

Git & Github 사용법

들어가기 전 필수 용어설명

- 디렉토리(Directory) : 우리가 평소에 사용하는 폴더(Folder)입니다. 폴더를 디렉토리라고 부를뿐!!
- 커맨드(command) : git bash에서 사용하는 명령문입니다.
- 워킹 디렉토리(working directory) : 작업을 하는 project directory
- 스테이징 아레아(staging area) : git add를 한 파일들이 존재하는 영역
- 레포지토리(repository) : 워킹 디렉토리의 변경 이력이 저장되어 있는 영역
- 로컬 레포지토리(local repository) : 나의 컴퓨터에 만들어 놓은 git의 commi을 관리하는 숨겨진 디렉토리라고 생각하시면 됩니다.
- 리모트 레포지토리(remote repository) : Github에 만들어 놓은 원격 commit 저장소입니다.

Git사용법



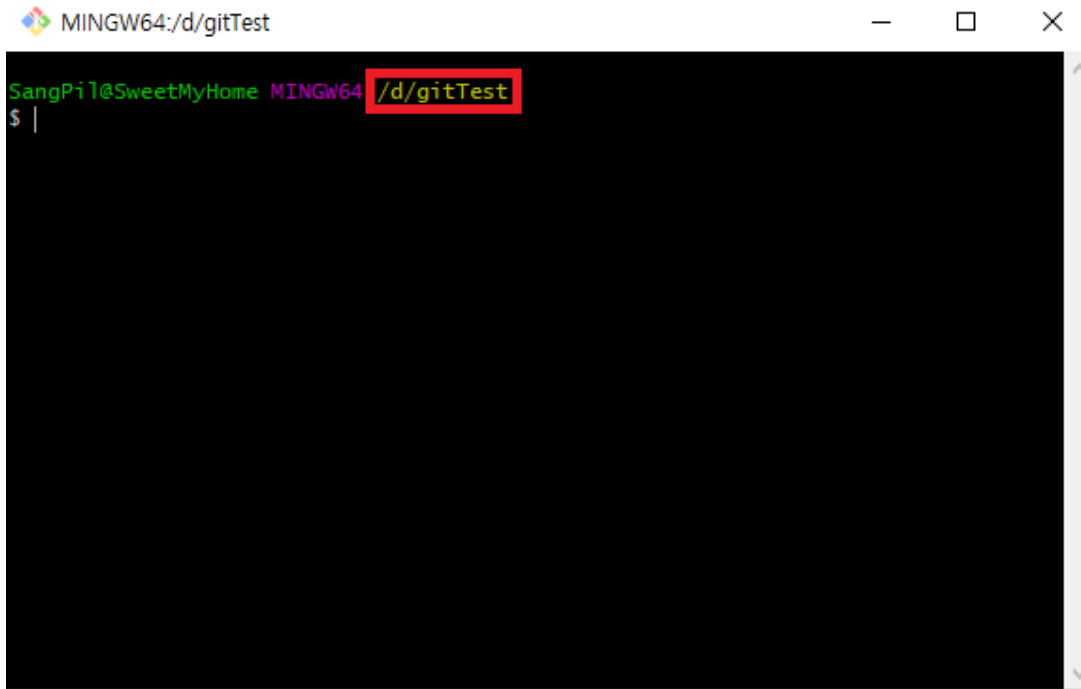
- git 다운로드하기(<https://git-scm.com/>)
- ‘Download 2.29.2 for Windows 버튼’을 누르면 잠시 후 자동으로 다운로드를 시작합니다.
- 다운로드 된 Git-2.29.2.2-64-bit.exe파일을 실행시키고 별다른 설정 건드리실 거 없이 체크되어 있는 그대로 끝까지 설치해주세요.

Git사용법



- git 설치 완료 후 새로운 디렉토리에서 우클릭 해보면 좌측 화면처럼 git과 관련된 항목이 2개 생긴 걸 볼 수 있습니다.
- 저는 D드라이브 - gitTest라는 워킹 디렉토리를 만들고 그 안에서 우클릭을 한 상태입니다.(경로명 참고하세요.)
- gitTest라는 워킹 디렉토리에서 git기능을 사용하기 위해 작업을 해주어야 하는데 해당 작업은 Git Bash(Mac의 경우 터미널)에서 가능합니다.
- Git Bash Here을 클릭해줍니다.

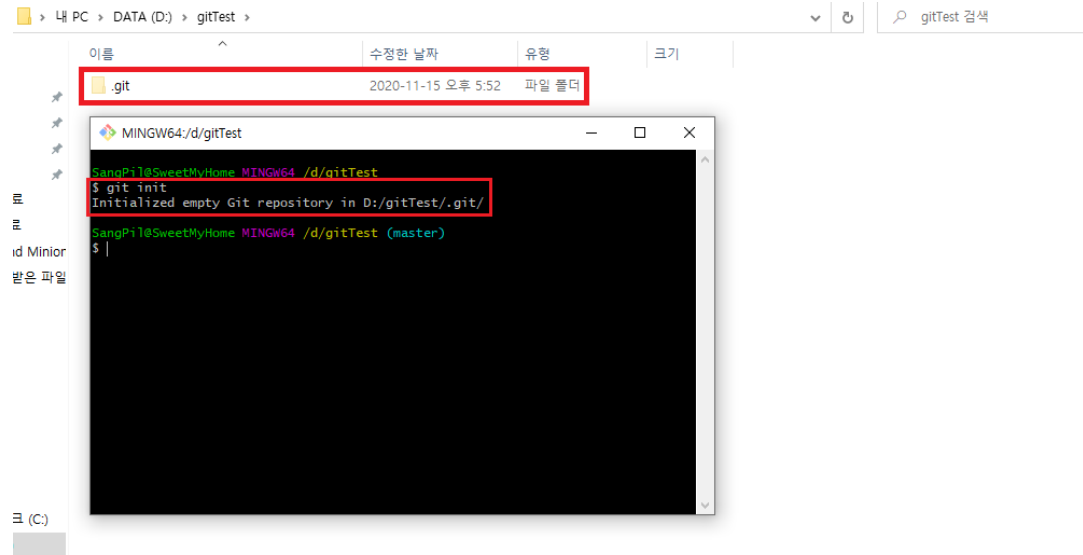
Git사용법



A screenshot of a Windows command prompt window. The title bar at the top reads 'MINGW64:/d/gitTest'. The command prompt shows the user 'SangPil@SweetMyHome' in a 'MINGW64' environment, with the current directory highlighted as '/d/gitTest' in a red box. The prompt is '\$ |'.

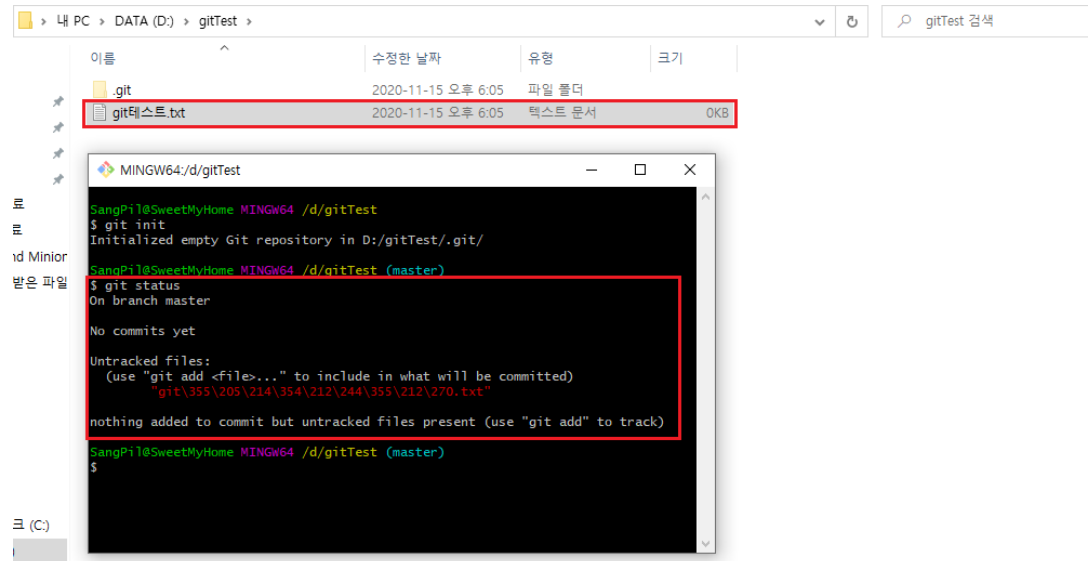
- git bash를 실행시키면 자동으로 /d/gitTest처럼 내가 git bash here한 디렉토리의 경로가 잡혀 있습니다 (Mac은 이 기능이 없어서 찾아서 들어 가셔야 합니다.)
- 경로가 다를 경우 cd 커맨드를 사용해서 디렉토리 이동을 하게 되는데 커맨드는 git bash 파일에 정리해 두었으니 참고하시기 바랍니다.
- 이 상태에서 git bash창에 차근차근 커맨드를 입력하여 해당 디렉토리에서 git을 사용해 보겠습니다.

Git사용법



- git init 이라고 입력하시면 Initialized empty Git repository in D:/gitTest/.git/ 이라는 문구가 출력되고 워킹 디렉토리에 .git이라는 숨겨진 디렉토리가 추가되는 걸 확인할 수 있습니다. 지금 생성된 이 .git 디렉토리가 gitTest의 로컬 레포지토리입니다.
- 이렇게 git init을 가장 먼저 해주어야 해당 워킹 디렉토리에서 git 기능을 사용 및 github와 연동할 수 있습니다.

Git사용법

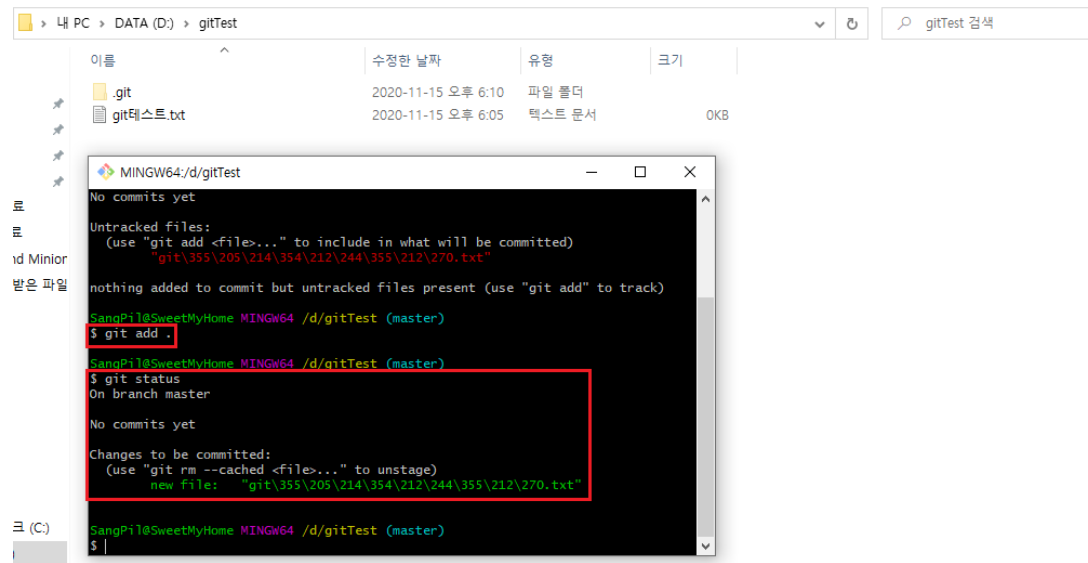


• 비어있던 gitTest 디렉토리에 git테스트.txt라는 빈 텍스트 파일을 추가하였습니다. 그리고 이 상태에서 git bash 창에 git status를 입력해보면 Untracked files가 있다고 나옵니다.

• git status는 git이 현재 해당 워킹 디렉토리를 어떻게 인식하고 있는지 알고 싶을 때 사용합니다.(수정사항이 있는지 확인할 때 보통 사용합니다.)

• Untracked files는 'Git에 의해 아직 추적되지 않고 있다 -> 버전 관리의 대상이 아니다.' 라는 의미이고 이 파일을 버전 관리하기 위해서는 staging area에 추가해줘야 합니다. 이 때 사용하는 커맨드가 git add 입니다.

Git사용법

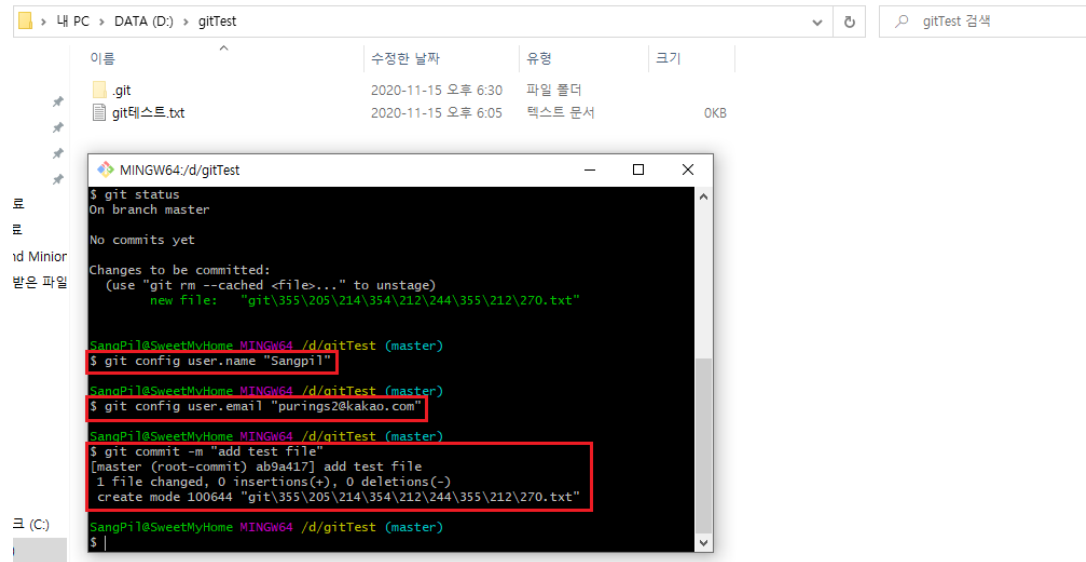


The image shows a Windows File Explorer window and a terminal window. The File Explorer window displays the contents of the 'gitTest' directory, which includes a '.git' folder and a 'git테스트.txt' file. The terminal window shows the output of the 'git add .' command, indicating that the file has been staged for commit.

```
MINGW64/d/gitTest
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  git\355\205\214\354\212\244\355\212\270.txt
nothing added to commit but untracked files present (use "git add" to track)
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git add .
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new file:   git\355\205\214\354\212\244\355\212\270.txt
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$
```

- git add [파일명] 으로 파일 하나씩 add 할 수 있고, 혹은 좌측에 보이는 것처럼 git add . 으로 모든 파일을 한 번에 add 할 수 있습니다.
- add 이후 git status를 해보시면 초록색으로 변경된 것이 보일텐데 이 상태가 되면 정상적으로 staging area에 변경사항이 add가 되었다는 겁니다.
- 다만 이 add가 되었다는 건 워킹 디렉토리 -> staging area로 add가 되었다는 것이고, 아직 흔히 말하는 commit이 된 건 아닙니다.

Git사용법



The image shows a Windows File Explorer window and a terminal window. The File Explorer window displays the contents of the 'gitTest' directory, which includes a '.git' folder and a 'git테스트.txt' file. The terminal window shows the output of the 'git status' command, indicating that there are changes to be committed. The changes include a new file 'git\355\205\214\354\212\244\355\212\270.txt'. The terminal also shows the execution of 'git config user.name "Sangpil"' and 'git config user.email "purings2@kakao.com"', followed by 'git commit -m "add test file"', which successfully creates the commit.

```
$ git status
On branch master

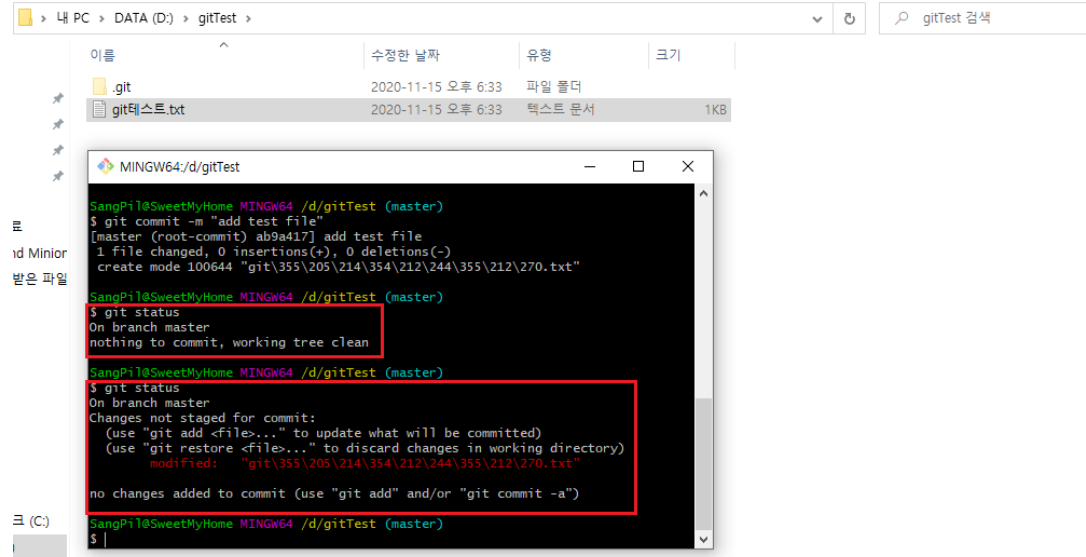
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   "git\355\205\214\354\212\244\355\212\270.txt"

SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git config user.name "Sangpil"
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git config user.email "purings2@kakao.com"
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git commit -m "add test file"
[master (root-commit) ab9a417] add test file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "git\355\205\214\354\212\244\355\212\270.txt"
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ |
```

- staging area에 올린 파일을 하나의 버전으로 남기고 싶다면 commit을 해주어야 합니다.
- 다만 처음 commit을 할 때는 두가지 작업을 해주어야 하는데 user.name과 user.email을 저장하는 작업이 필요합니다.
- 먼저 git config 커맨드를 사용하여 name과 email을 저장한 상태에서 commit을 진행해주세요.
- git commit -m “커밋메시지” 형태로 commit을 하면 됩니다.

Git사용법



The screenshot shows a Windows File Explorer window at the top, displaying the contents of the 'gitTest' directory. It contains a '.git' folder and a 'git테스트.txt' file, both modified on 2020-11-15 at 6:33. Below the file explorer is a terminal window titled 'MINGW64:/d/gitTest'. The terminal shows the following sequence of commands and outputs:

```
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git commit -m "add test file"
[master (root-commit) ab9a417] add test file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "git\355\205\214\354\212\244\355\212\270.txt"

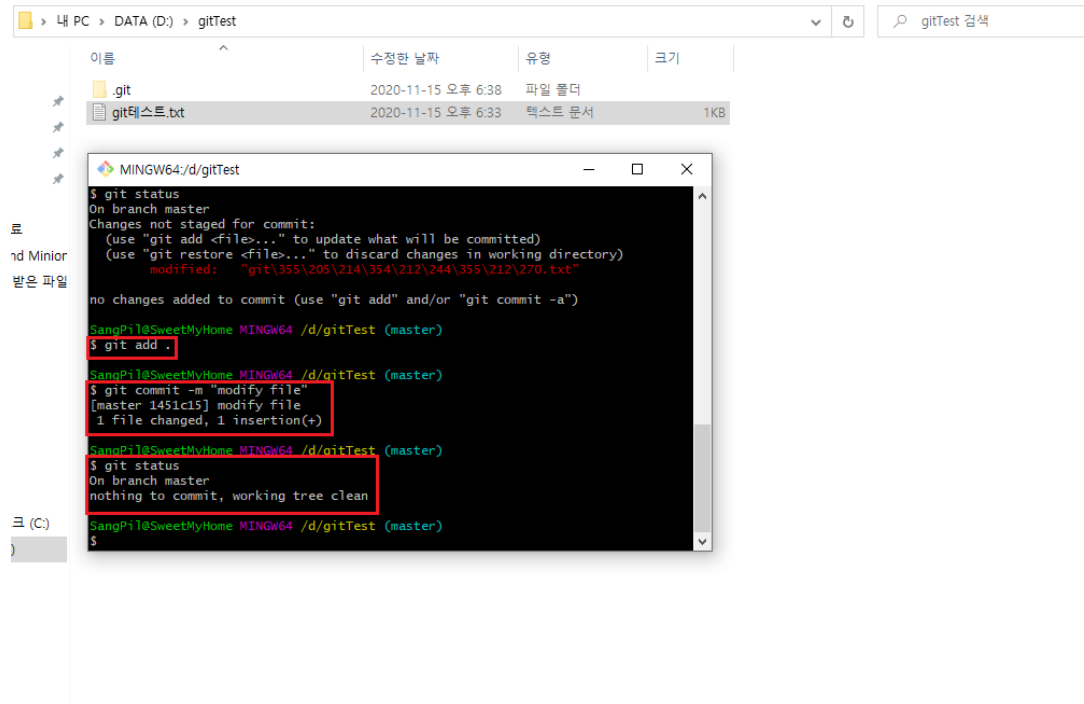
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git status
On branch master
nothing to commit, working tree clean

SangPil@SweetMyHome MINGW64 /d/gitTest (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:   "git\355\205\214\354\212\244\355\212\270.txt"
no changes added to commit (use "git add" and/or "git commit -a")

SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ |
```

- git commit 직후 git status를 하면 nothing to commit, working tree clean 이라고 뜹니다. Commit 이후 어떠한 변경사항도 없기때문에 이런 메시지가 출력되는 것입니다.
- 하지만 이후에 git테스트.txt 파일에 ddddddd라고 내용을 추가한 뒤 저장, git status를 다시해보면 Change not staged for commit이 출력되는 걸 확인 할 수 있습니다.
- 처음엔 git에 add하지 않아 Untracked였지만 이제는 등록이 되었고, commit이후로 수정사항이 생겼기 때문에 Change not staged for commit 라고 표시되는 겁니다.
- 이 시점부터는 내가 어떤 파일을 수정하고 저장해두고 싶은 시점의 순간마다 git add, git commit을 활용하여 버전을 저장하면 됩니다.

Git사용법



The image shows a Windows File Explorer window and a terminal window. The File Explorer window displays the contents of the 'gitTest' directory, which includes a '.git' folder and a 'git테스트.txt' file. The terminal window shows the output of several Git commands: 'git status' (no changes staged), 'git add .' (staging the file), 'git commit -m "modify file"' (committing the file), and 'git status' (nothing to commit, working tree clean).

```
$ 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:   git\355\205\214\354\212\244\355\212\270.txt
no changes added to commit (use "git add" and/or "git commit -a")

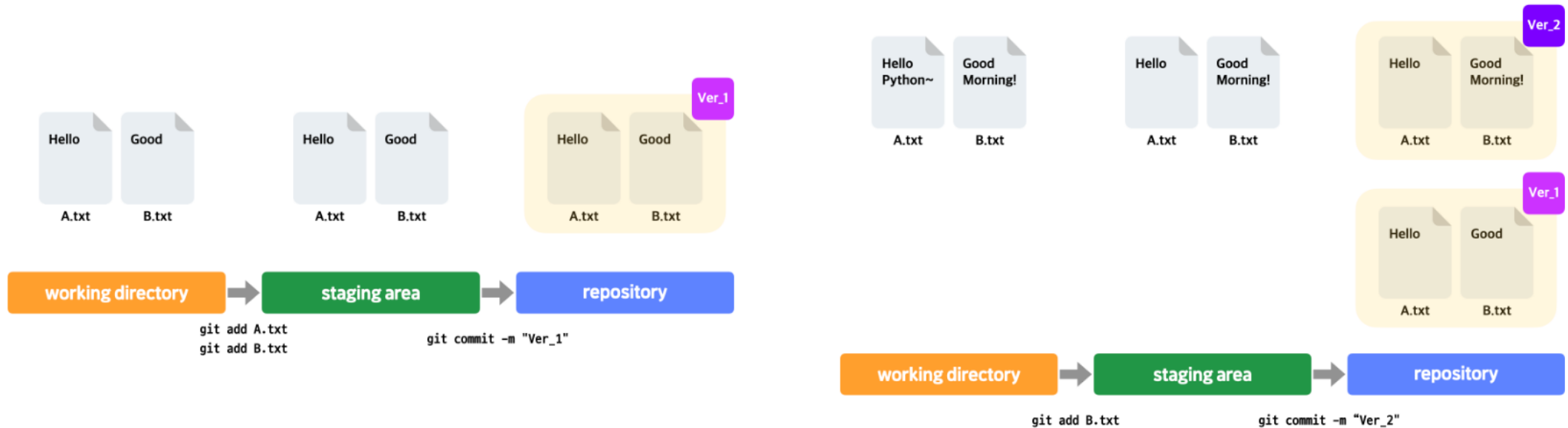
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git add .
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git commit -m "modify file"
[master 1451c15] modify file
1 file changed, 1 insertion(+)

SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git status
On branch master
nothing to commit, working tree clean

SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$
```

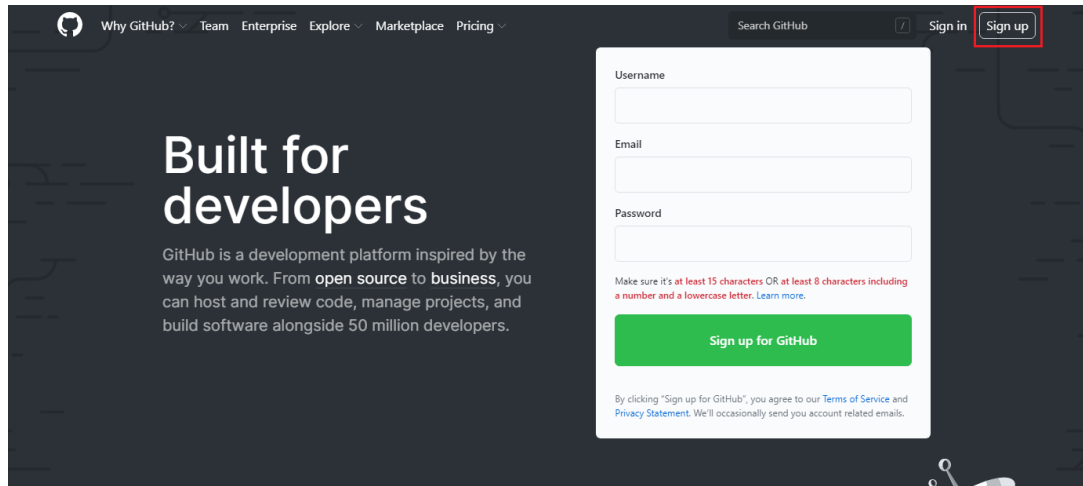
- 변경된 사항을 두번째 버전으로 남기기 위해
git add . 이후 git commit을 한 모습입니다.
- 이렇듯 내가 남기고 싶은 순간마다
 1. git status로 상태 확인
 2. git add . 로 staging area에 추가
 3. git commit -m “커밋메시지” 로 commit
을 해주시면 됩니다!
- 커밋메시지는 가급적 영어로 작성해주세요.

Git의 3가지 영역 한눈에 보기



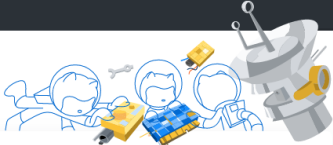
출처 : Codeit

Git + Github 사용법



The screenshot shows the GitHub homepage with a dark background. On the left, the text "Built for developers" is prominent, followed by a description of GitHub as a development platform. On the right, there is a white sign-up form with fields for Username, Email, and Password. A green button labeled "Sign up for GitHub" is at the bottom of the form. Above the form, the "Sign up" link in the top navigation bar is highlighted with a red box.

- Github (<https://github.com/>) 접속
- Sign up을 클릭하여 회원가입 진행해주세요.



Get started with
GitHub Enterprise

Take collaboration to the next level with security and administrative features built for teams.

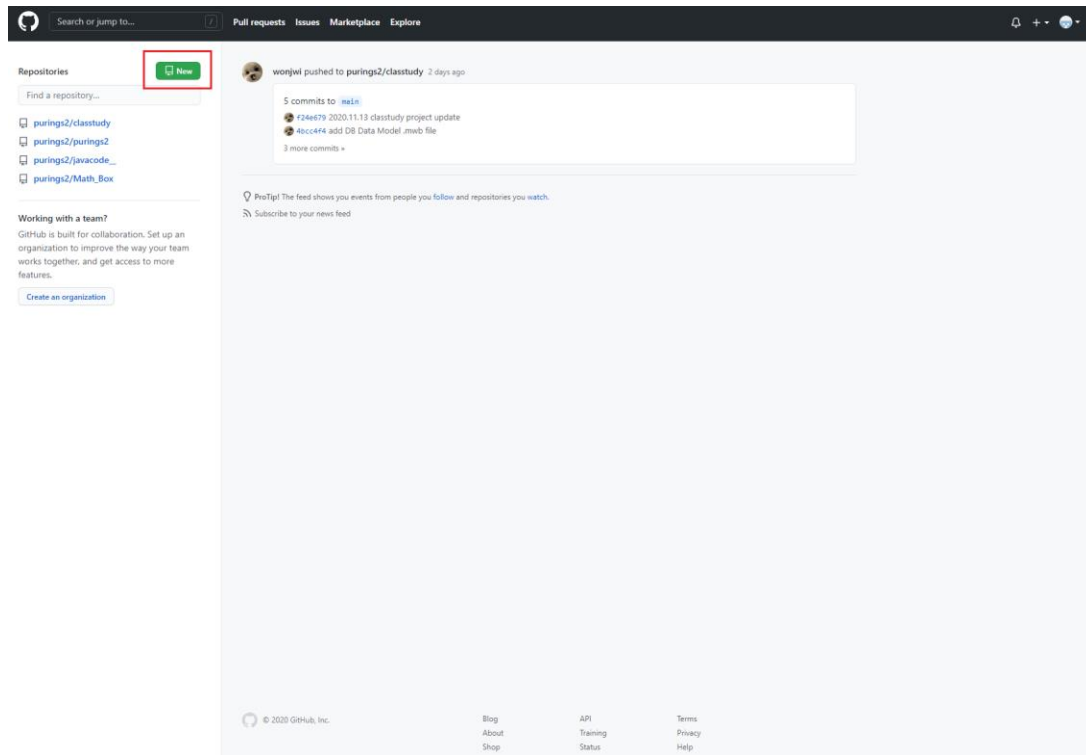
Enterprise
Deploy to your environment or the cloud.

[Start a free trial](#)

Talk to us
Need help?

[Contact Sales →](#)

Git + Github 사용법



- 회원가입 후 Sign in을 누르고 로그인하시면 좌측과 같은 화면이 나오되 아마 저랑은 다르실 겁니다. 저는 이미 레포지토리를 몇 개 만들었고 몇가지 테스트를 위해 작업을 진행해서 화면이 다릅니다. 하지만 이 화면에서 필요한 건 초록색 버튼인 New 입니다!
- New를 클릭해주세요.

Git + Github 사용법

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

Great repository names are short and memorable. Need inspiration? How about refactored-lamp?

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:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

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

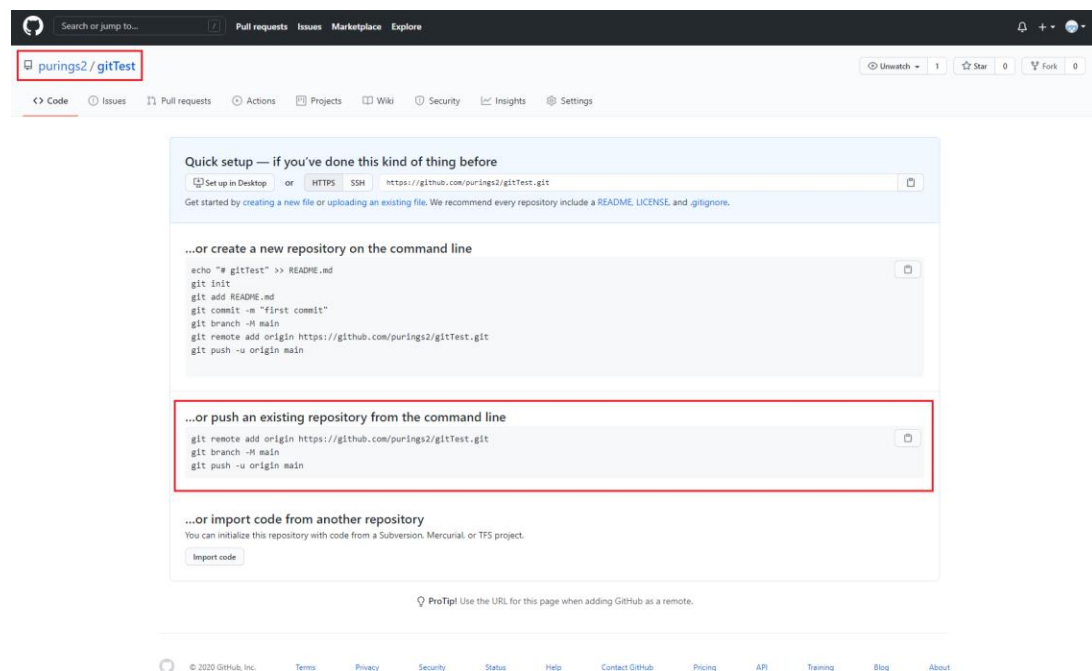
© 2020 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

- Repository name은 github에서 사용할 리모트 레포지토리 이름을 입력해주시면 됩니다. 우리가 먼저 내 컴퓨터에서 만든 gitTest라는 워킹 디렉토리 이름과 같지 않아도 아무 상관없습니다. 저는 그냥 같은 이름으로 만들었을 뿐입니다.

- Description에 적는 것은 자유입니다! 이 리모트 레포지토리가 무엇을 위한 레포지토리 인가에 대한 설명을 적는다고 생각하시면 됩니다. 한글or영어 상관없으며 적지 않으셔도 무방합니다.

- 다 되셨다면 다른 설정은 그대로 두시고 Create repository를 눌러주세요.

Git + Github 사용법



- 리모트 레포지토리를 생성하면 방금 전 repository name에 적었던 단어가 url주소 맨 뒷부분이 되는 겁니다. 지금 이 리모트 레포지토리에 들어오려면 <https://github.com/purings2/gitTest>를 주소창에 입력하면 들어올 수 있습니다.

- 이제 이 리모트 레포지토리와 저희가 만들었던 gitTest 워킹 디렉토리의 로컬 레포지토리를 연결해보겠습니다. 우리는 이미 만들어 놓은 로컬 레포지토리와 연결하기 때문에 2번째 블록의 커맨드를 사용합니다.

- 우리가 연결하고 싶은 워킹 디렉토리(아까 만든 d드라이브 /gitTest)에 들어가 git bash를 실행합니다. 이 부분은 굉장히 중요합니다. 경로를 항상 정확하게 확인하는 습관을 들이세요.

Git + Github 사용법

```
MINGW64:/d/gitTest
$ git status
On branch master
nothing to commit, working tree clean

SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git remote add origin https://github.com/purings2/gitTest.git

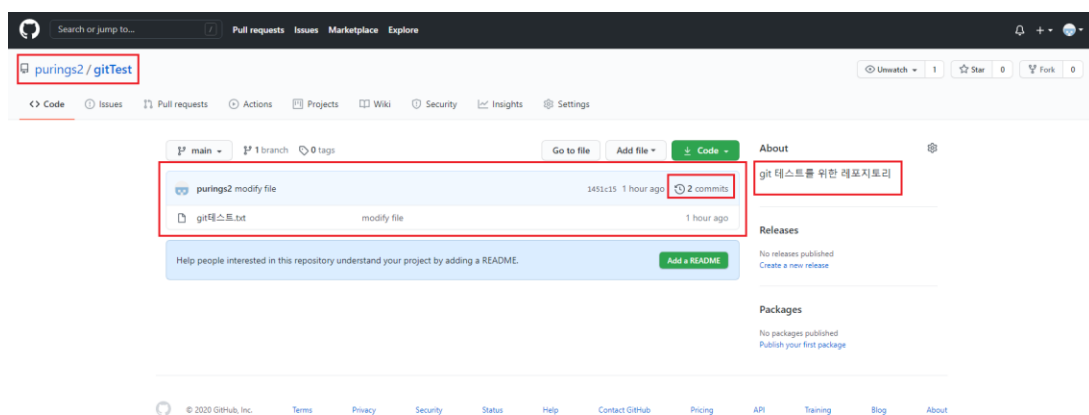
SangPil@SweetMyHome MINGW64 /d/gitTest (master)
$ git branch -M main

SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 442 bytes | 442.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/purings2/gitTest.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ |
```

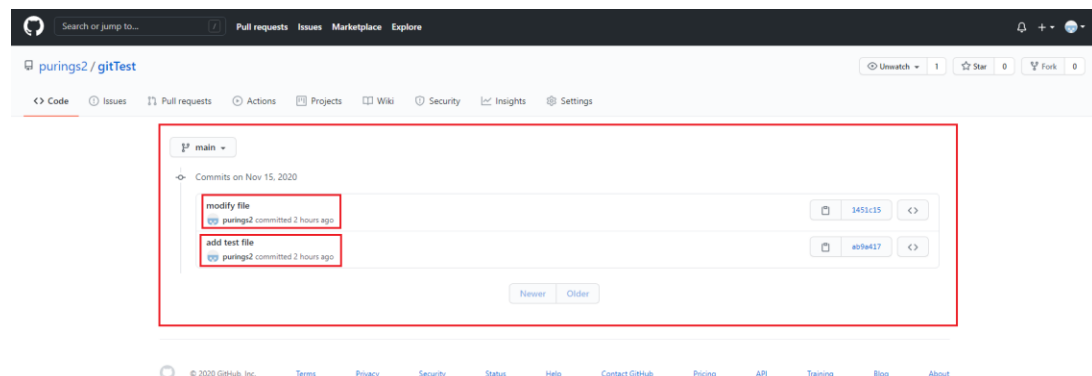
- 연결 하고자하는 워킹 디렉토리에서 git bash here을 해주시고, github 페이지에 있는 3가지 커맨드를 순차적으로 입력해주시면 됩니다. 이때 복사 붙여넣기가 조금 다른데 git bash안에서는 복사 붙여넣기 단축키가 ctrl + insert, shift + insert 입니다. 즉 ctrl + c로 github에서 복사해 오신 다음 shift + insert로 git bash에 붙여넣기 해주시면 됩니다.
- 3가지 커맨드를 모두 완료하시고 github페이지로 돌아가 F5(새로고침)을 한 번 해주세요!

Git + Github 사용법



- 새로그침을 하면 좌측과 같이 화면이 변경 되는걸 확인할 수 있습니다.
- About에 아까 만들 때 작성한 Description이 나타납니다.
- 그리고 저희는 이미 로컬 레포지토리를 만들어 놓고 연동했기에 저희가 이전에 했던 commit(총 2번) 내역이 모두 보이는 걸 볼 수 있습니다. 이제 2 commit을 눌러보세요.

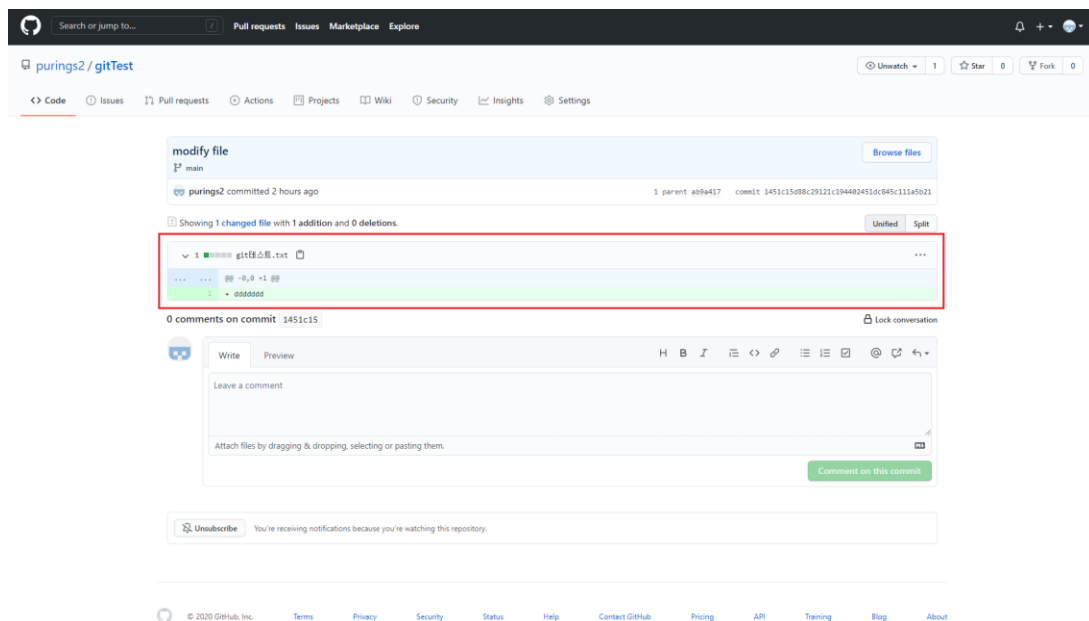
Git + Github 사용법



- 저희가 이전에 git bash에서 2번의 commit을 했는데 그때 커밋메시지로 작성했던 내용이 그대로 적혀있는 걸 볼 수 있습니다.

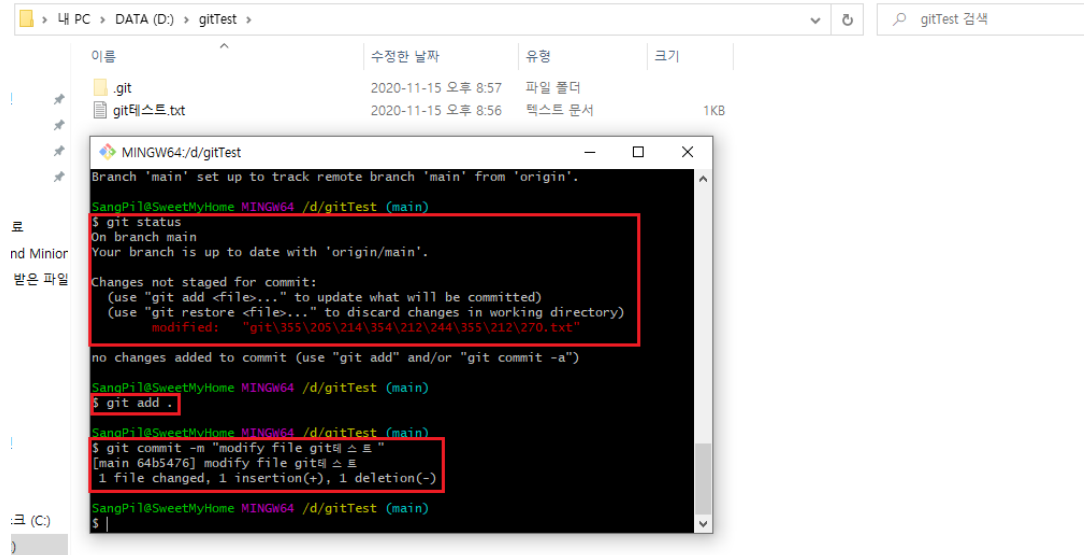
- 이렇듯 git commit -m “커밋메시지” 로 남기는 그 순간이 하나의 버전으로 저장되고, 언제든지 해당 버전을 확인할 수 있으며 그 버전으로 작업할 수 있다는 것이 git의 가장 큰 장점입니다. 두번째로 commit했던 modify file을 눌러보세요.

Git + Github 사용법



- 비어있는 git테스트.txt 파일에 제가 아까 전 임의로 ddddddd를 추가하고 저장했습니다. 이전 commit 버전과 비교하여 어느 부분이 추가되거나 지워졌는지 보여주는데, 저는 ddddddd라고 추가했기 때문에 초록색으로 추가되었다고 알려줍니다.
- 여기까지가 지금까지 내가 로컬 레포지토리에서 작업한 내용을 github와 연동하고 리모트 레포지토리로 불러오는 것 입니다. 이 작업 이후로 내 로컬 레포지토리에서 새롭게 commit 한다면 로컬 레포지토리에는 새로운 commit이 저장되지만 github에 자동으로 해당 commit이 저장되지는 않습니다. 이 때 로컬 레포지토리의 commit을 Github에도 동일하게 적용시키는 커맨드가 git push 입니다.

Git + Github 사용법



The image shows a Windows File Explorer window and a terminal window. The File Explorer window displays the contents of the 'd:\gitTest' directory, which includes a '.git' folder and a 'git테스트.txt' file. The terminal window shows the output of several Git commands: 'git status' (On branch main, Your branch is up to date with 'origin/main'), 'git add .' (no changes added to commit), and 'git commit -m "modify file git테스트"' (1 file changed, 1 insertion(+), 1 deletion(-)).

```
Branch 'main' set up to track remote branch 'main' from 'origin'.
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

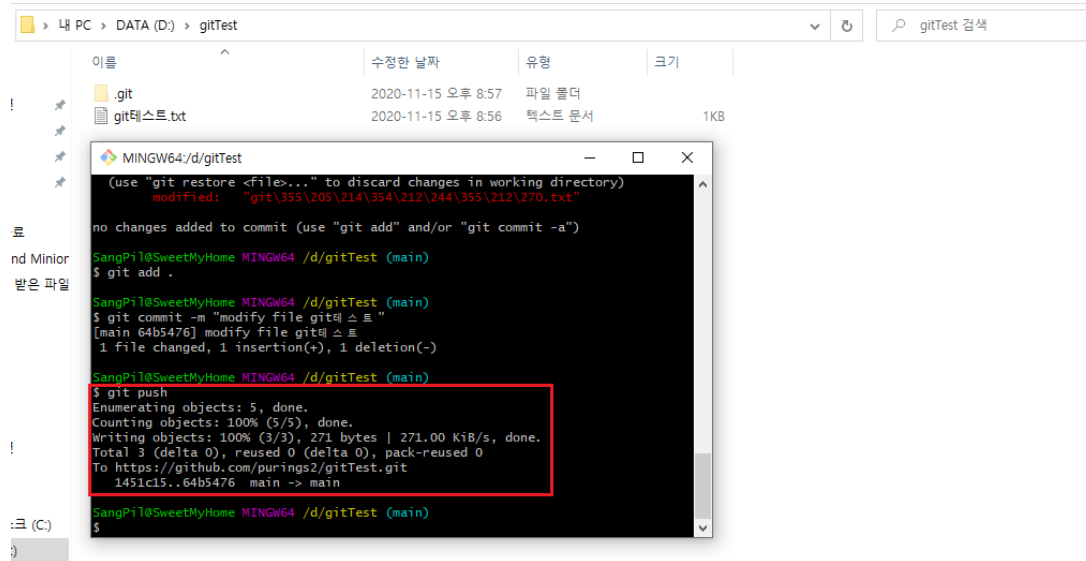
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   "git\355\205\214\354\212\244\355\212\270.txt"

no changes added to commit (use "git add" and/or "git commit -a")
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git add .
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git commit -m "modify file git테스트"
[main 64b5476] modify file git테스트
1 file changed, 1 insertion(+), 1 deletion(-)
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$
```

- 우선 gitTest 워킹 디렉토리안의 git테스트.txt 파일의 내용을 dddddddd에서 ttttt로 바꾸고 저장한 다음 git status와 git add, git commit까지 한 상태입니다.
- 지금 github와도 연동이 되어있고 commit도 했지만 이 상태에서 github로 가서 새로고침을 눌러도 3번째로 한 commit은 github에서 확인이 되지 않습니다.(딱 여기까지만 커맨드 입력하시고 github에서 새로고침 해보세요!)

Git + Github 사용법

- 3번째 commit 내역이 github에 반영되지 않은걸 확인하셨다면 이제 git push 커맨드를 사용해 보세요!
- git push가 성공적으로 완료되면 github 페이지로 돌아가 새로고침을 해주세요.



The screenshot shows a Windows File Explorer window at the path > 내 PC > DATA (D:) > gitTest. It contains two files: .git (folder, modified 2020-11-15 오후 8:57) and git테스트.txt (text file, modified 2020-11-15 오후 8:56, 1KB). Overlaid on this is a terminal window titled 'MINGW64/d/gitTest'. The terminal shows the following commands and output:

```
(use "git restore <file>..." to discard changes in working directory)
modified:   "git\355\205\214\354\212\244\355\212\270.txt"

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

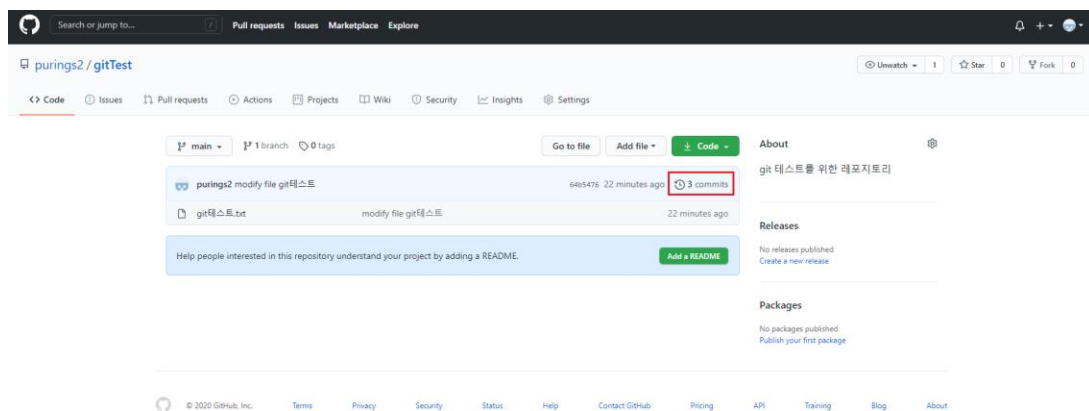
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git add .

SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git commit -m "modify file git테스트"
[main 64b5476] modify file git테스트
1 file changed, 1 insertion(+), 1 deletion(-)

SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 271 bytes | 271.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/purings2/gitTest.git
1451c15..64b5476  main -> main

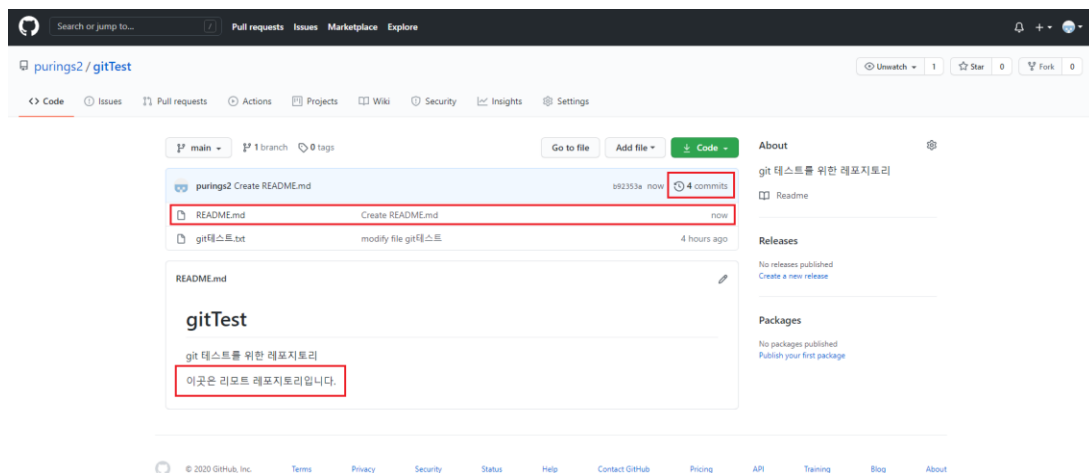
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$
```

Git + Github 사용법



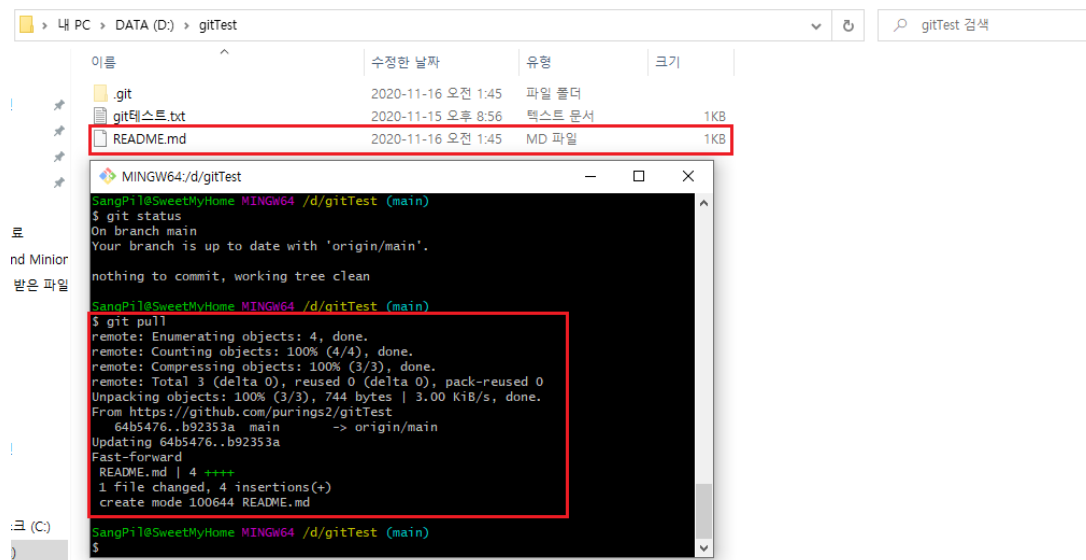
- git push를 사용하기 전엔 내 컴퓨터의 로컬 레포지토리에만 3번째 commit 내역이 저장되어 있었지만 git push를 하는 순간 리모트 레포지토리에도 3번째 commit이 정상적으로 업데이트 된 걸 확인할 수 있습니다.
- 지금처럼 로컬 레포지토리가 리모트 레포지토리보다 더 최신의 자료일 때 push를 해주는데, 이 반대의 경우가 있을 수 있습니다.
- 즉, 리모트 레포지토리가 로컬 레포지토리보다 더 최근 자료인 경우입니다. 이때는 git pull 커맨드를 사용해야 합니다.

Git + Github 사용법



- Github 하단 초록색 글씨로 된 README.md 파일을 열어서 “이곳은 리모트 레포지토리입니다.”라는 내용을 추가한 README.md 파일을 만든 상태가 좌측 화면입니다.
- 커밋이 4개로 늘어났고, README.md 파일이 생겼으며 하단에도 README.md파일의 내용이 출력되고 있는 걸 볼 수 있습니다.
- 하지만 이 상태에서 로컬 레포지토리의 gitTest디렉토리를 새로고침 해도 3번째 커밋한 상태 그대로입니다. 이때 리모트 레포지토리의 최신내용을 로컬 레포지토리에 반영하려면 git pull 커맨드를 사용해야 합니다.

Git + Github 사용법



The image shows a Windows File Explorer window and a terminal window. The File Explorer window displays the contents of the 'gitTest' directory, which includes a '.git' folder, 'git테스트.txt', and 'README.md'. The 'README.md' file is highlighted with a red box. The terminal window shows the output of the 'git status' and 'git pull' commands. The 'git pull' command output is highlighted with a red box, showing that the local repository is up to date with the remote repository.

```
이름      수정한 날짜      유형      크기
.git      2020-11-16 오전 1:45      파일 폴더
git테스트.txt      2020-11-15 오후 8:56      텍스트 문서      1KB
README.md      2020-11-16 오전 1:45      MD 파일      1KB
```

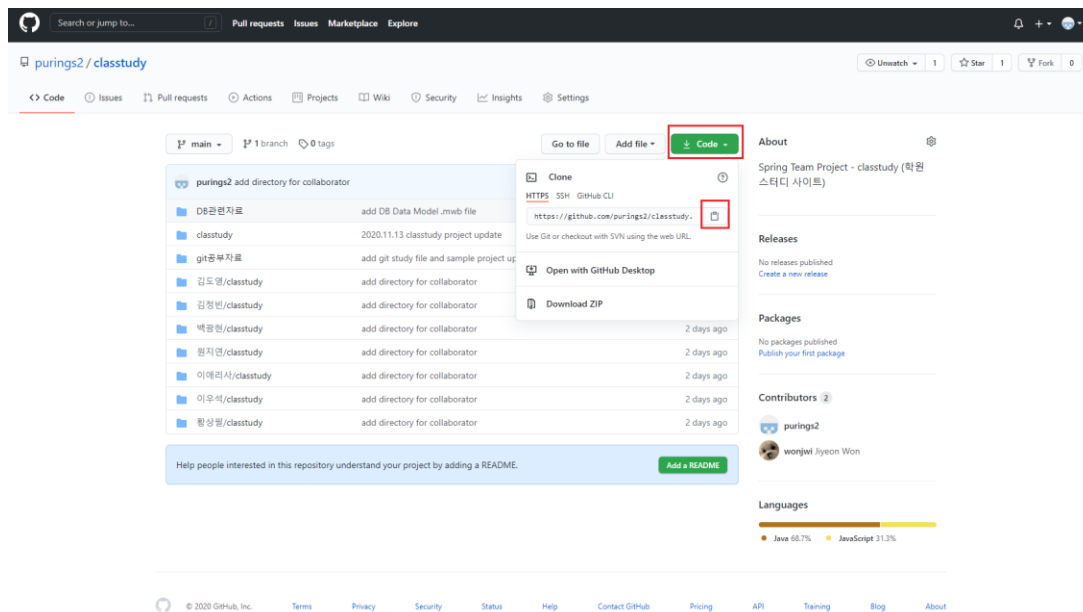
```
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
nothing to commit, working tree clean

SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 744 bytes | 3.00 KiB/s, done.
From https://github.com/purings2/gitTest
  64b5476..b92353a  main    -> origin/main
Updating 64b5476..b92353a
Fast-forward
 README.md | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 README.md

SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$
```

- git pull 커맨드를 사용하면 README.md 파일이 로컬 레포지토리에서도 확인됩니다.
- git push와 git pull은 git 사용 시 무조건 최소 한 번은 사용해야하는 커맨드입니다. 가장 먼저 git pull을 사용해 최신변경사항이 있다면 로컬 레포지토리를 업데이트해주고, 이후 일련의 작업을 하다가 끝낼때는 꼭 git push로 로컬 레포지토리에서 한 작업을 리모트 레포지토리에도 저장해주어야 합니다. 여기까지는 개인이 혼자 작업할 때를 기준으로 작성하였으며, 이 부분들에 대한 정확한 이해가 있어야 협업이 가능합니다. 꼭 정확하게 숙지해주세요.

Git + Github을 활용한 협업

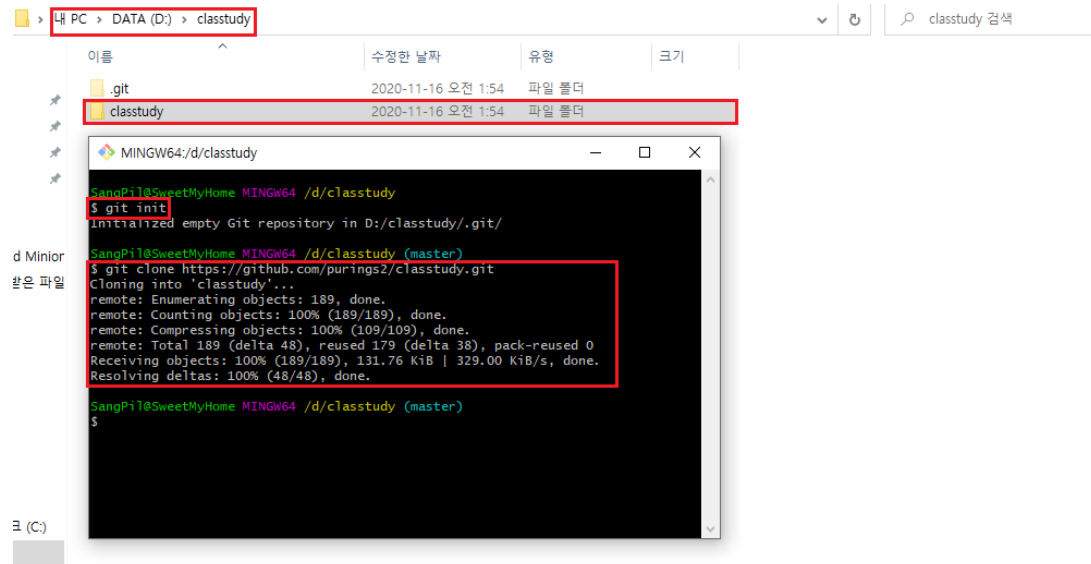


• 이제 본격적으로 하나의 리모트 레포지토리를 공유하여 협업을 하겠습니다. <https://github.com/purings2/classtudy>로 접속해 주세요.

• Code라고 되어있는 초록색 버튼을 누르면 url주소를 확인할 수 있고 오른쪽에 버튼을 누르게 되면 자동으로 복사가 되니 복사해 주세요.

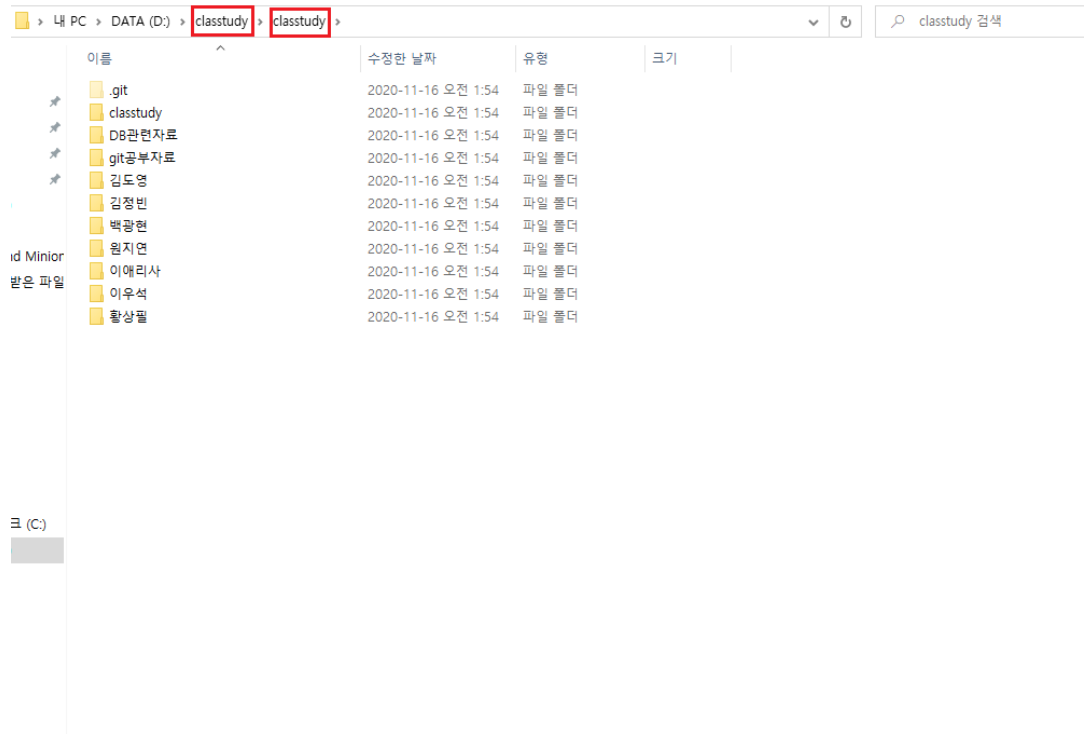
• 우리는 purings2/classtudy 라는 리모트 레포지토리를 사용하여 협업을 할 것이고, 이 레포지토리를 각자 컴퓨터의 로컬 레포지토리와 연결해야 합니다. 지금까지는 본인이 만든 로컬 레포지토리, 리모트 레포지토리를 서로 연결했지만 지금은 다른 사람이 만든 리모트 레포지토리를 나의 로컬 레포지토리와 연결하는 작업입니다. 이 때 사용하는 커맨드가 git clone 입니다.

Git + Github을 활용한 협업



- 우선 리모트 레포지토리의 자료를 저장할 새로운 워킹 디렉토리를 컴퓨터에 만들어줘야 합니다. 저는 D드라이브에 classtudy라는 워킹 디렉토리를 생성했습니다.
- 그리고 해당 워킹 디렉토리에서 우클릭 -> git bash here을 하여 bash를 실행시켰고 우선적으로 이 워킹 디렉토리에서 git 기능을 사용하기위해 git init을 해주었습니다.
- 그리고 이제 리모트 레포지토리의 모든 내용을 그대로 가지고 오는 git clone url주소 커맨드를 사용하여 모든 내용을 가져오시면 됩니다.
- 리모트 레포지토리 이름인 classtudy라는 디렉토리가 생성된 걸 볼수 있고 이 디렉토리에 들어가면 리모트 레포지토리의 모든 자료가 그대로 들어와 있습니다.

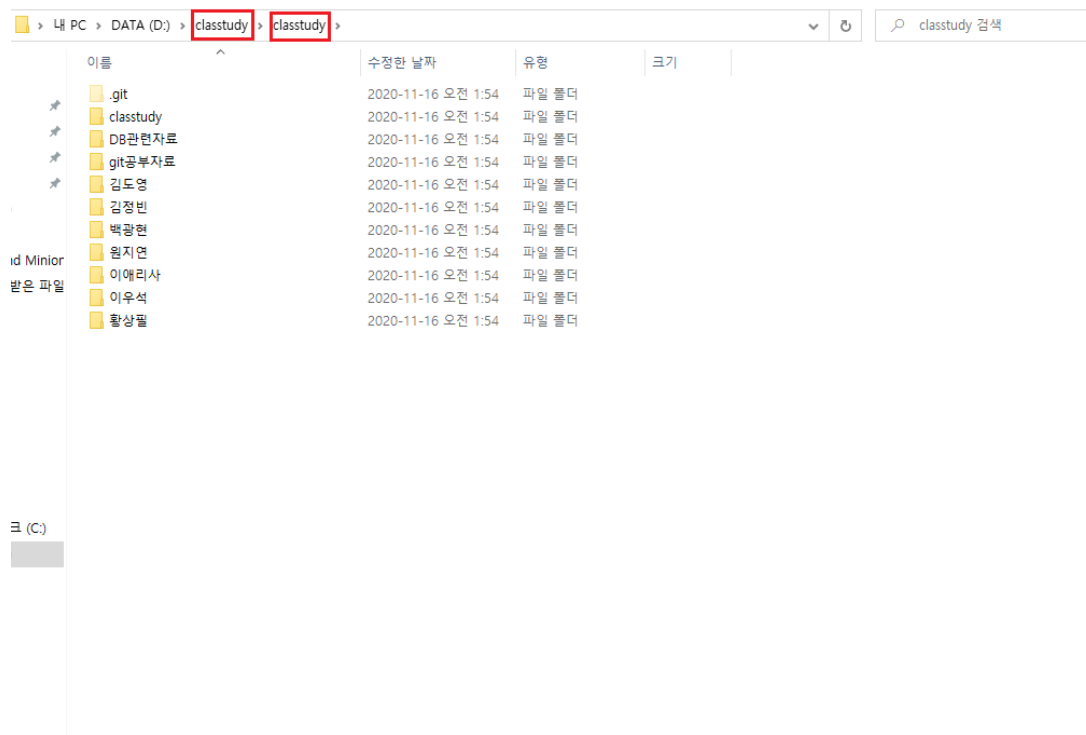
Git + Github을 활용한 협업



- 경로에서 첫번째 classtudy는 제가 로컬 레포지토리에 만들 때 워킹 디렉토리의 이름이며, 두 번째 classtudy는 git clone으로 가지고 온 리모트 레포지토리의 이름입니다.

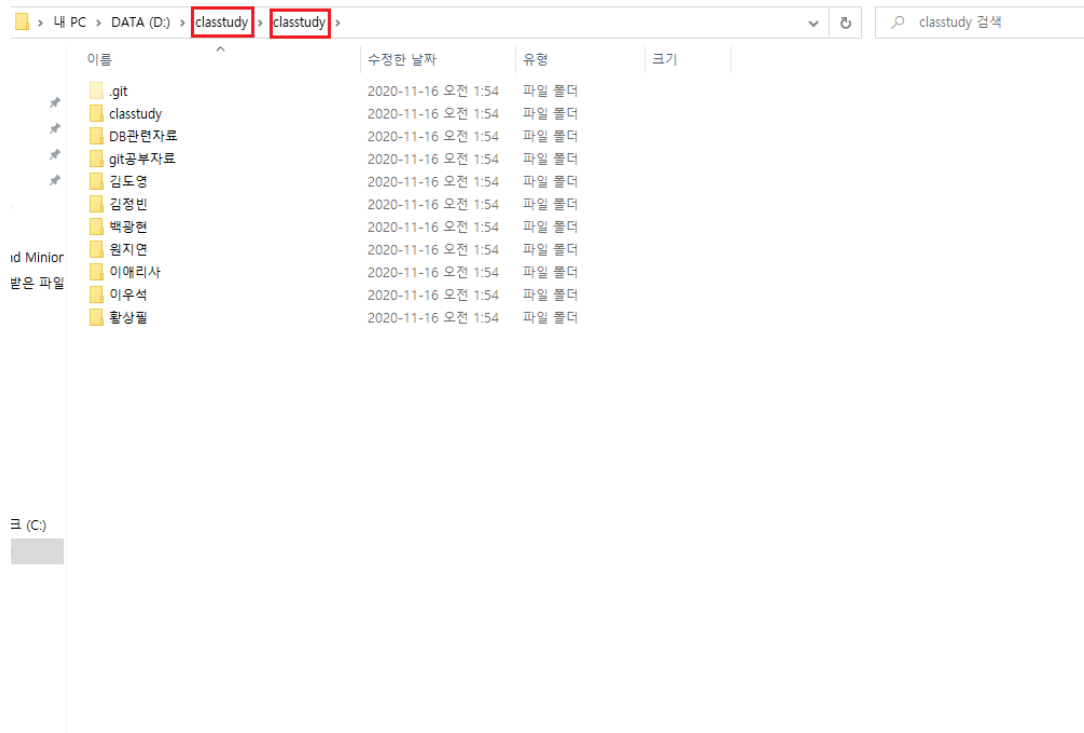
- 앞으로는 모두 이 리모트 레포지토리를 clone한 뒤, 본인 이름이 적혀있는 폴더 안에 들어있는 classtudy spring파일을 기준으로 작업을 하시면 됩니다. 작업 중간중간 저장하고 싶은 순간이 있으시면 commit과 push를 사용해주시면 됩니다.

Git + Github을 활용한 협업



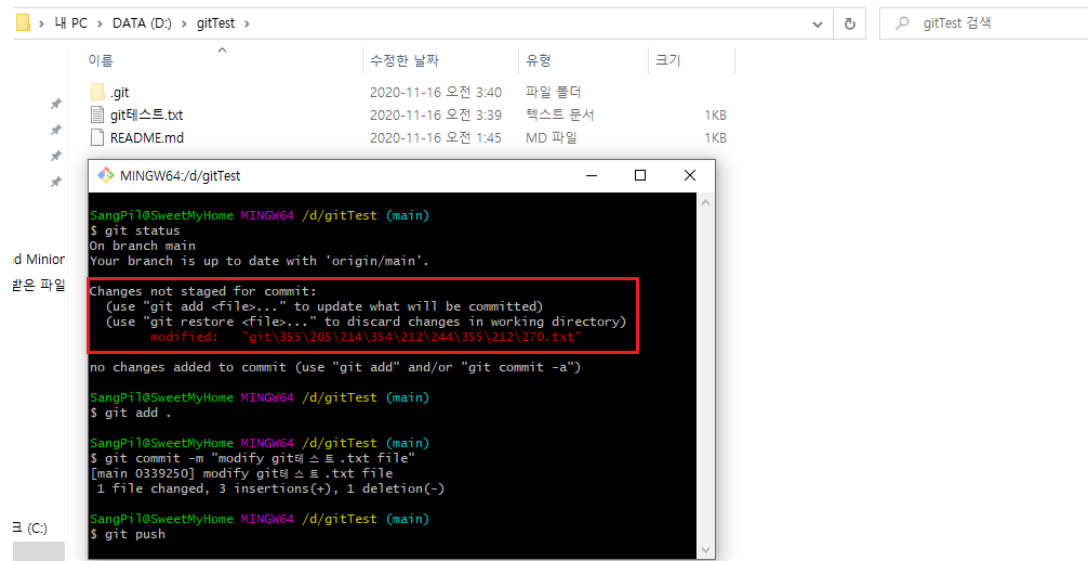
- 이에 따른 주의사항이 있어 말씀드리겠습니다.
- 우선 한 번 클론 한 뒤로는 어떤 작업을 시작하시기전에 git bash를 켜주시고 무조건 git pull을 입력해주세요. 로컬 레포지토리에서 작업을 할 때는 리모트 레포지토리의 가장 최신자료를 기준으로 작업을 시작해야 혼선이 없기때문에 git bash here – git pull은 가장 먼저 꼭 해주셔야 합니다.
- classtudy라는 디렉토리가 하나 더 보이실텐데 저 디렉토리가 우리 프로그램 최종본이 될 디렉토리입니다. 각자가 어떤 기능을 본인 디렉토리에서 시험해보고 잘 되면 다른 팀원들과 내용을 공유 + 확인한 뒤 문제가 없을 때 classtudy를 수정하는 것으로 협업을 진행하겠습니다.

Git + Github을 활용한 협업



- 또한 push를 할 때 꼭 git status -> git add -> git commit -> git push 순서로 해주시고, git status를 했을 당시 bash에 출력되는 내용을 복사해두세요.
- push를 해서 리모트 레포지토리가 업데이트 되면 내가 한 커밋에 들어가 첫번째 코멘트로 복사한 git status 내용을 붙여넣기 해주시고, 두번째 코멘트로 내가 어느 부분을 어떻게 바꿨는지 써주시면 됩니다.
- 다음장에서 그림으로 설명드리겠습니다.

Git push 시 주의사항



The screenshot shows a Windows File Explorer window with the path `> 내 PC > DATA (D:) > gitTest >`. The file list includes `.git`, `git테스트.txt`, and `README.md`. Below the file explorer is a terminal window titled `MINGW64/d/gitTest`. The terminal shows the following commands and output:

```
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git status
On branch main
Your branch is up to date with 'origin/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:   "git\355\205\214\354\212\244\355\212\270.txt"

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

SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git add .

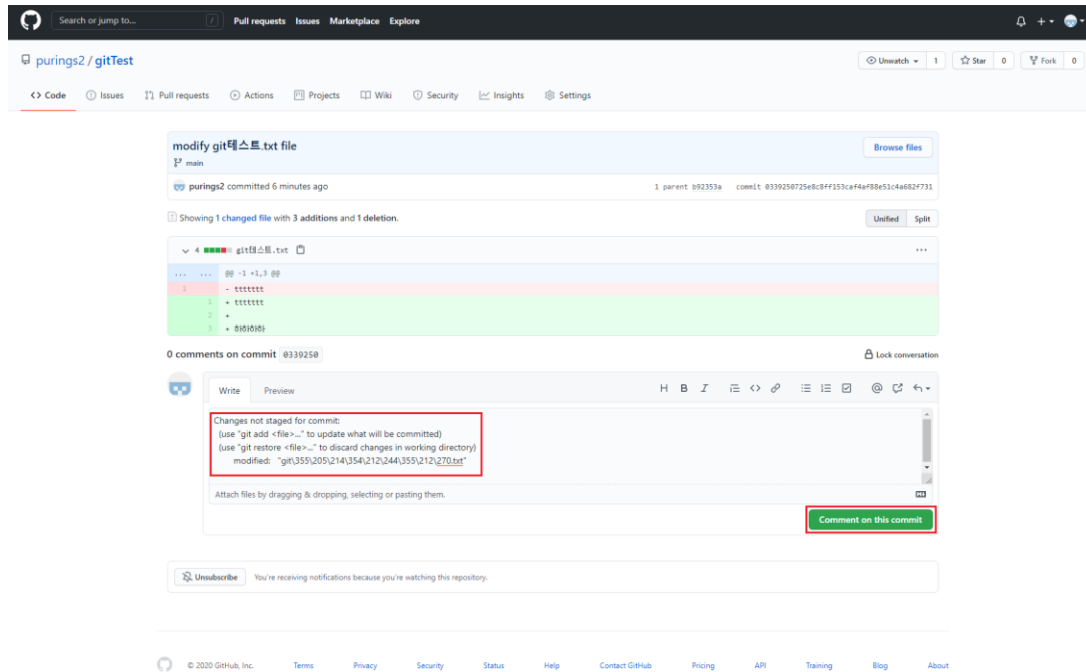
SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git commit -m "modify git테스트.txt file"
[main 0339250] modify git테스트.txt file
1 file changed, 3 insertions(+), 1 deletion(-)

SangPil@SweetMyHome MINGW64 /d/gitTest (main)
$ git push
```

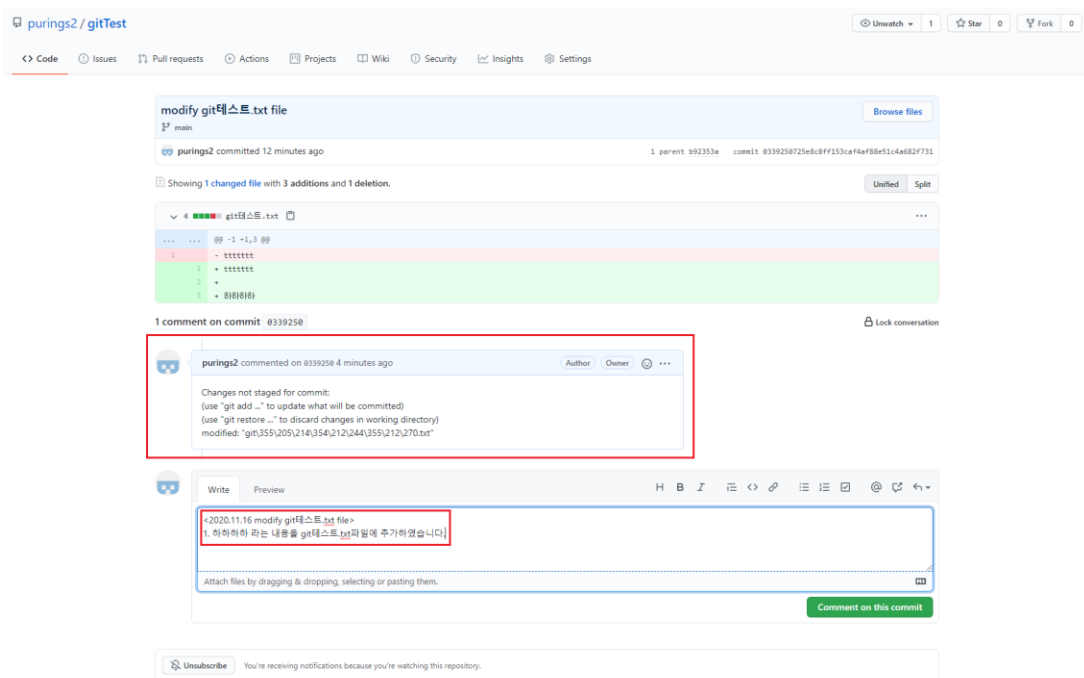
- git status -> git add -> git commit -> git push 커맨드를 순차적으로 입력해 새로운 commit을 리모트 레포지토리에 push했다면 꼭 좌측에 빨간색 박스로 잡아둔 영역을 ctrl + insert로 복사해주세요.
- 이후 연결되어 있는 github 페이지로 가서 commit 내역을 확인해주세요.
- 그리고 내가 마지막으로 commit한 내역을 클릭해주세요.

Git push 시 주의사항

- commit 내역에 들어와서 아래로 쪽 내려보시면 코멘트를 달 수 있습니다. 이 부분에 내가 git bash에서 복사한 status 내용을 붙여넣기 해주시고 Comment on this commit이라는 녹색 버튼을 눌러주세요.



Git push 시 주의사항



- git status 내용으로 코멘트를 작성한 모습이 잘 저장된 걸 볼 수 있습니다.

- 이렇게 git status 내용을 첫 코멘트로 달아주시고, 두 번째는 <날짜 커밋메시지>로 제목을 작성해주시고 어떤 부분을 변경했는지 적어주시면 됩니다. 이 부분은 잘 적어 주신 예시를 아래 링크 달아둘테니 들어가서 확인하시고 앞으로 이 양식과 같이 코멘트 남겨주시면 됩니다.

-

<https://github.com/purings2/classtudy/commit/f24e6798edb9fc3debcaeb05d2365e42eada832f#commitcomment-44153003>

Git + Github를 활용한 협업 총정리

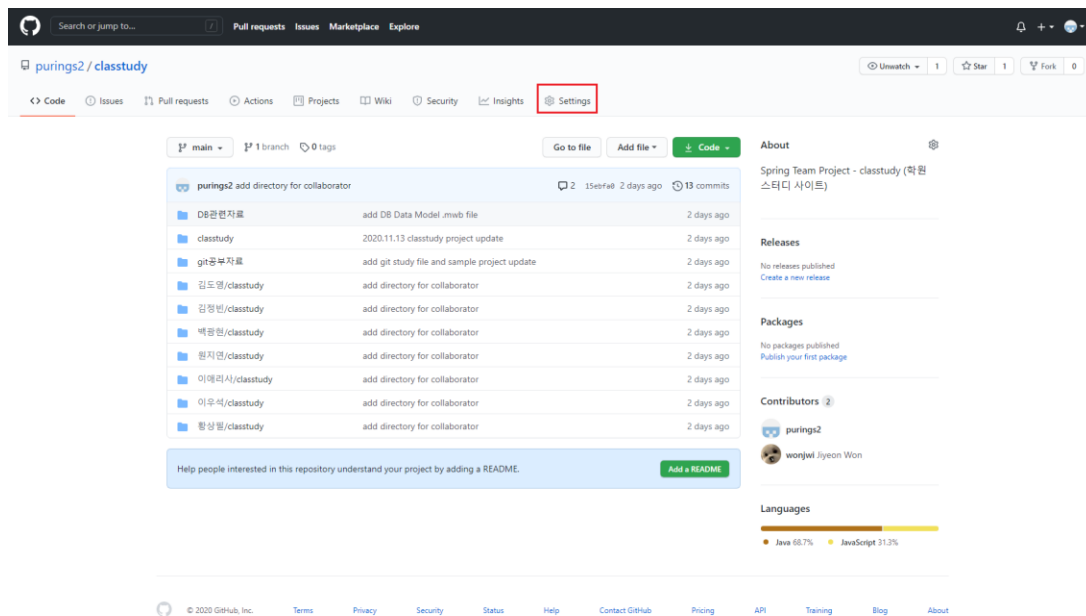
새로운 컴퓨터에서 최초로 한 번만 하는 작업

1. <https://github.com/purings2/classtudy>로 접속해 url주소를 복사한다.
2. 내 컴퓨터에 해당 리모트 레포지토리의 내용을 가지고 와 작업할 워킹 디렉토리를 만들고 git bash를 실행한다.
3. git init으로 해당 워킹 디렉토리를 git을 사용할 수 있도록 만들어준다.
4. git clone <https://github.com/purings2/classtudy>를 입력하여 리모트 레포지토리 내용을 복사해온다.

Clone 이후 작업 방법

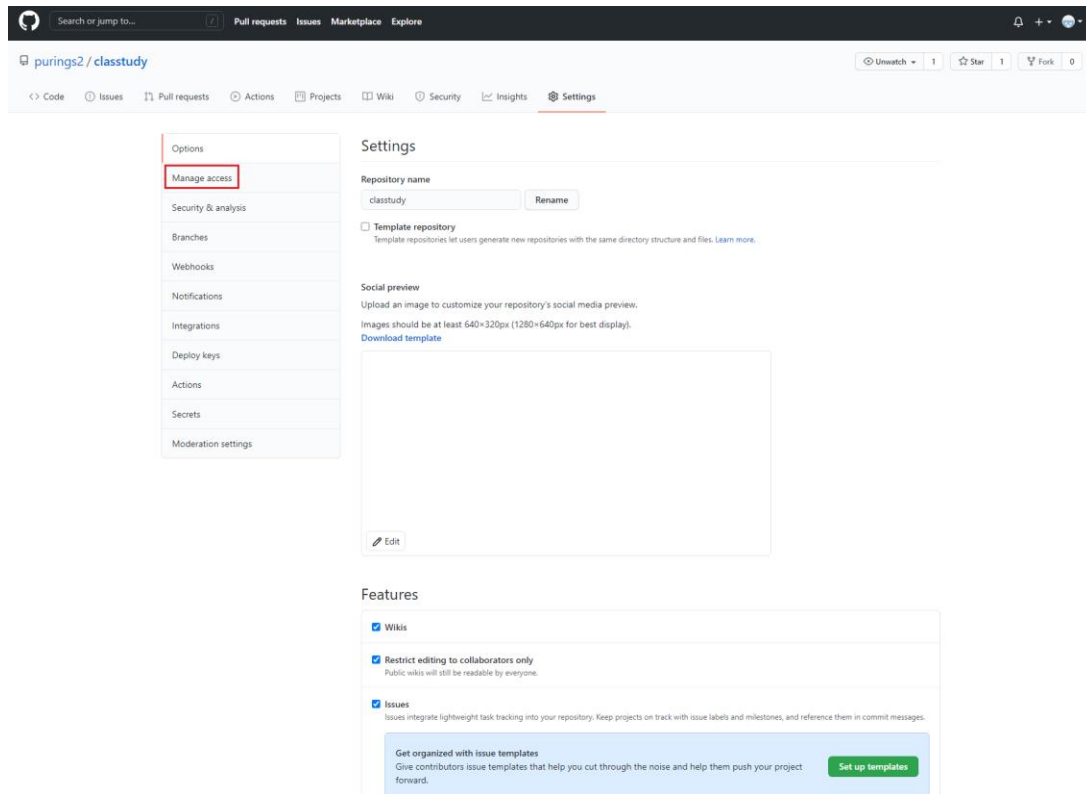
1. git pull 커맨드로 리모트 레포지토리에 변경사항이 있다면 최신 내역으로 로컬 레포지토리를 업데이트한다.
2. 나의 이름이 적힌 디렉토리의 spring project 파일을 가지고 작업을 한다.
3. 작업 중 남기고 싶은 버전이 있으면 git status -> git add -> git commit -m “커밋메시지”를 통해 커밋한다.
- 최초 한 번은 git config user.name “”, git config user.email “” 을 하고 commit 해야 한다.
4. commit 이후 git push를 통해 리모트 레포지토리를 업데이트 한다.
5. git status를 했던 당시 bash에 출력된 내용을 복사하고 commit한 내역에 들어가 코멘트에 그대로 붙여 넣기한다.
6. 어느 부분을 어떻게 수정했는지 코멘트를 하나 더 달도록 한다.

Github push권한 주기



- 기본적으로는 내가 만든 리모트 레포지토리는 다른 사람이 자료를 가지고 갈수는 있지만 push는 할 수 없습니다. Push를 하기 위해선 리모트 레포지토리 관리자 권한을 부여해주어야 합니다.
- Settings로 들어가주세요.

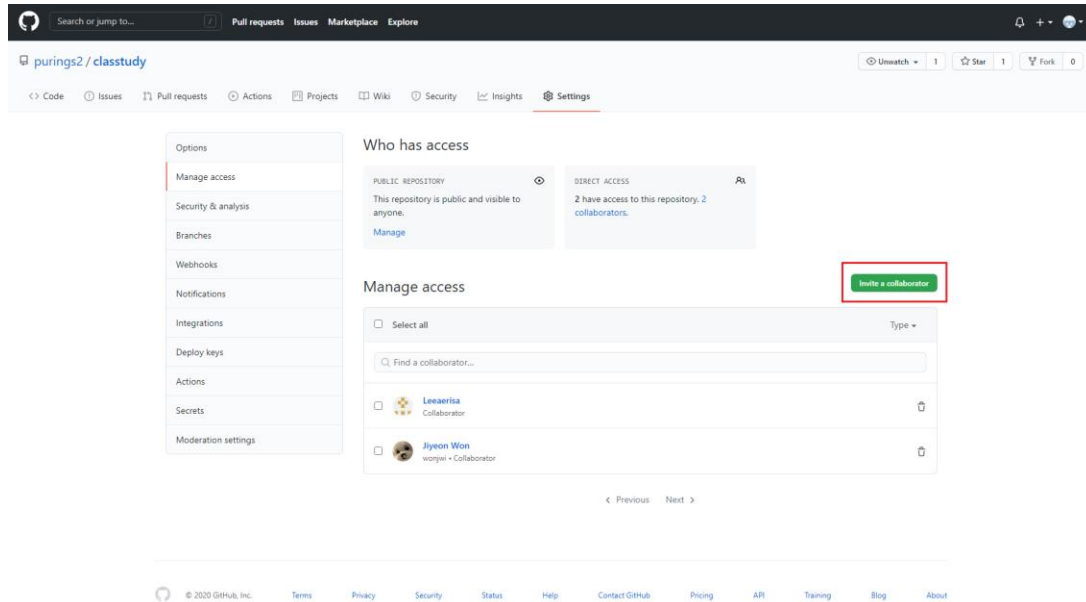
Github push권한 주기



- 다양한 메뉴가 보일텐데 Manage access를 눌러주세요.

- 누르시면 비밀번호를 한 번 더 입력해달라는 화면이 나옵니다. 비밀번호를 정확하게 입력해주세요.

Github push권한 주기



• 아마 첫 화면이 저와는 다르실텐데, 저는 이미 두분께 권한을 드린 상태여서입니다. 이 화면에서 중요한 건 초록색 버튼인 invite a collaborator이고 해당 버튼을 누르면 사용자 아이디를 입력하는 창이 뜹니다.

• 여기에서 초대하고자 하는 사용자의 아이디를 입력하면 그 사용자에게 권한부여에 대한 내용이 담긴 메일이 발송됩니다.

• github가입 시 입력한 메일주소를 들어가 권한을 수락하 시면 이후부터 해당 리모트 레포지토리어서 push가 가능해집니다. 즉, 이 리모트 레포지토리의 collaborator가 되어야만 push가 가능합니다.