txt 파일에 B를 추가



- git status

- a.txt 파일이 수정되었음을 확인

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git status
On branch 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:   a.txt

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

(실습)

- add 및 commit 동시 수행 예제 실습
 - a.txt 파일을 add 및 commit 동시에 수행하기

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git commit -am "second commit"
[master 593a62e] second commit
1 file changed, 2 insertions(+), 1 deletion(-)
```

- git log
 - 2번째 커밋도 성공했음을 확인

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git log
commit 593a62e285322ed99cda05059459207b8a47a98b (HEAD -> master)
Author:
Date:
    second commit

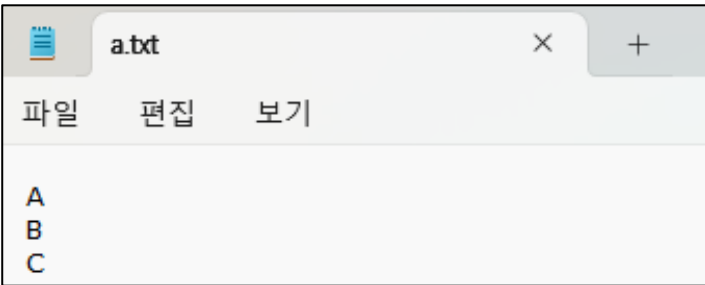
commit 0a295a93d9583b2d22bedf2f49bba7ceba0e860c
Author:
Date:
    first commit

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$
```

(실습)

■ commit 시 제목 및 메시지 입력해보기

- a.txt 파일에 C를 추가



- git add a.txt
 - 스테이지에 추가

(실습)

■ commit 시 제목 및 메시지 입력해보기

● git commit

- 커밋 메시지 제목: 첫번째 줄부터 작성
 - (예) third commit
- 커밋 메시지 내용: 세번째 줄부터 작성
 - (예) This is my third commit.
- 입력모드 : a 또는 i
- 명령 모드: esc
 - :write 또는 :w 입력한 내용이 저장
 - :quit 또는 :q 입력 창이 닫힘
 - :wq 입력한 내용 저장 and 창이 닫힘
- git log를 통해 commit 된 정보 확인



The screenshot shows a terminal window titled 'MINGW64:/c/test'. The user has entered 'third commit' as the commit message, which is highlighted with a red box. Below it, the commit message content 'This is my third commit.' is also highlighted with a red box. The terminal displays the standard git commit prompt: '# Please enter the commit message for your changes. Lines starting with '#' will be ignored, and an empty message aborts the commit.' It also shows the current date and time, the branch name 'master', and the changes to be committed: 'modified: a.txt'. The bottom status bar indicates the file being edited is '.git/COMMIT_EDITMSG' and shows the cursor position '1,1 All'.

• (참고) git commit 메시지 수정하기 : git commit --amend

버전 만들기

■ git log: 커밋 조회하기

● git log

- 저장소의 커밋 목록 조회하기

● git log --oneline

- 커밋 목록을 커밋당 한 줄로 출력해주는 옵션
- 커밋이 매우 복잡하고 많이 쌓여 있는 상황에서 요긴하게 사용
- 이 명령은 다음과 같이 짧은 커밋 해시와 커밋 메시지 제목만을 출력

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master)
$ git log --oneline
f52c589 (HEAD -> master) third commit
593a62e second commit
0a295a9 first commit
```

짧은 커밋 해시

커밋 메시지 줄역

버전 만들기

■ git log의 추가 옵션

- git log --patch

- git log -p

- 각각의 커밋이 파일에서 어떤 내용을 변경했는지 상세하게 제공.

- git log --graph

- 각 커밋을 그래프의 형태로 출력하는 방법.

- SourceTree의 커밋 그래프와 유사함.

- 브랜치가 여러 개로 나뉘어지고 합쳐지는 환경에서
--graph 옵션을 이용하면 브랜치별 커밋의
가독성을 높일 수 있음.

```
ysseo@DESK
$ git log
* commit f
  Author:
  Date:
    thir
    This
* commit 5
  Author:
  Date:
    seco
* commit 0
  Author:
  Date:
    firs
```

```
| *   commit 5
|   \ Merge: 0
|   / Author:
|   / Date:
|   \
|   \ Merge
|   \
|   \ *   commit e
|   \ \ Merge: 8
|   \ / Author:
|   \ / Date:
|   \
|   \ Merge
```


Summary

- git bash에서 사용하는 일반적인 명령어 흐름
 - git init
 - cat > test.txt (텍스트 파일 작성, 파일 작성 후 ctrl + d 로 종료)
 - ls test.txt (파일 목록 확인)
 - git add test.txt
 - (git status)
 - git commit -m "add text.txt"
 - (git log)

브랜치 관리하기

- `git branch <브랜치>`

- 브랜치 나누기
- (예) foo라는 브랜치를 만드는 명령은 `git branch foo`

- `git checkout <브랜치>`

- 특정 브랜치로 체크아웃하기
 - 해당 브랜치로 작업 환경을 바꾸는 것

- `git diff <브랜치><브랜치>`

- 브랜치 간의 차이 확인하기

브랜치 관리하기

■ 현재 브랜치 확인하기

- git bash에서 확인

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master) $
```

현재 브랜치 확인

- git branch 명령어 사용

- 현재 브랜치의 목록과 함께 현재 여러분이 작업 중인 브랜치가 *로 표시
- 어떤 브랜치도 만들지 않았고, master 브랜치에서만 작업하고 있는 경우

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test (master) $ git branch
* master
```

원격 저장소와 상호 작용하기

■ GitHub와 상호작용하기

- 클론(clone): 원격 저장소를 복제하기
- 리모트(remote): 원격 저장소를 추가하고, 조회하고, 삭제하기
- 푸시(push): 원격 저장소에 밀어넣기
- 패치(fetch): 원격 저장소를 일단 가져만 오기
- 풀(pull): 원격 저장소를 가져와서 합치기 *fetch + merge*

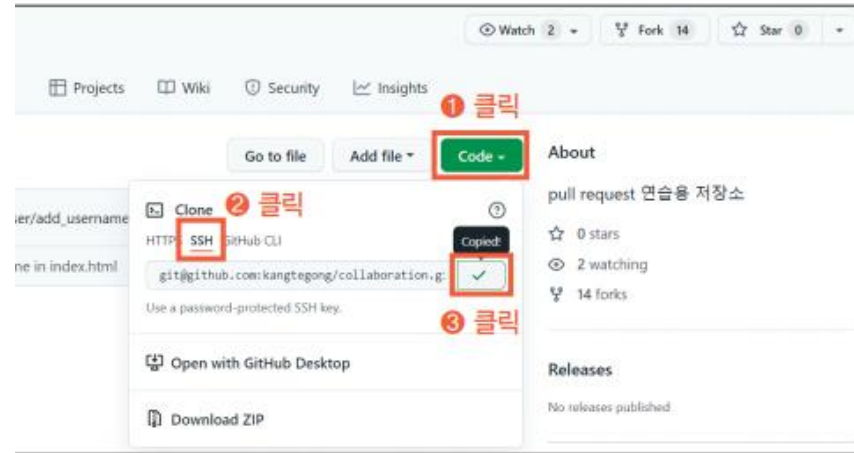
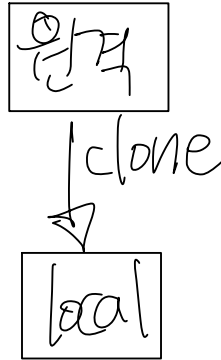
원격 저장소와 상호 작용하기

■ git clone <원격저장소 경로>

- 원격 저장소 복제하기 현재 디렉토리에 원격저장소와 동일한 이름의 폴더 생성
- (예)

■ git clone <원격저장소 경로>

■ git clone <원격저장소 경로> <클론받을 경로> // 로컬에 특정 경로 지정 가능



```
minchul@DESKTOP-9KULGUE MINGW64 /c
$ git clone git@github.com:kangtegong/collaboration.git
Cloning into 'collaboration'...
remote: Enumerating objects: 102, done.
remote: Counting objects: 100% (102/102), done.
remote: Compressing objects: 100% (71/71), done.
remote: Total 102 (delta 16), reused 65 (delta 13), pack-reused 0
Receiving objects: 100% (102/102), 23.35 KiB | 97.00 KiB/s, done.
Resolving deltas: 100% (16/16), done.
```

```
minchul@DESKTOP-9KULGUE MINGW64 /c
$ git clone git@github.com:kangtegong/collaboration.git /C/test
Cloning into 'C:/test/collaboration'...
remote: Enumerating objects: 106, done.
remote: Counting objects: 100% (106/106), done.
remote: Compressing objects: 100% (73/73), done.
remote: Total 106 (delta 17), reused 68 (delta 14), pack-reused 0
Receiving objects: 100% (106/106), 24.20 KiB | 6.05 MiB/s, done.
Resolving deltas: 100% (17/17), done.
```

원격 저장소와 상호 작용하기

■ git remote

- 로컬 저장소에 원격 저장소를 추가, 조회, 삭제할 수 있는 명령

- (예)

- git remote add <원격 저장소 이름><원격 저장소 경로>

- 로컬 저장소에 원격 저장소를 추가(연결)하는 명령

- (사용 예) git remote add origin <원격 저장소 경로>

- 원격 저장소를 origin이라는 이름으로 로컬 저장소에 추가

- 이렇게 원격 저장소를 추가하면 추후 origin이라는 이름으로 원격 저장소와 상호작용 가능

- git remote

- 추가된 원격 저장소 목록 조회

- git remote -v 또는 git remote --verbose 명령을 입력하면 원격 저장소의 이름과 경로까지 함께 확인할 수 있음

원격 저장소와 상호 작용하기

■ git remote

- 로컬 저장소에 원격 저장소를 추가, 조회, 삭제할 수 있는 명령
- (예)
 - `git remote rename <기존 원격 저장소 이름> <바꿀 원격 저장소 이름>`
 - 원격 저장소 이름 변경하기
 - (사용 예) `git remote rename origin changed` : 원격 저장소의 이름을 changed로 변경
 - `git remote remove <원격 저장소 이름>`
 - 추가한 원격 저장소를 삭제하기
 - (사용 예) `git remote remove changed` : changed 라는 이름의 원격 저장소 삭제하기

(실습)

■ remote 옵션 활용해보기

- 원격저장소인 GitHub에서 신규 repository 생성

- Repository 이름은 임의로 test-instructions
- Add a README file 체크 안함

- 로컬저장소에 test-instructions 폴더 생성 후 초기화

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions
$ git init
Initialized empty Git repository in C:/test-instructions/.git/
```

- 현재 결과

로컬저장소
test-
instructions



원격저장소
test-
instructions

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

ysseo29

Repository name *

test-instructions

✓ test-instructions is available.

Great repository names are short and memorable. Need inspiration? How about [psychic-eureka](#) ?

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

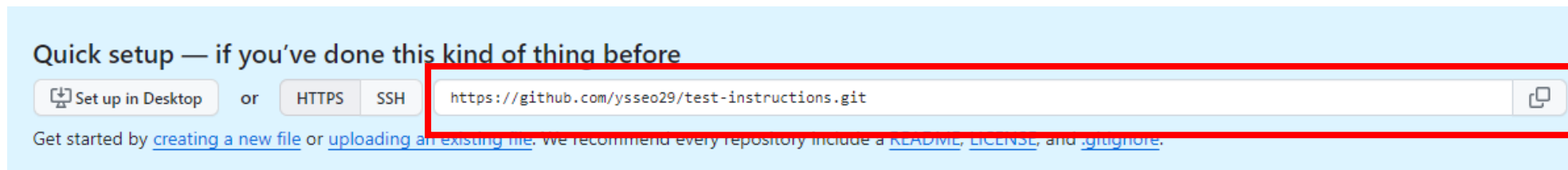
You are creating a public repository in your personal account.

Create repository

(실습)

■ remote 옵션 활용해보기

- 로컬저장소에 origin이라는 이름으로 원격저장소 추가(연결)

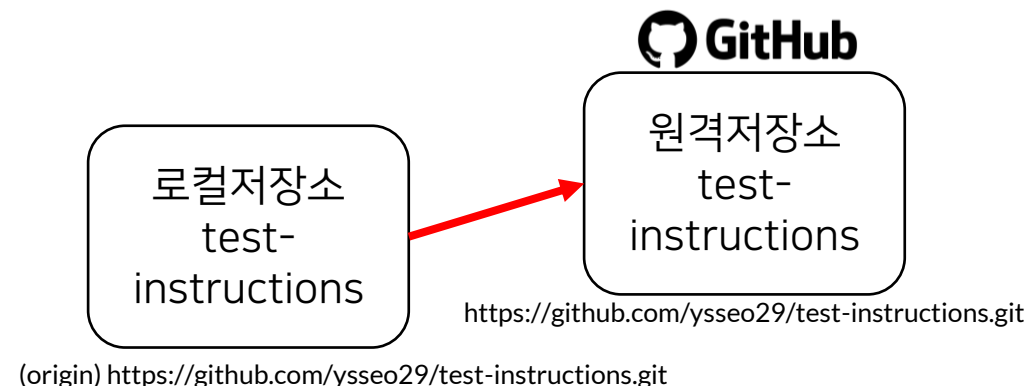


```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git remote add origin https://github.com/ysseo29/test-instructions.git
```

- 연결된 원격저장소 목록 조회

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git remote
origin

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git remote -v
origin https://github.com/ysseo29/test-instructions.git (fetch)
origin https://github.com/ysseo29/test-instructions.git (push)
```



(실습)

■ remote 옵션 활용해보기

- 원격 저장소 이름인 origin을 changed로 변경

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git remote rename origin changed

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git remote
changed

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git remote -v
changed https://github.com/ysseo29/test-instructions.git (fetch)
changed https://github.com/ysseo29/test-instructions.git (push)
```

- 추가한 원격 저장소 삭제

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git remote remove changed
```

원격 저장소와 상호 작용하기

- git push

- 로컬 저장소의 변경사항을 원격 저장소에 밀어넣는 명령어

(실습)

■ git push 활용해보기

● 버전 관리할 대상 만들기

- 로컬 저장소인 test-instructions에 문자 A가 적힌 a.txt 만들기

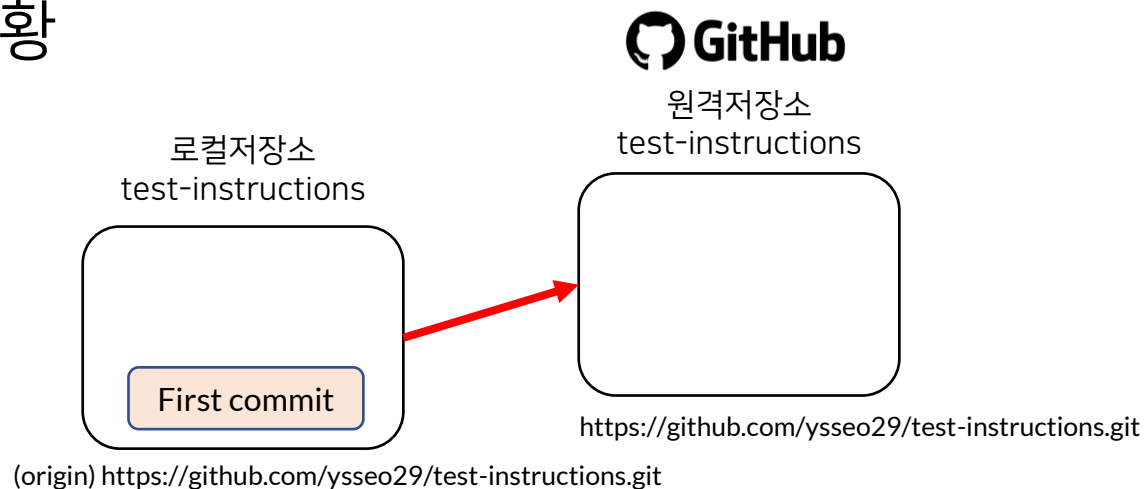
● 커밋하기

- 커밋메시지: first commit

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git add a.txt

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (master)
$ git commit -m "first commit"
[master (root-commit) e968825] first commit
1 file changed, 1 insertion(+)
create mode 100644 a.txt
```

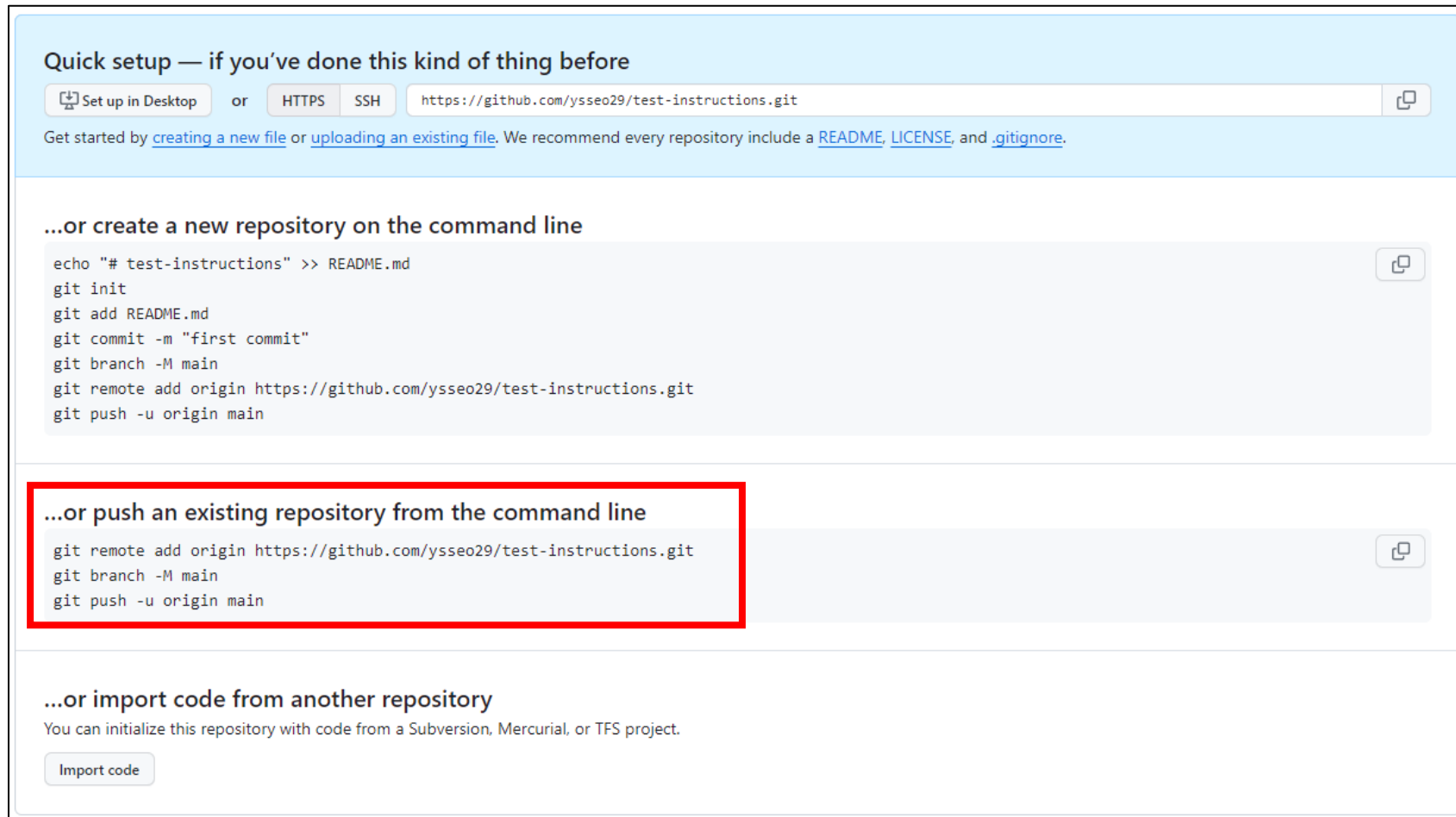
● 현재 상황



(실습)

■ git push 활용해보기

● push 하기



The screenshot shows the GitHub 'Quick setup' page. At the top, there's a section titled 'Quick setup — if you've done this kind of thing before' with a light blue background. It contains a 'Set up in Desktop' button, an 'or' separator, and 'HTTPS' and 'SSH' tabs. The 'HTTPS' tab is selected, showing the URL 'https://github.com/ysseo29/test-instructions.git'. Below this, it says 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).' Below this is a section titled '...or create a new repository on the command line' with a list of commands: `echo "# test-instructions" >> README.md`, `git init`, `git add README.md`, `git commit -m "first commit"`, `git branch -M main`, `git remote add origin https://github.com/ysseo29/test-instructions.git`, and `git push -u origin main`. Below that is a section titled '...or push an existing repository from the command line' which is highlighted with a red rectangle. It contains the commands: `git remote add origin https://github.com/ysseo29/test-instructions.git`, `git branch -M main`, and `git push -u origin main`. At the bottom is a section titled '...or import code from another repository' with the text 'You can initialize this repository with code from a Subversion, Mercurial, or TFS project.' and an 'Import code' button.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/ysseo29/test-instructions.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# test-instructions" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ysseo29/test-instructions.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/ysseo29/test-instructions.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

(실습)

■ git push 활용해보기

● push 하기

■ git branch -M main

- git branch -M <브랜치 이름> 으로 사용
- 현재 브랜치 이름을 <브랜치 이름>으로 바꾸는 명령
- 즉, 이 명령은 현재 브랜치(master) 이름을 main으로 변경하는 명령
- 로컬 저장소의 기본 브랜치는 master이지만 깃허브의 기본 브랜치는 main이기 때문에 로컬 저장소의 기본 브랜치(master)에서 만든 변경 사항을 깃허브의 기본 브랜치(main)로 푸시하기 위해서는 이와 같이 브랜치 이름을 main으로 변경
- 현재 버전에서는 앞서 수행한 명령어인 git remote add ~~~ 를 통해 자동 변경됨.

(실습)

■ git push 활용해보기

● push 하기

■ git push -u origin main

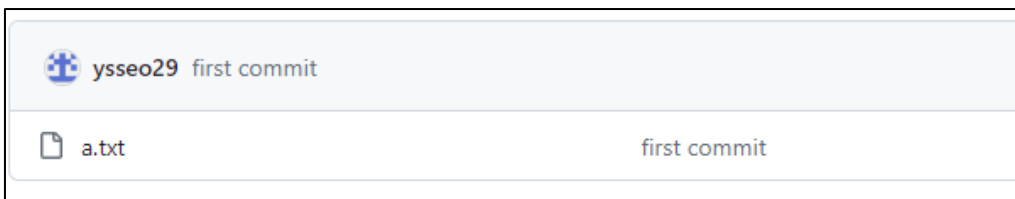
- git push <원격 저장소 이름> <브랜치 이름> 으로 사용
- <원격 저장소 이름>으로 <브랜치 이름> 을 푸시하는 명령
- (예) git push origin main은 원격 저장소 origin으로 로컬 저장소 main 브랜치의 변경 사항을 푸시하는 명령
- -u 옵션은 처음 푸시할 때 한 번만 사용하면 되는데, 이 옵션과 함께 푸시하면 추후 간단히 git push(또는 git pull) 명령만으로 origin의 main 브랜치로 푸시(또는 풀)할 수 있음

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (main)
$ git remote add origin https://github.com/ysseo29/test-instructions.git
error: remote origin already exists.

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (main)
$ git branch -M main

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 213 bytes | 213.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ysseo29/test-instructions.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

결과:



(실습)

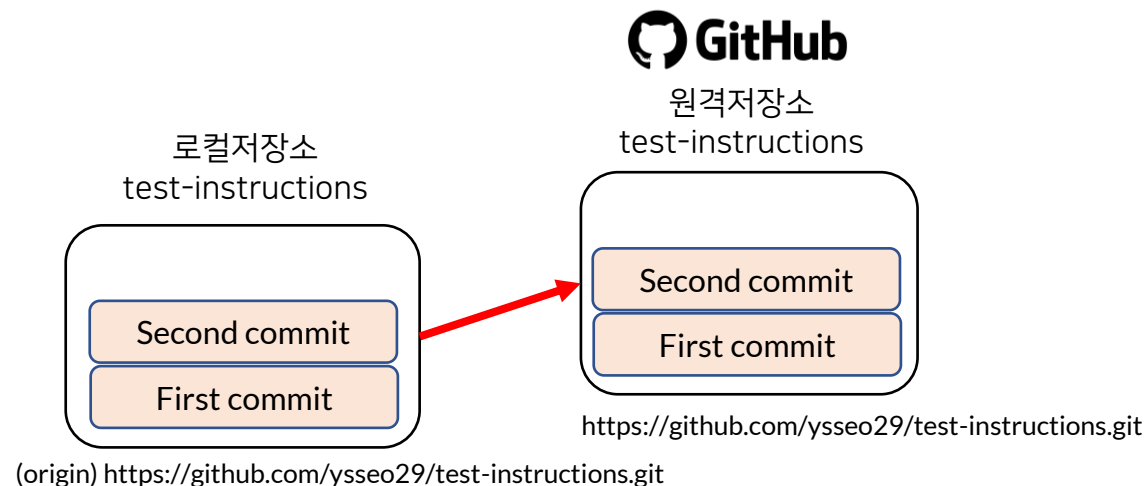
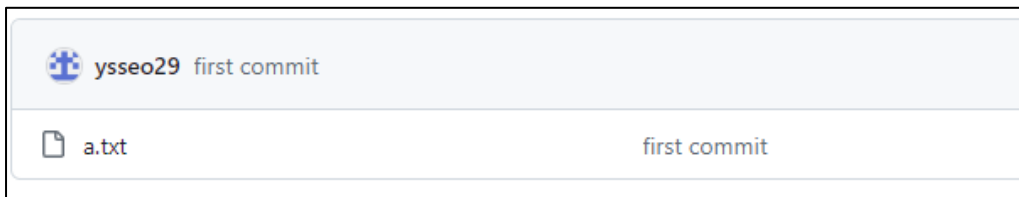
■ git push 활용해보기

- a.txt 파일에 문자 B를 추가한 이후 GitHub에 push 해보기

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (main)
$ git commit -am "second commit"
[main 22dc380] second commit
1 file changed, 1 insertion(+)

ysseo@DESKTOP-UVN9C3R MINGW64 /c/test-instructions (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 245 bytes | 245.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ysseo29/test-instructions.git
e968825..22dc380  main -> main
```

- 결과 확인



깃 명령으로 PR 보내기

■ PR을 통해 협업하는 방법

- (1) 기여하려는 저장소를 본인 계정으로 fork하기
- (2) fork한 저장소를 clone하기
- (3) branch 생성 후 생성한 branch에서 작업하기
- (4) 작업한 branch를 push하기
- (5) pull request 보내기

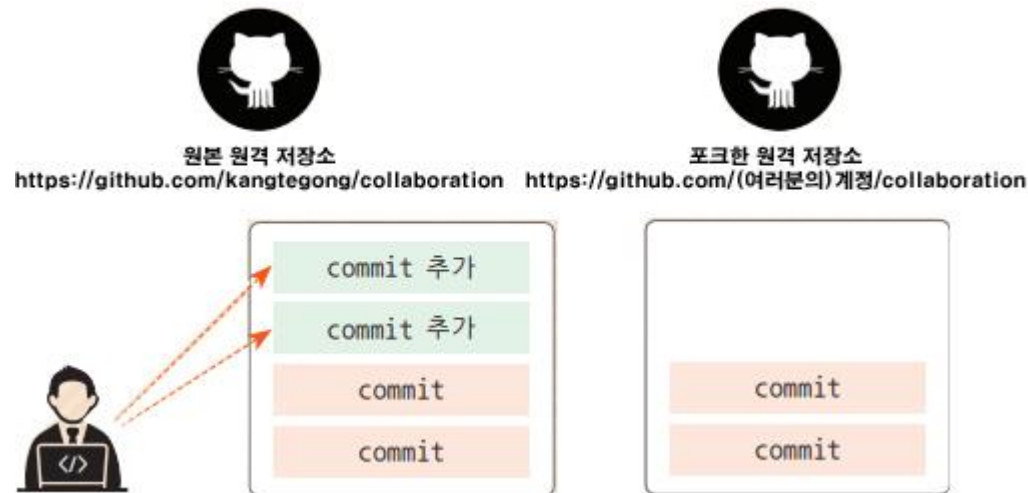
깃 명령으로 PR 보내기

- (1) 기여하려는 저장소를 본인 계정으로 fork하기
 - https://github.com/ysseo29/OSS_YU_test_repo

깃 명령으로 PR 보내기

■ (주의) fork 시 주의점

- 과거에 이미 특정 저장소를 fork 했을 경우, 현재 시점에서 해당 저장소의 업데이트가 많이 발생했을 수 있음 (즉, 원본 저장소에 비해 뒤쳐진 포크된 저장소)
- 다음 그림 속 박스처럼 This branch is X commits behind <저장소명칭> 메시지가 떠 있을 수 있음



깃 명령으로 PR 보내기

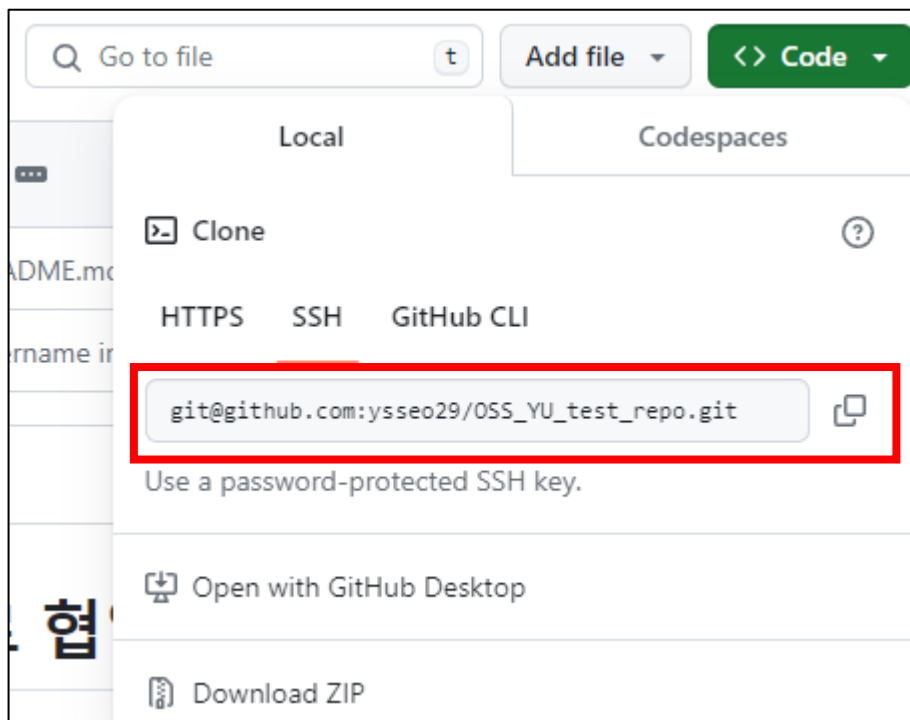
■ (주의) fork 시 주의점

- 이 경우 여러분의 계정으로 포크한 원격 저장소가 원본 저장소에 비해 뒤쳐지지 않도록 맞춰주어야 함
- “Sync fork”를 클릭 후 최신으로 맞추어 주기
 - upstream: 포크했던 원본 저장소를 의미함

깃 명령으로 PR 보내기

■ (2) fork한 저장소를 clone하기

- git clone <원격 저장소 주소>
 - ** 원본 저장소가 아닌 본인 계정으로 fork한 저장소를 클론
- 클론 받은 원격 저장소로 이동 (cd 명령어 이용)



```
ysseo@DESKTOP-UVN9C3R MINGW64 /c
$ git clone git@github.com:ysseo29/OSS_YU_test_repo.git
Cloning into 'OSS_YU_test_repo'...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 31 (delta 6), reused 3 (delta 1), pack-reused 0
Receiving objects: 100% (31/31), 14.83 KiB | 2.47 MiB/s, done.
Resolving deltas: 100% (6/6), done.
```

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c
$ cd OSS_YU_test_repo

ysseo@DESKTOP-UVN9C3R MINGW64 /c/OSS_YU_test_repo (main)
$ |
```

이름

- 📁 .git
- 🌐 index.html
- 📄 README.md

깃 명령으로 PR 보내기

■ (3) branch 생성 후 생성한 branch에서 작업하기

● branch 생성 후 해당 branch로 checkout

- `git branch <브랜치 이름>`, `git checkout <브랜치 이름>`
- `Git branch -b <브랜치 이름>` (브랜치 생성 및 checkout이 동시에 수행)

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/OSS_YU_test_repo (main)
$ git branch add_myname

ysseo@DESKTOP-UVN9C3R MINGW64 /c/OSS_YU_test_repo (main)
$ git checkout add_myname
Switched to branch 'add_myname'
```

● 실제 작업 진행

- index.html을 메모장이나 코드 편집기 등으로 열기
- `` 태그를 이용해 학번_영문이니셜 추가
- 커밋이후 push하기 전에 `git diff` 명령어로 변경 사항이 올바르게 만들어 졌는지 최종 확인
- index.html 커밋하기

- 커밋 메시지: add myname in index.html

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/OSS_YU_test_repo (add_myname)
$ git add index.html

ysseo@DESKTOP-UVN9C3R MINGW64 /c/OSS_YU_test_repo (add_myname)
$ git commit -m "add myname in index.html"
[add_myname 133819f] add myname in index.html
1 file changed, 2 insertions(+)
```

깃 명령으로 PR 보내기

■ (4) 작업한 branch를 push하기

- git push origin add_myname

- add_myname이라는 브랜치를 원격 저장소 origin에 푸시하겠다는 의미

```
ysseo@DESKTOP-UVN9C3R MINGW64 /c/OSS_YU_test_repo (add_myname)
$ git push origin add_myname
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 24 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 358 bytes | 358.00 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.
remote:
remote: Create a pull request for 'add_myname' on GitHub by visiting:
remote:   https://github.com/ysseo29/OSS_YU_test_repo/pull/new/add_myname
remote:
To github.com:ysseo29/OSS_YU_test_repo.git
 * [new branch]      add_myname -> add_myname
```

깃 명령으로 PR 보내기

■ (5) pull request 보내기

- fork한 원격 저장소로 돌아가보면 새롭게 활성화된 Compare & pull request 버튼 클릭



- 가운데 위치한 create pull request 클릭
- 원본 저장소 소유자는 받은 pull request를 병합해주거나 댓글을 달아줄 수 있음

추가 명령어

- git bash 상에서 명령어 매뉴얼 페이지 보기
 - git <명령어> --help : 웹브라우저에서 매뉴얼 페이지가 열림
 - git <명령어> -h : git bash 상에서 매뉴얼 정보 제공
- git 공식 사이트에서 매뉴얼 페이지 보기
 - git 홈페이지 - Documentation

