

# Git 교육자료 상편

2021.01 제작

# 목차

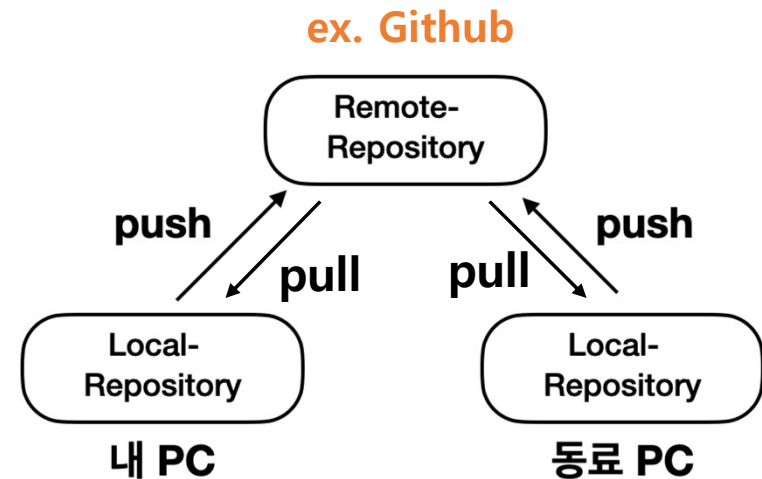
- Git과 Github의 개념
- Git 기초
  - git 구조
  - git init, git add, git commit,
  - git status
  - git checkout
  - git clone, git push
  - git remote, git fetch, git pull
  - .gitignore

# Git이란?

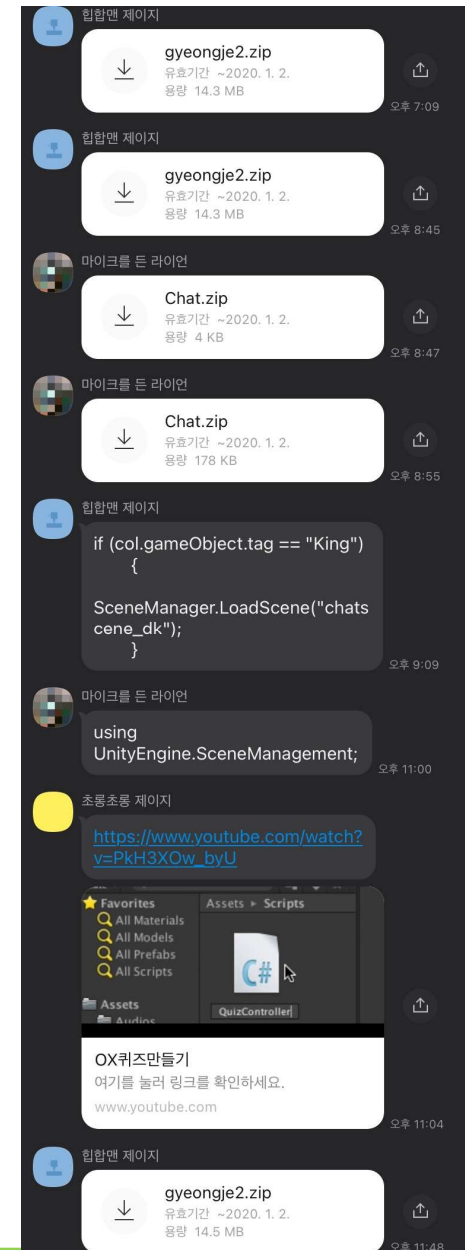
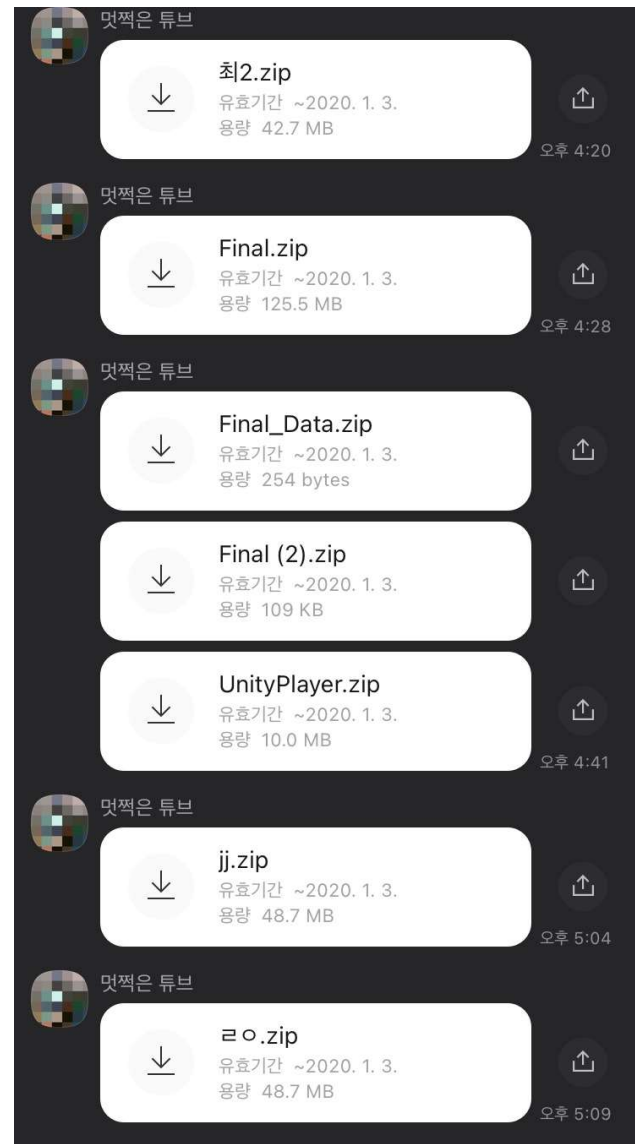
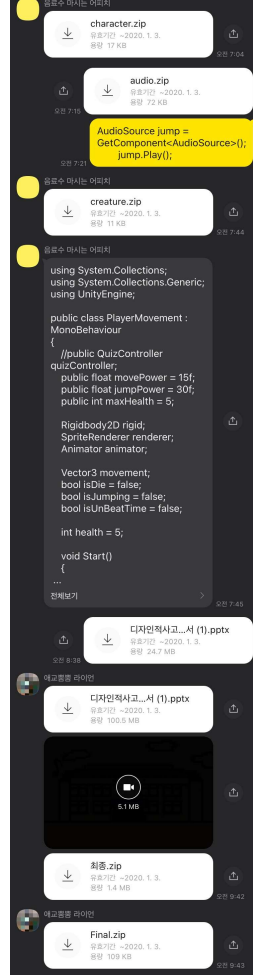
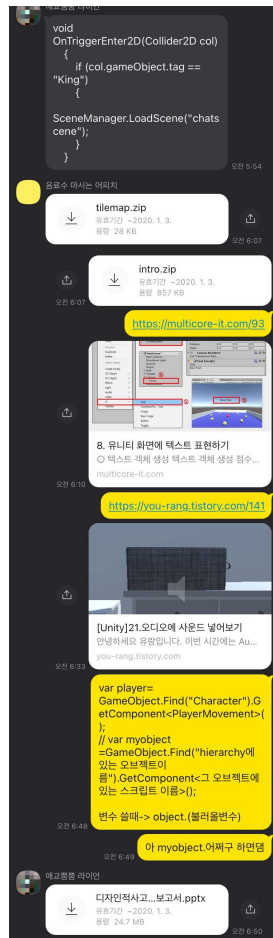
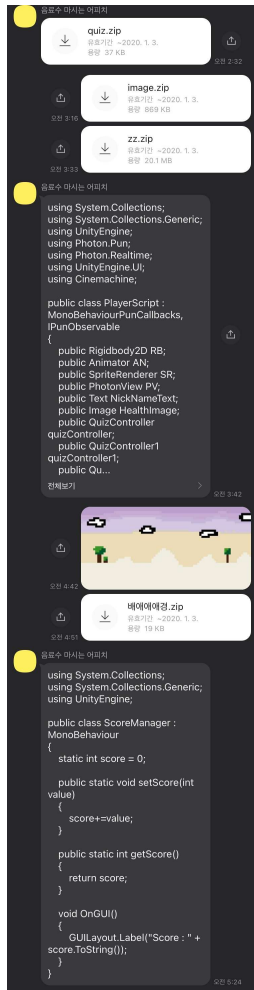
- 버전 관리 시스템(VCS, Version Control System)의 일종
- Git은 해당 local repository의 모든 파일의 변경 사항을 기록하여 특정 버전으로 만듦
  - 작업하고 있는 파일을, 과거의 원하는 시점으로 돌아갈 수 있게 하는 도구
- Git을 안 쓰면...?
  - 제3차\_설계변경-최종
  - 제3차\_설계변경-최종종
  - 제3차\_설계변경-최종종종
  - 제3차\_설계변경-정말최종
  - 제3차\_설계변경-정말최종종

# Github이란?

- 클라우드에 있는 Git 제공자
- 주로 협업 시 사용
- 내 컴퓨터에서 깃 히스토리를 가져와서 깃허브 웹사이트에 push  
→ 협업자가 깃허브에서 파일의 모든 기록을 볼 수 있음 (반대는 pull)

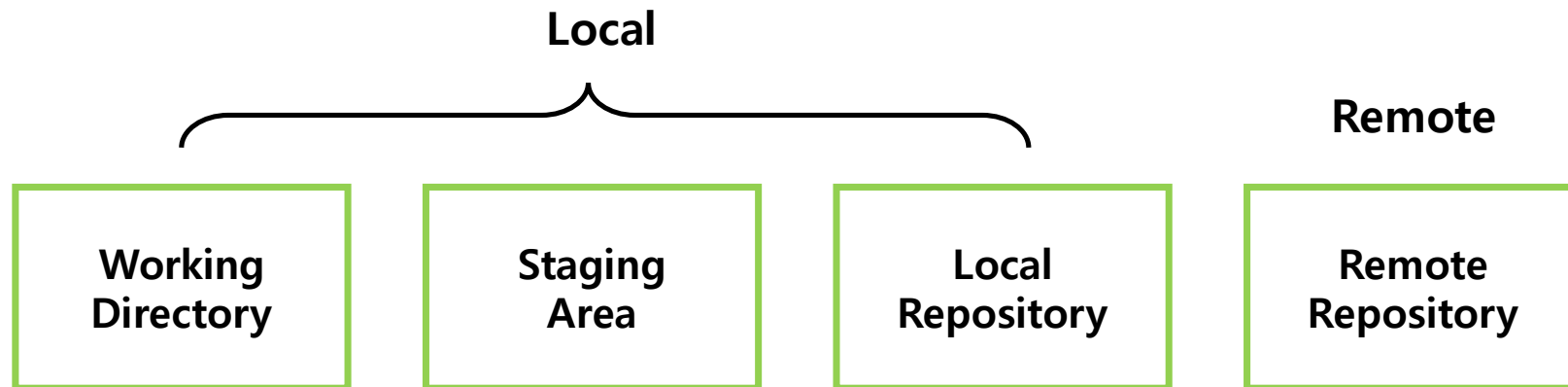


# Github가 왜 필요한가?



# Git의 구조

- **Repository**: 파일의 정보를 갖고 있는 저장소.  
실제 파일들보다 작은 용량을 가짐



# Git의 구조



- **Working Directory**

: 현재 작업공간. 파일을 생성, 수정, 삭제하는 것은 Working Directory에서 일어나는 일.

- **Staging Area**

: Working Directory에서 작업한 것을 Local Repository에 올리기 전. 어떤 파일을 올릴 것인지 선택함.

- **Local Repository**

: Git으로 관리되는 폴더가 저장되는 내 컴퓨터 상의 repository

- **Remote Repository**

: GitHub와 같이 외부에 있는 repository. 폴더가 원격으로 저장되는 공간. (github)

# Git 기초 – 사용자 정보 등록

- 사용자 이름과 이메일 주소를 설정해야 한다.
- **git config --global user.name "Hong Gil Dong"**
- **git config --global user.email "GilDong@khu.ac.kr"**

이후 github 가입 시, 이 이메일과 같은 계정으로 가입하는 게 권장되므로  
이를 참고하여 입력한다.

그리고 Github에서 학교 아이디를 쓰면 PRO 계정 혜택이 있으므로  
학교 계정을 쓰는 것이 좋다.

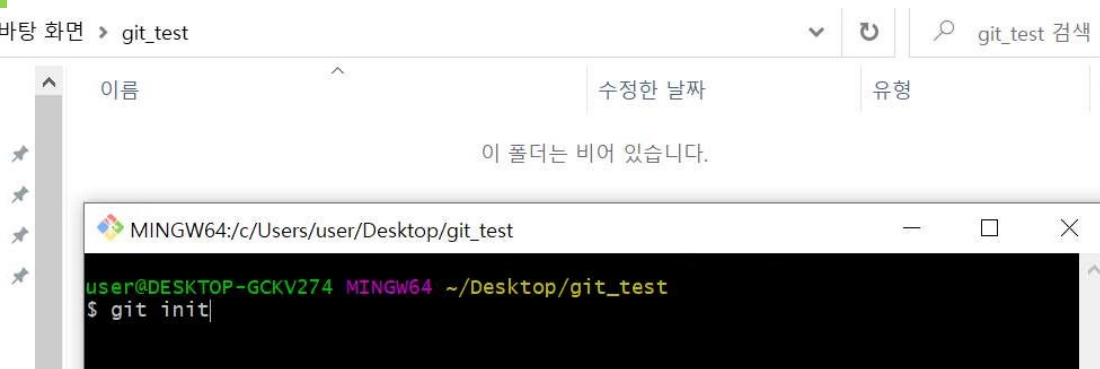
- **git config --list** : 설정한 정보 확인 가능



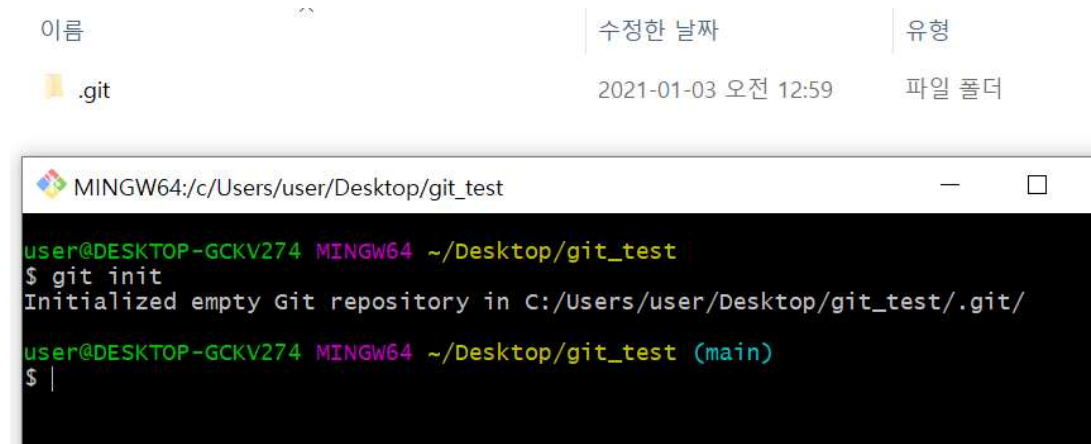
# Git 기초 – git init

- **git init**: 현재 폴더에서 git 시작
- **.git**(숨겨진 폴더)가 생성됨. 이 폴더에 들어갈 일은 없음.

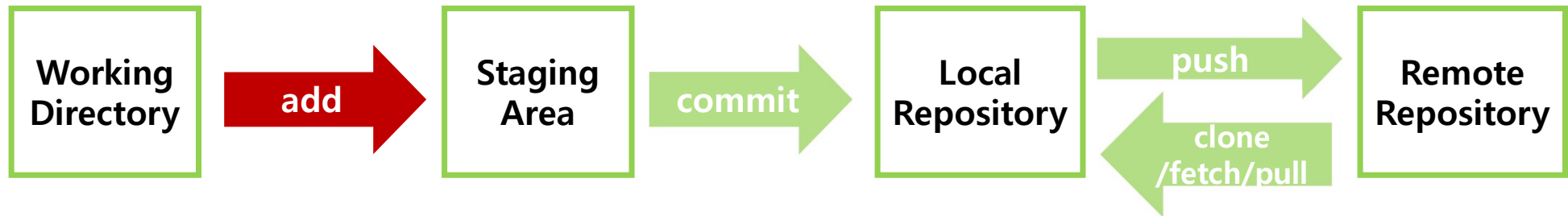
before



after

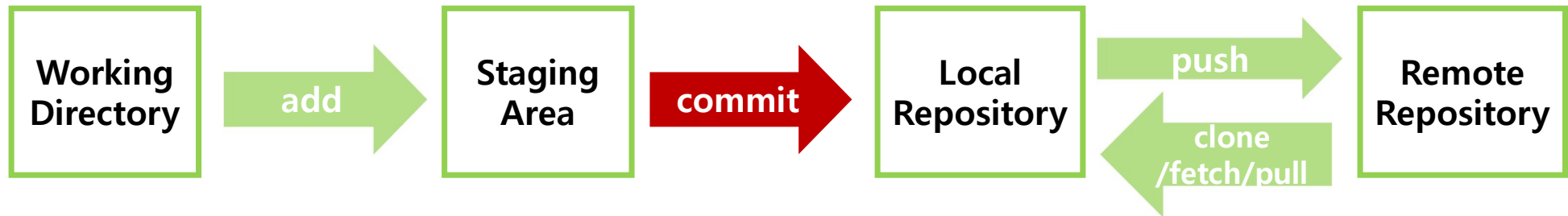


# Git 기초 – git add



- **git add 파일/폴더명**: git에 올릴 준비가 된 파일을 staging area에 올림
- **git add .** : 해당 디렉토리 내의 모든 파일을 git add함

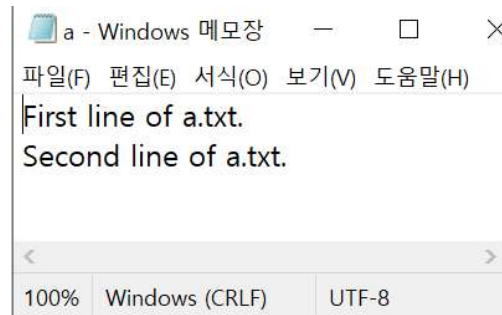
# Git 기초 – git commit



- 파일의 현재 상태를 저장
- **git commit -m "commit message"**
  - : staging area에 있던 파일들을 Local Repository에 올림.
- 이때 커밋 메시지는 보통 무엇을 했는지, 되도록 영어로 신경 써서 적어주는 것이 중요.

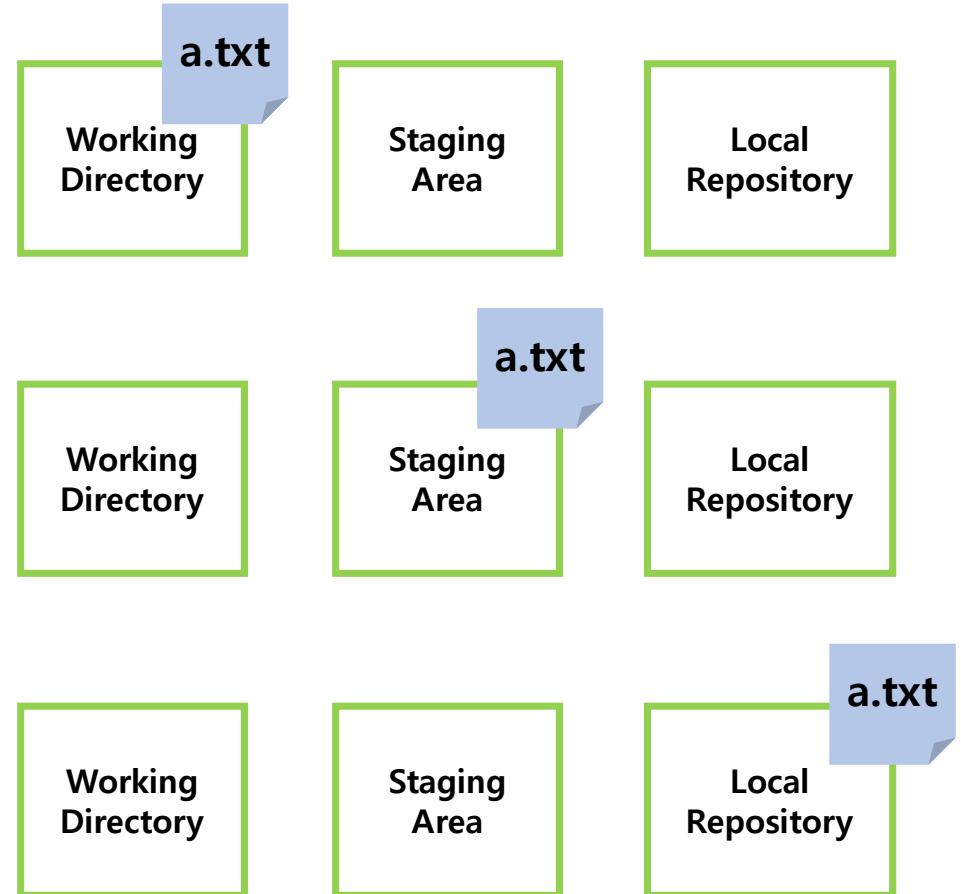
# Git 실습 1 – git add & git commit

- a.txt라는 새 텍스트 파일을 생성



- **git add a.txt**

- **git commit -m "Add a.txt"**



# Git 실습 1 – git add & git commit



A terminal window titled "MINGW64:/c/Users/user/Desktop/git\_test" with standard window controls. The terminal shows the following commands and output:

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test
$ git init
Initialized empty Git repository in C:/Users/user/Desktop/git_test/.git/

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git add a.txt

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git commit -m "Add a.txt"
[main (root-commit) 9ea3470] Add a.txt
1 file changed, 2 insertions(+)
create mode 100644 a.txt

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ |
```

# Git 기초 – git status

## **git status**

- working directory에서 파일의 생성, 수정, 삭제가 발생했지만, 이것이 staging area에 반영되지 않은 경우(git add를 안 한)를 확인
- 또는, staging area에 올라와 있지만, commit 되지 않은 경우를 알 수 있음.
- 습관적으로 해주는 것이 좋다.

# Git 실습 2 – git add & git commit

파일의 수정, 삭제도 같은 방식으로 진행

- 파일 수정

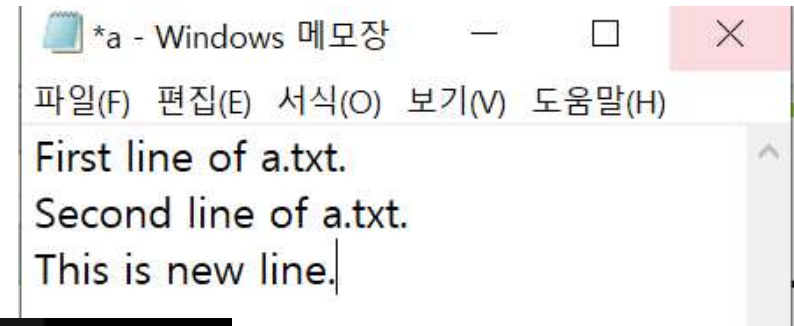
1. 파일(a.txt)을 수정한다.
2. **git status**
3. **git add <파일명>**
4. **git commit -m <commit message>**

# Git 실습 2 – git status

- a.txt를 수정해보자.
- **git status**

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ 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:   a.txt
no changes added to commit (use "git add" and/or "git commit -a")

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ |
```



- a.txt가 수정되었고, staging area에 반영되지 않았다고 알려준다.



# Git 실습 2 – git status

- git add a.txt
- git status
- git commit -m "Modified a.txt"

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git add a.txt

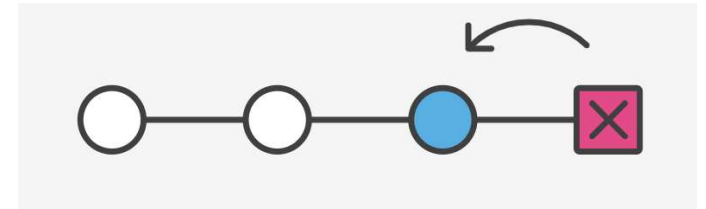
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   a.txt

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git commit -m "Modified a.txt"
[main 001b1ba] Modified a.txt
1 file changed, 2 insertions(+), 1 deletion(-)

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git status
On branch main
nothing to commit, working tree clean

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$
```

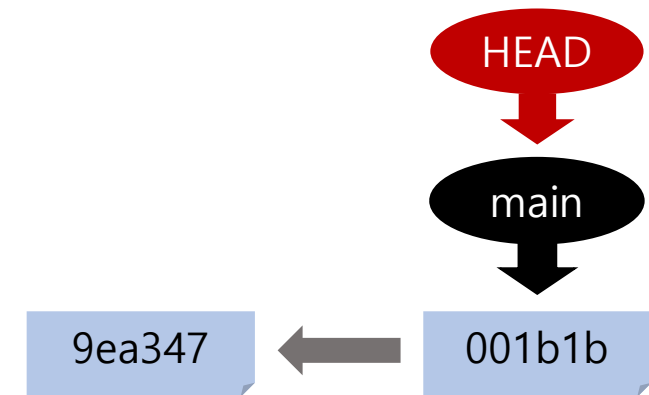
# Git 기초 – git checkout



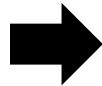
- git checkout은 HEAD 포인터를 원하는 곳으로 이동하는 것.  
이를 통해 이전에 커밋했던 내용으로 돌아갈 수 있다.
- **git checkout <Commit Hash>**: HEAD 포인터를 해당 commit 버전으로 이동.  
이때 Commit Hash는 **git log** 명령어를 통해 확인할 수 있다.
- **git checkout HEAD~n** : 현재 HEAD를 n 커밋 이전으로 이동
- **git checkout main**  
: main 브랜치의 가장 최근 커밋으로 HEAD를 이동시킨다.

# Git 실습 3 – git checkout

- **git log**
- checkout할 commit Hash 복사  
(앞 4~6자리만 복사해도 됨)



제일 최신 commit.



checkout할 버전



```
MINGW64:/c/Users/user/Desktop/git_test

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git log
commit 001b1ba787bc821b062526cac70d9e2466de1136 (HEAD -> main)
Author: 박민영 <pmy0792@khu.ac.kr>
Date: Mon Jan 4 01:31:31 2021 +0900

    Modified a.txt

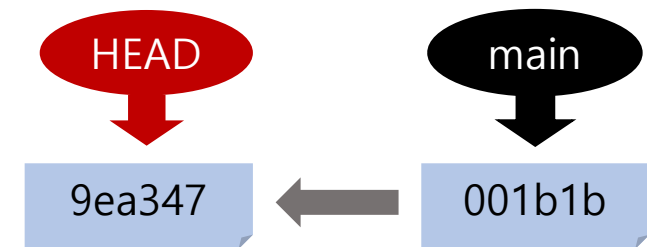
commit 9ea3470ed03a09b501f9c9c95a712998deec4ae1
Author: Park Min Yeong <pmy0792@naver.com>
Date: Sun Jan 3 01:04:01 2021 +0900

    Add a.txt

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ |
```

# Git 실습 3 – git checkout

- git checkout <복사한 commit Hash>
- git checkout HEAD~n  
: HEAD의 n단계 이전 버전을 checkout



```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git checkout 9ea347
Note: switching to '9ea347'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at 9ea3470 Add a.txt
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test ((9ea3470...))
$ |
```

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ git checkout HEAD~1
Note: switching to 'HEAD~1'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

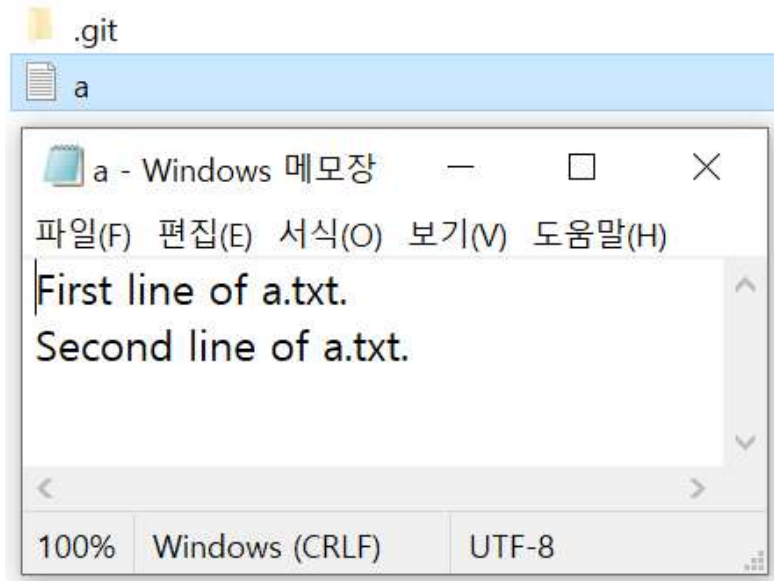
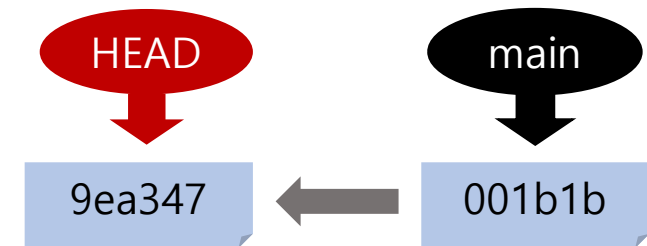
Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at 9ea3470 Add a.txt
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test ((9ea3470...))
$ |
```

# Git 실습 3 – git checkout

- **git checkout <복사한 commit Hash>**
- **git checkout HEAD~n**  
: HEAD의 n단계 이전 버전을 checkout

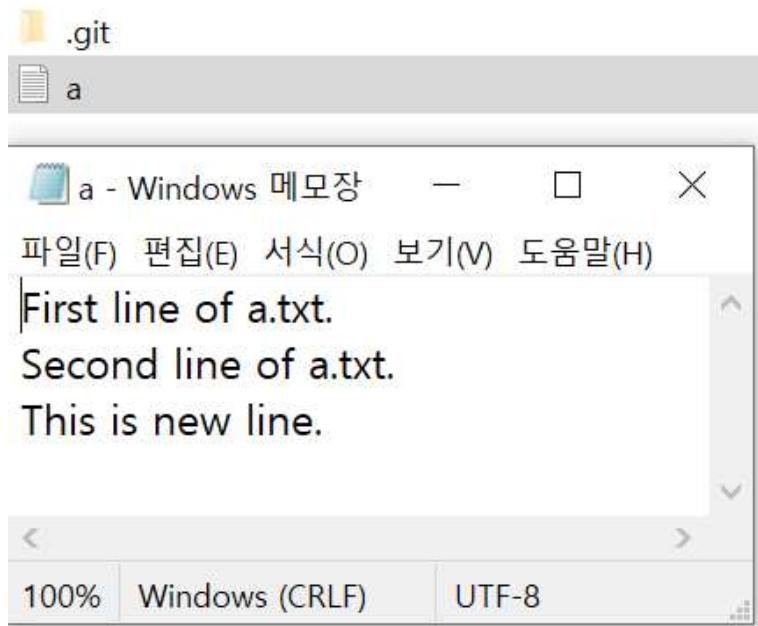
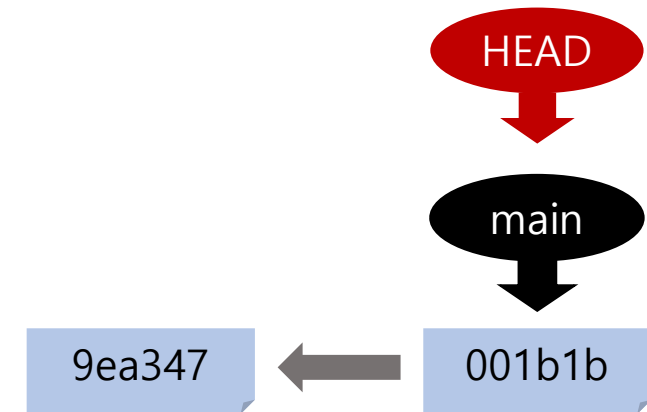


← 수정 전 a.txt를 확인 가능

# Git 실습 3 – git checkout

- **git checkout main**

: HEAD 포인터를 main의 가장 최근 커밋으로 이동. 다시 돌아감.



```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test ((9ea3470...))
$ git checkout main
Previous HEAD position was 9ea3470 Add a.txt
Switched to branch 'main'

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/git_test (main)
$ |
```

# Git 기초 – git push



## git push origin main

원격 저장소 이름    브랜치 이름

- origin이라는 remote repository에 main 브랜치의 커밋을 전송한다.

브랜치는 추후에 다룰 예정

# Git 기초 – git clone/pull/fetch의 차이



## clone

Local repository에 아무것도 없는 상태에서 Remote repository의 데이터를 가져옴  
(remote 설정을 자동으로 해주는 초기 다운로드에 사용)

## pull

Local repository에 이미 remote repository의 정보가 있고, remote repository의 변경 사항을 갱신하기 위해 하는 것

## fetch

Remote repository로부터 **변경 사항을 가져옴**  
but local repository에 변경 사항을 병합하지는 않음  
(pull은 병합함)



# Git 기초 – git clone



- GitHub와 같은 remote repository에서 그대로 가져와 시작 할 수 있다. git init 불필요.
- 초기 다운로드에 사용
- **git clone <원격 저장소의 SSH주소>**

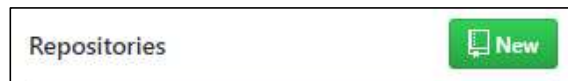
# Git 실습 4 – remote repository 생성

- 먼저 실습에 사용할 원격 레파지토리를 생성해보자.

(git clone, git remote 시 연결할 레파지토리)

- GitHub(<https://github.com>)에 가입.
- 가입 시, git에 처음 등록했던 이메일과 같은 아이디로 하는 것이 좋다.
- 그리고 학교 계정으로 가입하면 PRO 혜택이 있다는 이점이 있다.

- 로그인 후, 좌측의 new repository 클릭.



# Git 실습 4 – remote repository 생성

- Repository name을 입력하고 create repository 클릭

- 중간의 public vs private에 대해,

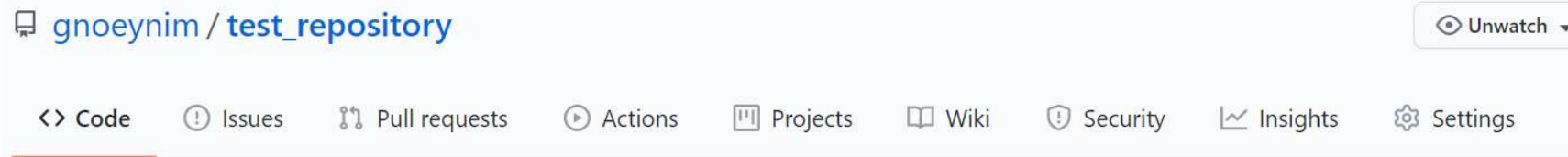
GitHub에서 생성되는 repository는 기본적으로 오픈소스이다.

이전에는, private는 다른 사람에게 소스코드를 공개하지 않지만 유료 이용자만 선택할 수 있었다.

하지만 2019년 1월자로, 무료로 private를 선택할 수 있게 되었다. 단, 협업은 3인까지만 가능하다.

# Git 실습 4 – remote repository 생성

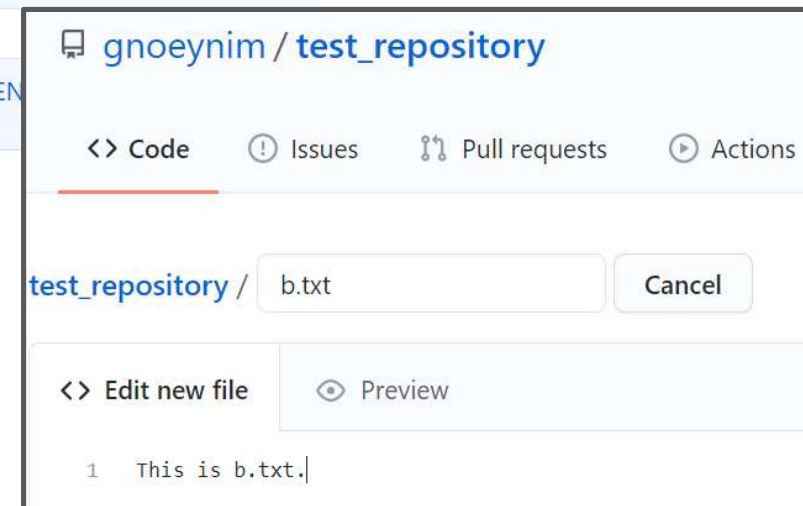
- 새로 생성한 레파지토리에 b.txt를 웹에서 추가한다.



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

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) [https://github.com/gnoeynim/test\\_repository.git](https://github.com/gnoeynim/test_repository.git)

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



# Git 실습 5 – git clone & git push

방금 만들었던 remote repository를 새 폴더에 **clone** 한 후,

README.md 파일을 만들어

그 변경사항을 **commit & push** 해보자.


# Git 실습 5 – git clone & git push

- 오른쪽의 Code 클릭 → SSH 클릭 → SSH 주소 복사

The screenshot shows the GitHub interface for the repository 'gnoeynim/test\_repository'. The 'Code' button is highlighted with a red circle. The 'Clone' dropdown menu is open, showing options for cloning the repository. The 'SSH' option is selected and highlighted with a red circle. The SSH URL 'git@github.com:gnoeynim/test\_repositor' is displayed and highlighted with a red circle. A red text annotation on the right side of the image states: '해당 repository의 SSH주소가 복사됨.' (The SSH address of the repository is copied).

# Git 실습 5 – git clone & git push

**git clone <복사한 SSH주소>**

 MINGW64:/c/Users/user/Desktop/test\_folder

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder
$ git clone git@github.com:gnoeynim/test_repository.git
Cloning into 'test_repository'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

※ 처음 git clone을 하면 인증 오류가 뜨기 때문에 인증을 해줘야 한다.(다음 슬라이드)

# Git 실습 5 – git clone & git push

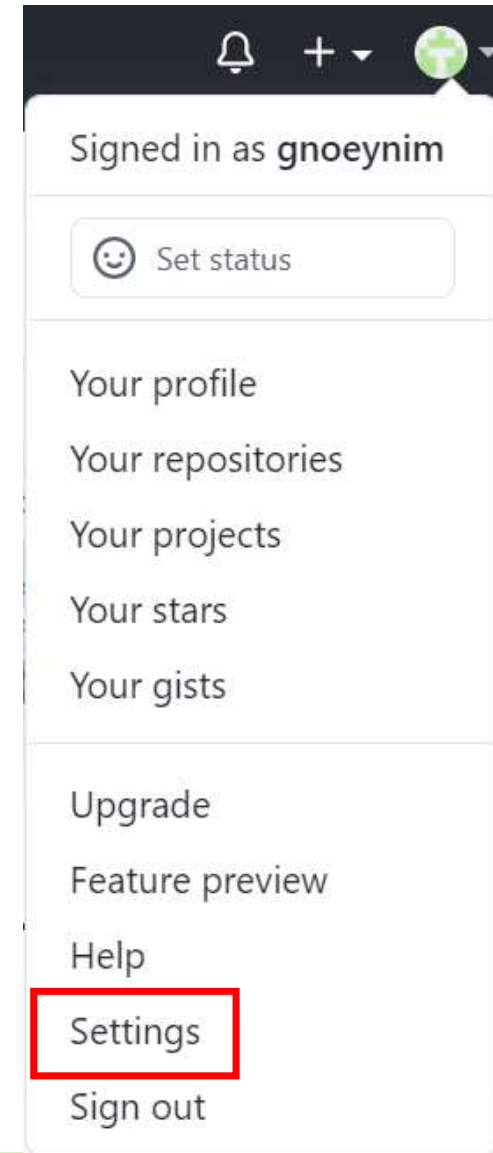
- 먼저 인증을 위한 ssh key를 생성한다.
- **ssh-keygen -t rsa -C "[your\\_email@example.com](mailto:your_email@example.com)"**
- 계속 엔터를 누른다.
- **cat ~/.ssh/id\_rsa.pub**
- 입력 후 나온 출력을 모두 복사한다.

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/webscraper
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQCsyFv1gyX2TZq38B+5w
/GBRxpGjEuJ/HCA18OubFD4ST5he3EKZRhAq0dIrJcIZqqRXzwIwXmqx7nvJ
T83soTUT9iMvmabI142UP9T9sWob0bk7eamcm7v26pzX0ioKtaf6eAvCRd7L
I3arbav9uLuPDs1tJAJuIEvM79fQTMx2ek0Ve+mx+gRKAL6tTz1qiudSyOkp
E3fahizVoUzGejtQNAaPjnxWvpXMhNGsNYGzJZZEXJfKKSRowZeH1UbnCpcg
```



# Git 실습 5 – git clone & git push

- GitHub의 우측 상단 Settings 클릭



# Git 실습 5 – git clone & git push

- 좌측 SSH and GPG keys 클릭
- New SSH key 클릭 후 복사했던 것 붙여넣기

SSH keys / Add new

Title

Key

begins with 'ssh-rsa', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

여기에 복사한 것 붙여넣기

Add SSH key

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

Sessions

Blocked users

Repositories

Organizations

Saved replies

Applications

# Git 실습 5 – git clone & git push

- 다시 SSH주소를 복사한 다음 **git clone <SSH 주소>**

MINGW64:/c/Users/user/Desktop/test\_folder

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder
$ git clone git@github.com:gnoeynim/test_repository.git
Cloning into 'test_repository'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

> 내 PC > 바탕 화면 > test\_folder

이름

test\_repository



# Git 실습 5 – git clone & git push

- `cd <git clone을 통해 생성된 폴더>`: 현재 위치를 이동한다.

```
MINGW64:/c/Users/user/Desktop/test_folder/test_repository

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder
$ git clone git@github.com:gnoeynim/test_repository.git
Cloning into 'test_repository'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder
$ cd test_repository

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder/test_repository (main)
$ |
```

# Git 실습 5 – git clone & git push

- **touch README.md** : README.md 파일 생성
- README.md 파일 수정
- **git add README.md**
- **git commit -m "Add README.md"**

※ .md 파일은 마크다운 파일로, jupyter notebook에서처럼 꾸밀 수 있다.

# Git 실습 5 – git clone & git push

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder/test_repository (main)
$ touch README.md

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder/test_repository (main)
$ git add README.md

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder/test_repository (main)
$ git commit -m "Add README.md"
[main 0b9ddb3] Add README.md
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder/test_repository (main)
$ |
```

# Git 실습 5 – git clone & git push

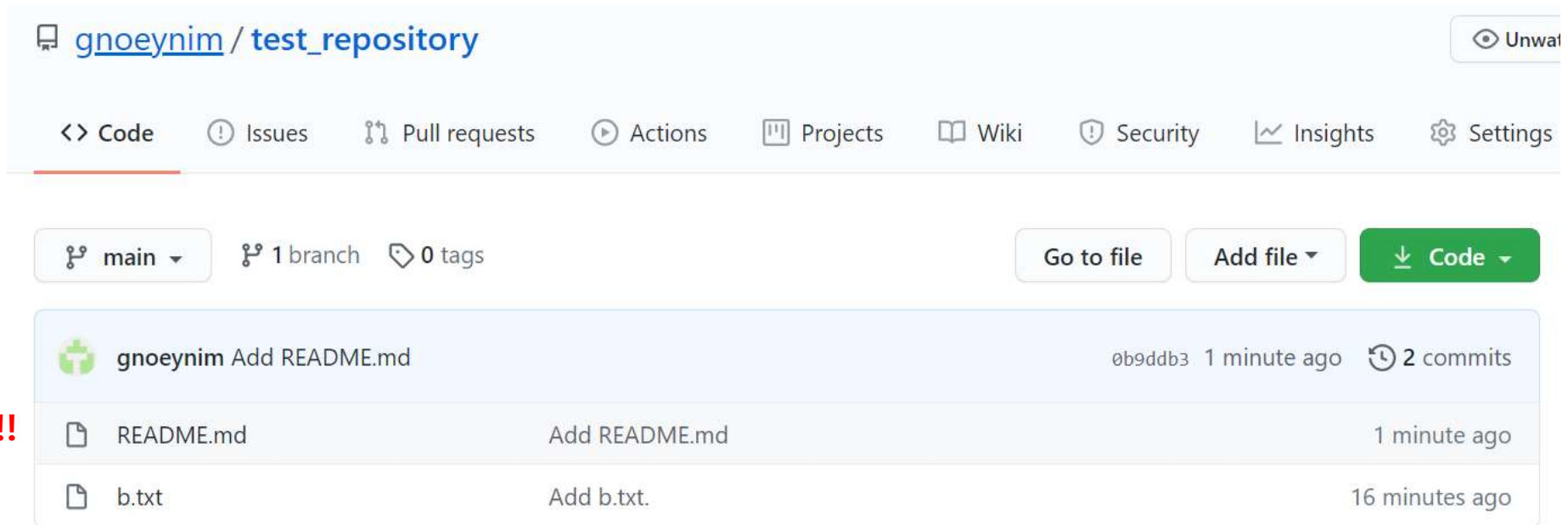
- git push origin main

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder/test_repository (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:gnoeynim/test_repository.git
   d321d42..0b9ddb3  main -> main

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder/test_repository (main)
$ |
```

# Git 실습 5 – git clone & git push

- GitHub가서 확인



gnoeynim / test\_repository

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

gnoeynim Add README.md		0b9ddb3 1 minute ago 2 commits
README.md	Add README.md	1 minute ago
b.txt	Add b.txt.	16 minutes ago

New!!

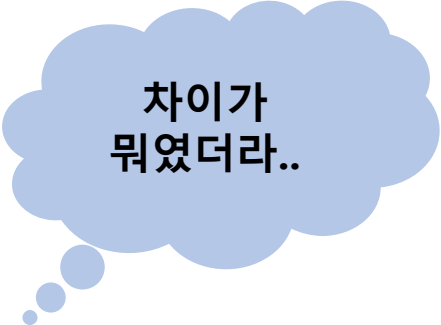


# Git 기초 – git remote add

- git clone에서는 git init을 따로 하지 않았음을 알 수 있었다.
- 그렇다면 이미 local에서 작업하고 있었다면, remote와 연결할 수 없는가? => X
- **git remote add origin <SSH 주소>**  
: origin이라는 remote repository에 연결한다.

# Git 기초 – git fetch & git pull

- git clone은 remote repository를 통째로 가져왔다.
- 이미 작업하던 local repository에서 remote repository를 가져오고 싶다면 => **git fetch, git pull**



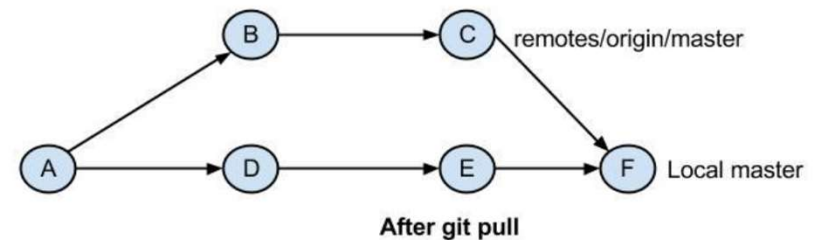
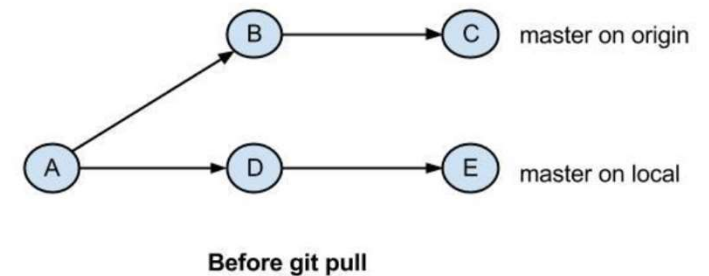
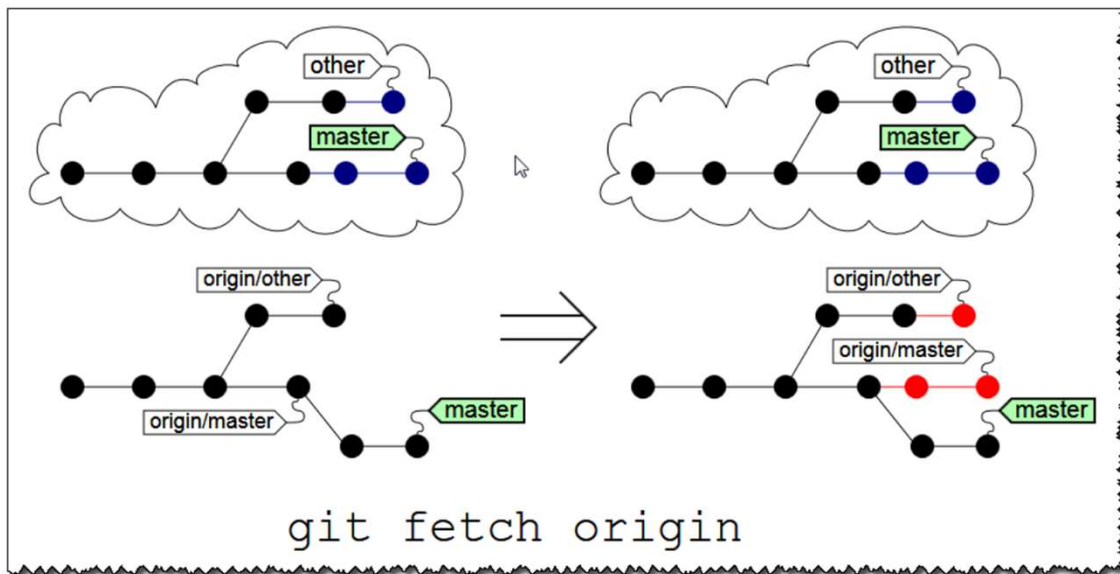
차이가  
뭘였더라..

# Git 기초 – git fetch & git pull

- **git fetch origin <branch>**  
: origin에서 해당 브랜치의 데이터를 가져온다.
- **git pull origin <branch>**  
: origin에서 해당 브랜치의 데이터를 가져온 후,  
local과 merge시킨다. Merge는 다음 강의자료에서 설명.
- 주로 git pull을 사용

# Git 기초 – git fetch & git pull

- git fetch vs git pull



# Git 기초 – git clone/pull/fetch 정리

✓ **git clone <주소>**

✓ **git init + git remote add origin <주소> + git pull origin main**

## clone

Local repository에  
아무것도 없는 상태에서  
Remote repository의  
데이터를 가져옴  
(remote 설정을 자동으로  
해주는 초기 다운로드에 사용)

## pull

Local repository에 이미  
remote repository의 정보가  
있고, remote repository의  
변경 사항을 갱신하기 위해  
하는 것

## fetch

Remote repository로부터  
**변경 사항을 가져옴**  
but local repository에 변경  
사항을 병합하지는 않음  
(pull은 병합함)

# Git 실습 6

컴퓨터에 새 폴더를 만들어 기존의 **remote repository**와 **연결**하고,  
README.md를 **삭제**한 후 그 변경사항을 **commit & push** 해보자.

## Git 실습 6 – ① git remote & git pull

- **git init**
- GitHub에서 이전에 생성했던 repository 주소 복사
- **git remote add origin <복사했던 주소>**
- **git pull origin main**

# Git 실습 6 – ② git remote & git pull

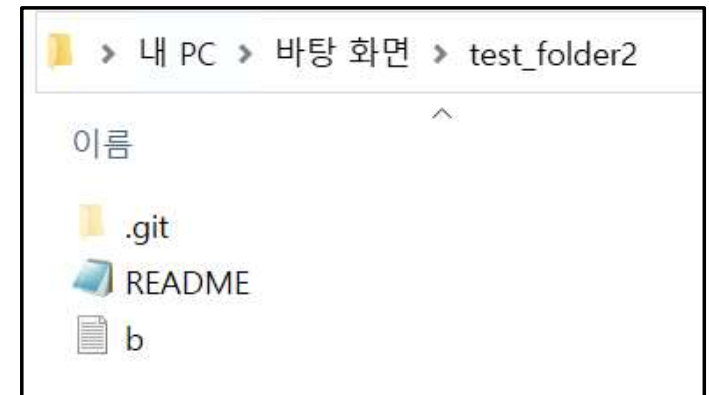
MINGW64:/c/Users/user/Desktop/test\_folder2

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder2
$ git init
Initialized empty Git repository in C:/Users/user/Desktop/test_folder2/.git/

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder2 (main)
$ git remote add origin git@github.com:gnoeynim/test_repository.git

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder2 (main)
$ git pull origin main
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
Unpacking objects: 100% (6/6), 841 bytes | 44.00 KiB/s, done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0
From github.com:gnoeynim/test_repository
* branch          main       -> FETCH_HEAD
* [new branch]     main       -> origin/main

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder2 (main)
$ |
```





# Git 실습 6 – ③ git add & git commit & git push

Git에 있는 파일을 삭제

- ① 파일을 삭제한다
- ② **git add <파일명>**
- ③ **git commit -m <commit message>**

- ① **git rm --cached <파일명>**  
: 실제로는 삭제되지 않았지만 삭제되었다고 함.
- ② **git commit -m <commit message>**

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder2 (main)
$ git rm --cached README.md
rm 'README.md'

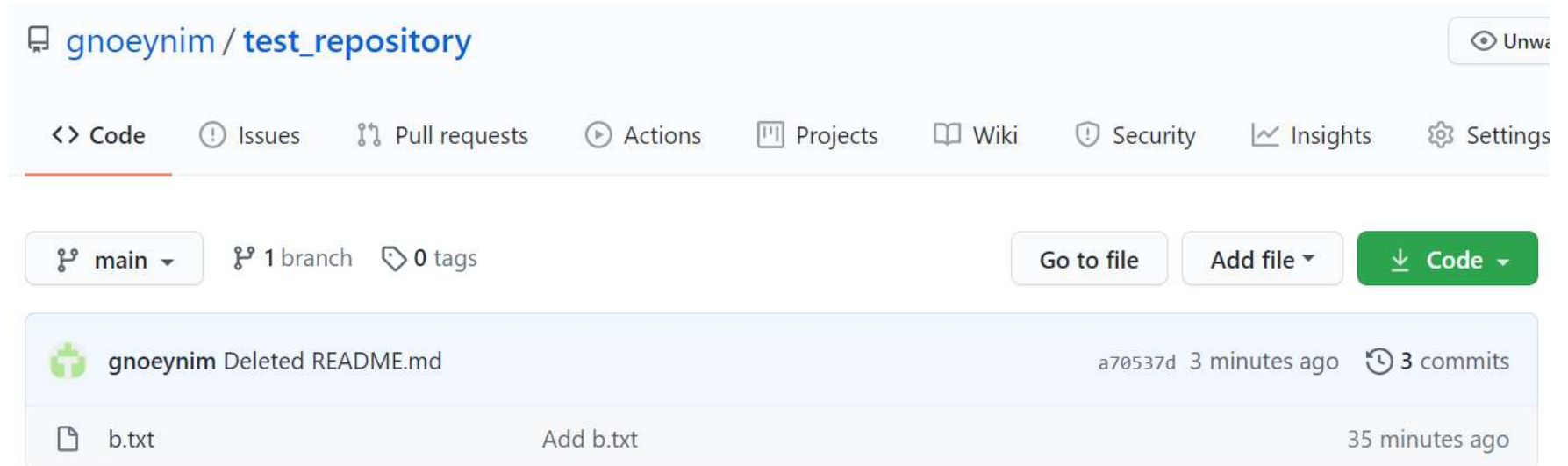
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder2 (main)
$ git commit -m "Deleted README.md"
[main a70537d] Deleted README.md
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 README.md

user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder2 (main)
$ |
```

# Git 실습 6 – ③ git add & git commit & git push

git push 후  
README.md가 삭제된 것을  
github에서 확인

```
user@DESKTOP-GCKV274 MINGW64 ~/Desktop/test_folder2 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (2/2), 247 bytes | 247.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
```



The screenshot shows the GitHub interface for the repository 'gnoeynim / test\_repository'. The repository has 1 branch (main) and 0 tags. The commit history shows a commit by 'gnoeynim' titled 'Deleted README.md' made 3 minutes ago, with 3 commits in total. Below the commit history, there is a file 'b.txt' with an 'Add b.txt' button, dated 35 minutes ago. The navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

# Git 기초 - .gitignore

- Why?

버전 관리가 필요하지 않은 파일도 있으므로, 모든 파일을 커밋하는 것은 불필요하다.

빌드 파일과 같이, 크게 중요하지 않고 재생성 가능한 파일들은 commit할 때 제외하는 것이 편하다.

- 최상위 directory에 .gitignore 파일을 만든 후, 커밋에서 제외할 파일을 .gitignore 파일에 적는다.
- [.gitignore template](#) [참고](#)

# Git 기초 - .gitignore

- **touch .gitignore** : .gitignore 생성.
- .gitignore을 열고, 제외하고자 하는 파일명 또는 폴더이름, 또는 파일 확장자(\*.exe와 같은 형식) 작성.

상편 끝.  
하편에서 계속...