

오픈소스SW개론 - Git, Github 기초

세종대학교 이은상

버전관리

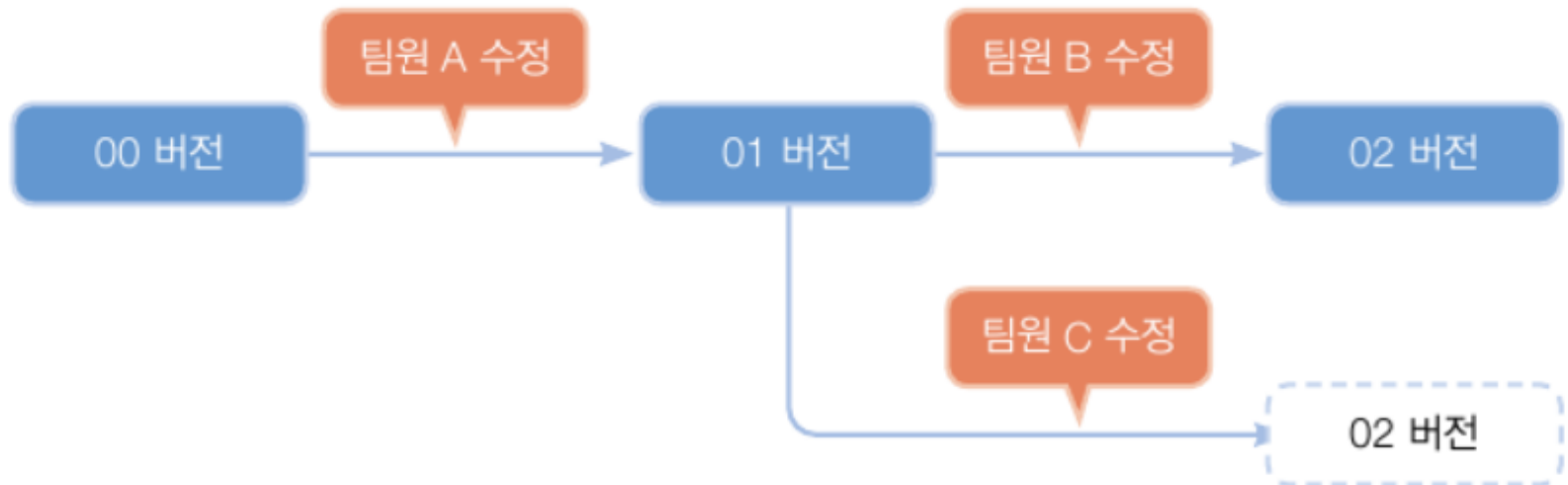
- 버전관리
 - 원하는 시점(버전)으로 이동할 수 있게 해주는 것
- 버전관리 예시
 - RPG 게임
 - 중간에 게임 데이터를 저장함
 - 이후에 load하여 이어서 게임을 진행할 수 있음
 - 포토샵
 - Ctrl+Z를 누르면 이전 상태로 돌아갈 수 있음
 - 이것도 일종의 버전관리

버전관리 시스템 필요성

- 혼자서는 버전관리 시스템 없이도 버전관리 어렵지 않음
- 가령 보고서를 작성할 때 중간중간 버전을 저장해주면 필요할 때 원하는 버전으로 돌아갈 수 있음
 - 오픈소스과제1_230301.hwp
 - 오픈소스과제1_230303.hwp
 - 오픈소스과제1_230306.hwp

버전관리 시스템 필요성

- 팀 프로젝트에 사람들이 참여하면 버전 관리를 잘 해야 함
- 예시)
 - 01 버전을 두 팀원이 동시에 수정하는 상황
 - 어떤 것이 최종 업데이트 파일일지 불분명



버전관리 시스템 필요성

- 예시)
 - 가져오는 것(pull), 비교/수정 등의 기능이 필요



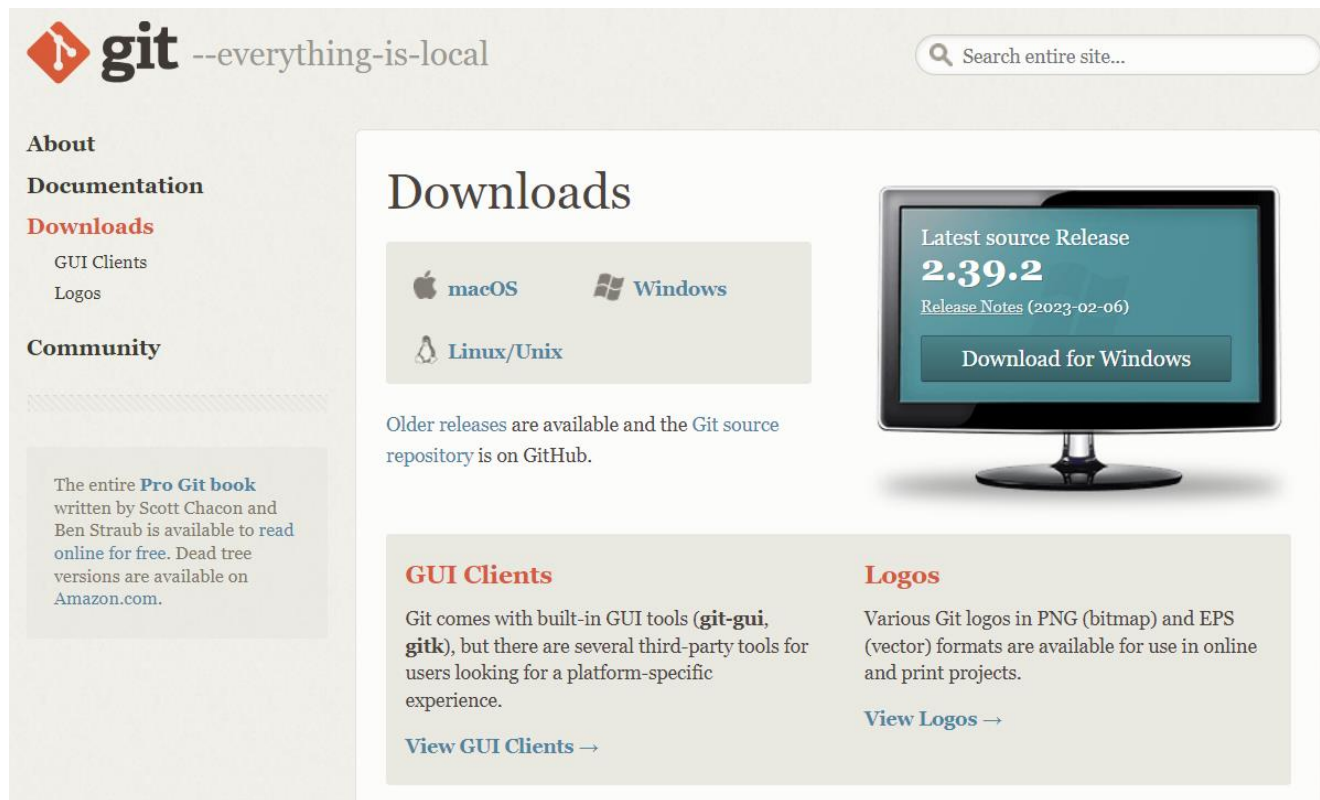
1시	<div>1장 2장 3장</div> <p>1, 2, 3장을 만들어 '고양이버전1'로 저장</p>	<div>4장</div> <p>고양이가 끝나는 것을 기다리지 않고 4장 바로 제작</p>
2시		<div>1장 2장 3장 4장</div> <p>'고양이버전1'을 가져와서 4장을 추가하고 '문어버전1'로 저장</p>
3시	<div>1장 2장 (new) 3장</div> <p>2장을 수정해서 '고양이버전2'로 저장</p>	<div>1장 2장 (new) 3장 4장</div> <p>'고양이버전2'와 '문어버전1'을 비교하니 2장이 바뀐 걸 확인하고 이를 반영해서 '문어버전2'로 저장</p>

Git

- 소스코드 버전 관리 시스템
- 자신이 원하는 시점마다 깃발을 꽂고, 필요시 깃발이 꽂힌 시점으로 자유롭게 이동할 수 있게 함
- 데이터 저장 공간만 있으면 어디서나 git을 사용할 수 있음
 - 개인 컴퓨터, usb, 클라우드 서버 등

Git 설치방법

- <https://git-scm.com/downloads> 로 접속
- 본인의 운영체제에 맞는 링크를 클릭함



Git 설치방법

- "Click here to download" 클릭하여 최신 git 설치파일을 다운받아 실행함

Download for Windows

[Click here to download](#) the latest (**2.39.2**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **4 days ago**, on 2023-02-14.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

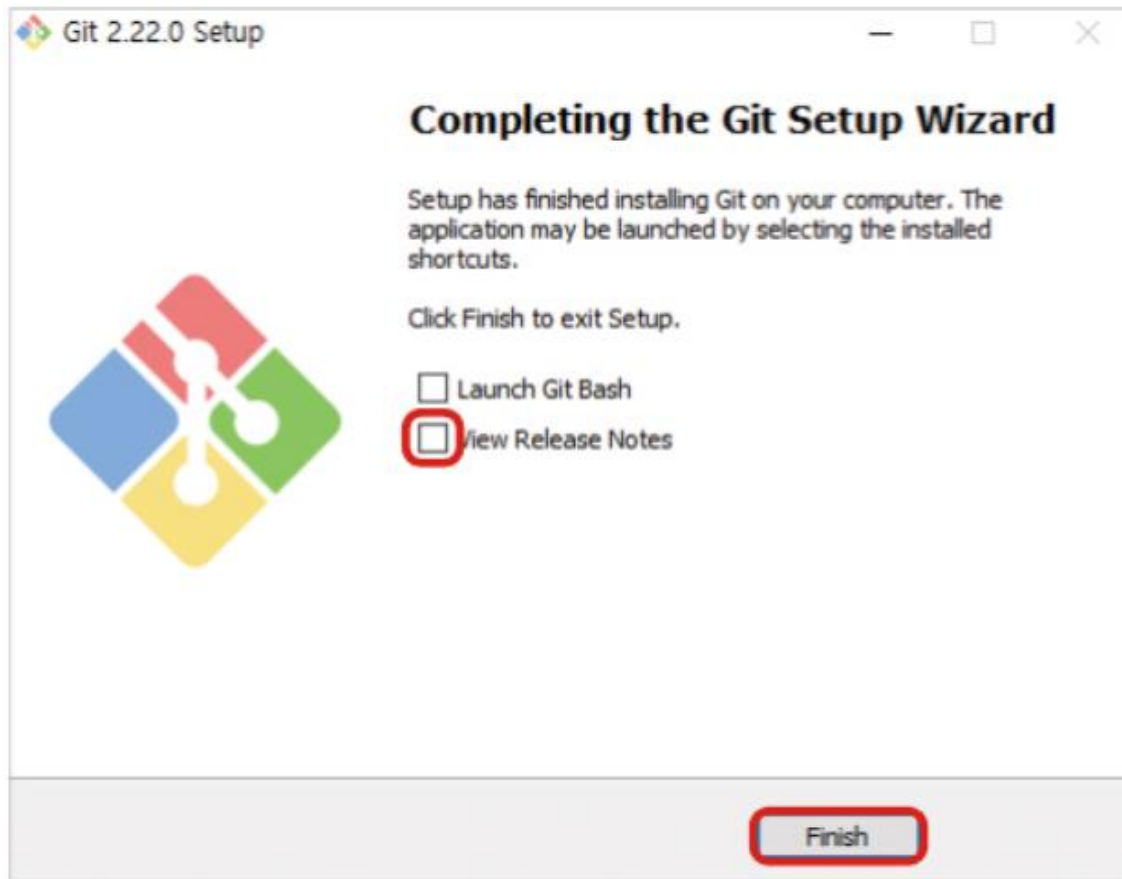
Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

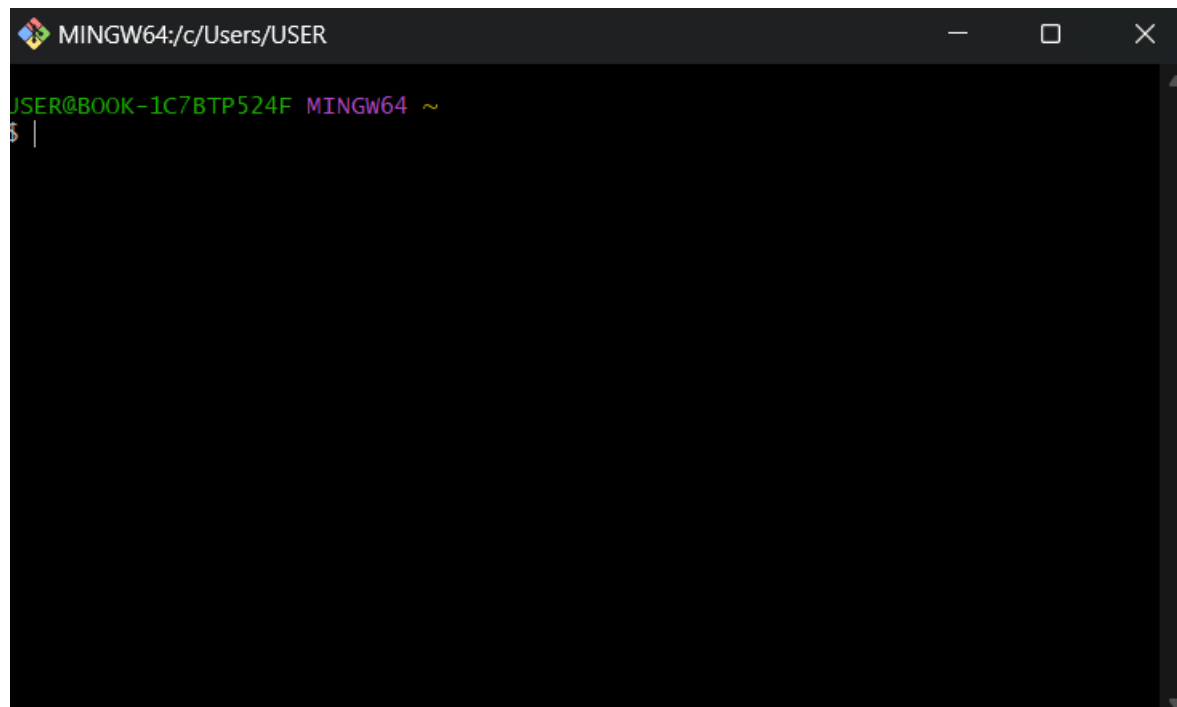
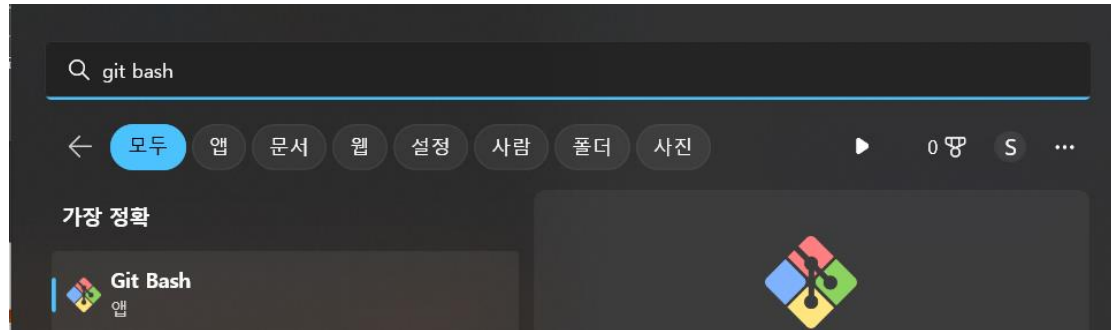
Git 설치방법

- 기본설정을 유지한 채 Next 및 Install 버튼을 클릭해서 Finish까지 감
- "View Release Notes" 체크 해제



Git 설치 확인

- 돋보기 아이콘 클릭해서 Git Bash 실행



Git 설치 확인

- "git" 입력 후 엔터
- 아래 화면이 뜨면 잘 설치된 것임

```
USER@BOOK-1C7BTP524F MINGW64 ~  
$ git  
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]  
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]  
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]  
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]  
          [--super-prefix=<path>] [--config-env=<name>=<envvar>]  
          <command> [<args>]  
  
These are common Git commands used in various situations:  
  
start a working area (see also: git help tutorial)  
  clone      Clone a repository into a new directory  
  init       Create an empty Git repository or reinitialize an existing one  
  
work on the current change (see also: git help everyday)  
  add       Add file contents to the index  
  mv        Move or rename a file, a directory, or a symlink  
  restore    Restore working tree files  
  rm        Remove files from the working tree and from the index
```

호스팅 사이트

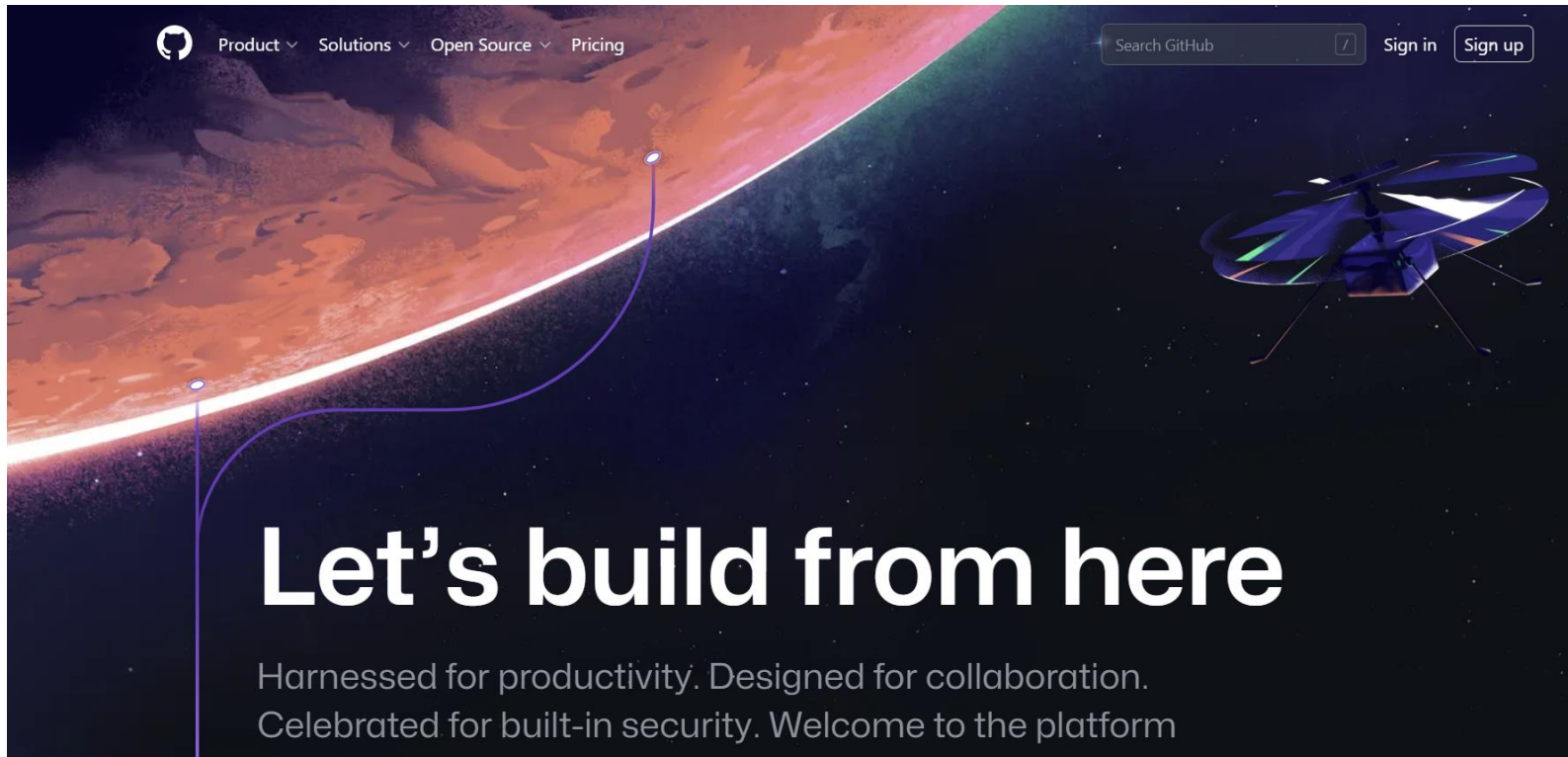
- Git 호스팅 사이트
 - Git으로 관리하는 프로젝트를 올려둘 수 있는 사이트
 - 블로그를 만들 수 있는 곳이 네이버, 다음 등 다양한 것처럼 git으로 관리하는 프로젝트를 올릴 수 있는 사이트도 다양함
 - 예시) github, gitlab, bitbutcket 등

오픈소스

- Github에 소스코드 올려두면 시간, 공간 제약 없이 협업 가능함
- 프로젝트를 공개저장소로 만들면 이름도, 얼굴도 모르는 전 세계 개발자와 협업할 수 있음
- 이렇게 누구든지 기여할 수 있는 오픈소스 공개저장소 프로젝트를 오픈소스라고 함
- 2017년 기준 github에 6700만개 공개저장소가 있음
- Tensorflow, Swift, 뷰 등 다양한 오픈소스가 활발하게 운영됨

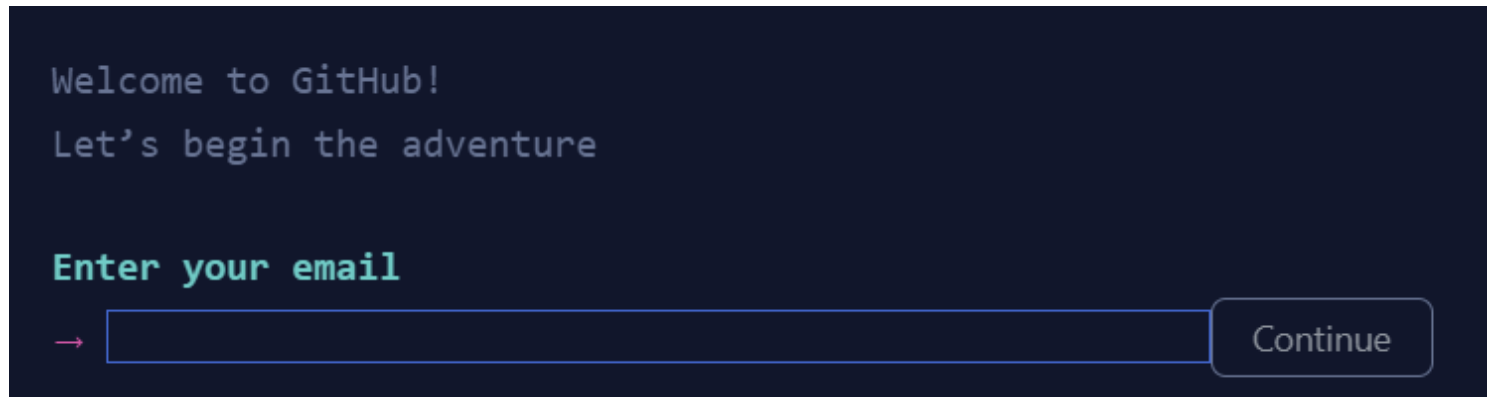
Github 가입 방법

- <https://github.com/> 사이트에 접속함
- 우측 상단에 [Sign up] 버튼 클릭



Github 가입 방법

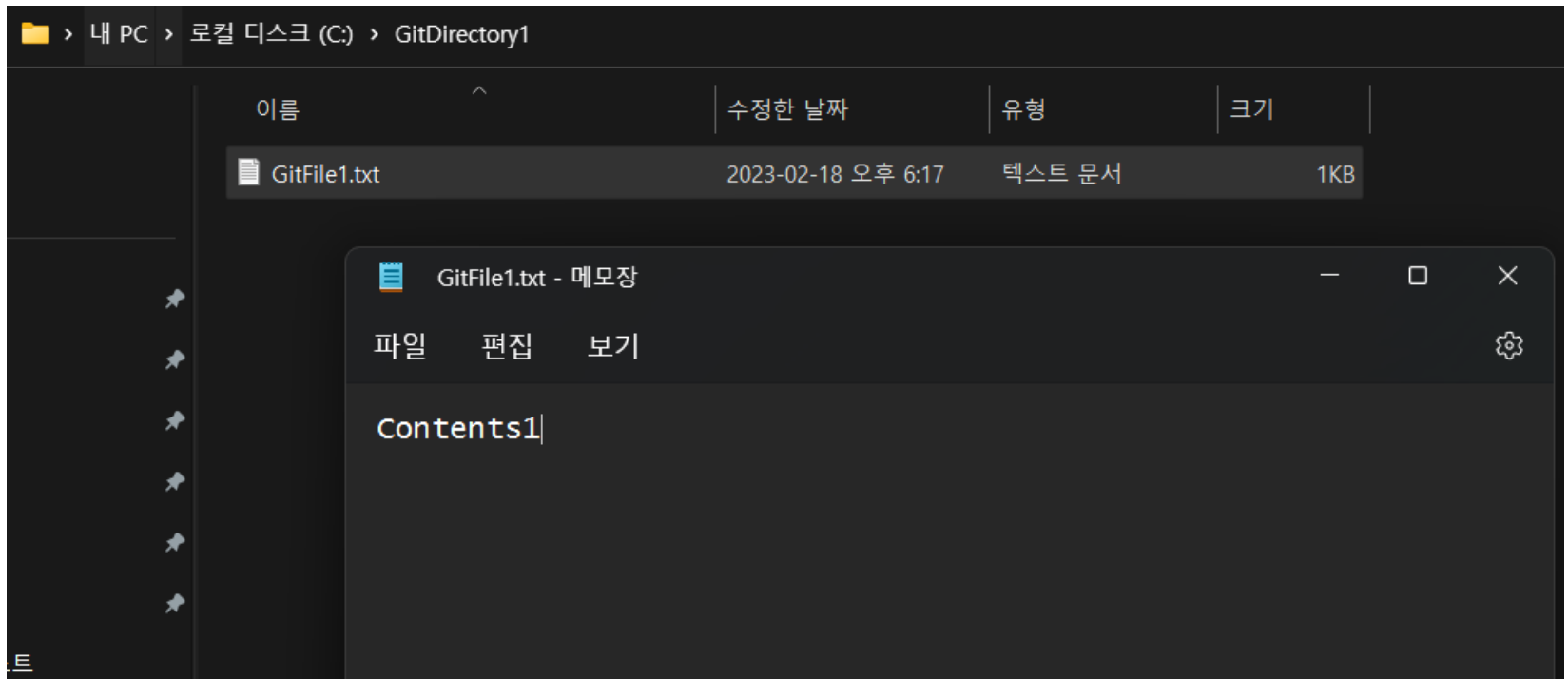
- 이메일 주소 입력 후 [Continue] 버튼 클릭
- 메일 인증 진행

A screenshot of the GitHub sign-up process. The background is dark blue. At the top, it says "Welcome to GitHub!" and "Let's begin the adventure" in a light blue monospace font. Below that, the text "Enter your email" is displayed in a green monospace font. Underneath is a long, empty text input field with a small red arrow pointing to its start. To the right of the input field is a rounded rectangular button with the word "Continue" in a light blue font.

- 비밀번호 입력 후 [Continue] 버튼 클릭
- Username 입력 후 [Continue] 버튼 클릭하면 가입 완료

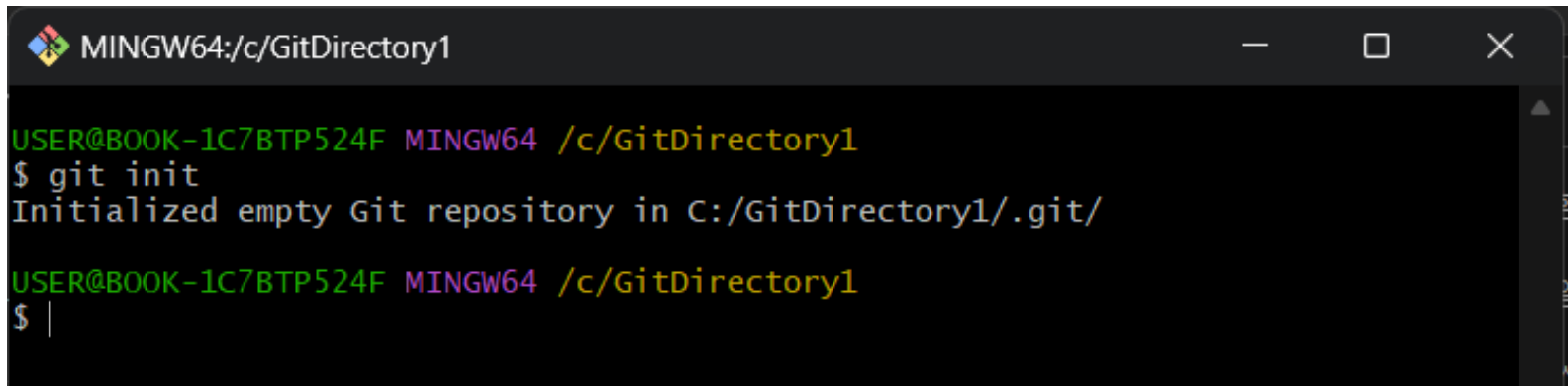
로컬 저장소 생성 실습

- 로컬 저장소는 실제 git으로 관리할 폴더를 말함
 - C:\GitDirectory1
- 해당 폴더에서 GitFile1.txt 텍스트 파일 생성함
- 텍스트 파일 열고 "Contents1" 라고 적어주고 저장함



로컬 저장소 생성 실습

- GitDirectory1 폴더에서 마우스 오른쪽 버튼 클릭하고 [Git Bash Here] 클릭함 ([더 많은 옵션 표시]를 먼저 누른 후 [Git Bash Here]를 클릭해야할 수도 있음)
- 혹은 Git Bash 창에서 GitDirectory1 폴더로 직접 들어가도 상관없음
- Git Bash 창이 열리면 아래 명령어를 입력함
 - git init

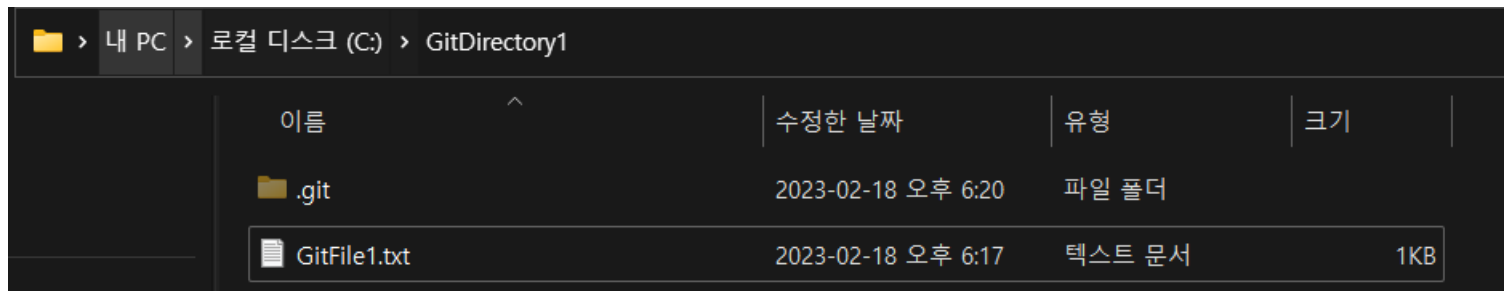


```
MINGW64:/c/GitDirectory1
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1
$ git init
Initialized empty Git repository in C:/GitDirectory1/.git/
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1
$ |
```

- "Initialized empty Git repository"라는 텍스트가 나오면 성공

로컬 저장소 생성 실습

- 명령어 실행 후 GitDirectory1 폴더에는 .git이라는 숨김 폴더가 자동적으로 생성됨



- 그러면 로컬저장소가 생성된 것이고 이 때부터 이 폴더에서 버전관리를 할 수 있음

로컬 저장소 생성 실습

- .git 폴더
 - .git 폴더는 git으로 생성한 버전들의 정보와 원격저장소 주소 등이 들어있는 숨겨진 폴더임
 - 이 .git 폴더를 로컬저장소라고 부르며 이 폴더에다 버전 관리를 할 수 있음

커밋 실습

- 커밋 개념
 - 생성하는 각 버전 혹은 각 버전을 통해 생성된 파일을 커밋이라고 함
 - RPG 게임에서의 '저장'이라고 생각하면 됨
 - 커밋을 행할 때마다 새로운 버전을 하나씩 만드는 것임

커밋 실습

- 자신의 정보 등록

- 커밋을 위해서는 먼저 자신의 정보를 .git 폴더에 등록해야함
- GitDirectory1 폴더에서 마우스 오른쪽 버튼 클릭하고 [Git Bash Here]를 클릭함
- 아래 두 명령어를 git bash에 입력함
 - git config --global user.email "이메일 주소"
 - git config --global user.name "유저이름"
 - ex)

```
git config --global user.email "eslee3209@sejong.ac.kr"
```

```
git config --global user.name "eslee3209"
```

커밋 실습

- 파일 선택
 - 커밋에 추가할 파일을 선택함
 - 아래 명령어 입력
 - git add [파일명]
 - ex) git add GitFile1.txt

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)  
$ git add GitFile1.txt
```

- 혹시 dubious ownership 에러가 생기는 경우 시키는 대로 명령어를 입력하면 됨

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1  
$ git add .  
fatal: detected dubious ownership in repository at 'C:/GitDirectory1'  
'C:/GitDirectory1' is owned by:  
    'S-1-5-32-544'  
but the current user is:  
    'S-1-5-21-2567812226-3210159831-4087323491-1001'  
To add an exception for this directory, call:  
  
    git config --global --add safe.directory C:/GitDirectory1
```

커밋 실습

- 커밋

- 커밋에 상세한 설명을 적음
- 설명 잘 적어놓으면 파일이 생성된 이유, 수정된 이유 등을 알 수 있음
- 나중에 원하는 버전 찾아 시간여행하기도 수월함
- 다른 개발자 및 미래의 자신을 위해 최대한 공들여 적는 것이 유용
- `git commit -m "커밋 설명"`
- ex) `git commit -m "Commit1"`

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git commit -m "Commit1"
[master (root-commit) d296e52] Commit1
1 file changed, 1 insertion(+)
create mode 100644 GitFile1.txt
```

- 위와 같이 되면 성공적으로 커밋이 된 것임

커밋 실습

- 두 번째 커밋
 - GitFile1.txt 내용을 다음과 같이 수정함
 - Contents1 -> Contents2
 - git add GitFile1.txt
 - git commit -m "Commit2"
 - 그러면 두 번째 커밋까지 생성됨

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git add GitFile1.txt

USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git commit -m "Commit2"
[master 62c466e] Commit2
1 file changed, 1 insertion(+), 1 deletion(-)
```


커밋 실습

- 히스토리를 보면 커밋이 잘 되었는지 확인 가능
- 히스토리 확인 방법
 - git log
 - 위 명령어를 입력하면 커밋 히스토리를 확인할 수 있음

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log
commit 62c466e6a569abefd5227043cae5ebab87bcf72b (HEAD -> master)
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date: Sat Feb 18 18:40:55 2023 +0900

    Commit2

commit d296e52b7bfc15b1b29c5d98133cf52a8175b7a7
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date: Sat Feb 18 18:37:16 2023 +0900

    Commit1
```

Checkout 실습

- Checkout 개념
 - 다른 커밋으로 시간여행하는 것
 - 예시) 개발하다 요구사항이 바뀌어 이전 커밋으로 돌아가 다시 개발해야하는 경우 생길 수 있음
 - 그 커밋으로 돌아가는 것이 checkout

Checkout 실습

- Checkout 방법

- git log 명령으로 commit 아이디를 확인함

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log
commit 62c466e6a569abefd5227043cae5ebab87bcf72b (HEAD -> master)
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date: Sat Feb 18 18:40:55 2023 +0900

    Commit2

commit d296e52b7bfc15b1b29c5d98133cf52a8175b7a7
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date: Sat Feb 18 18:37:16 2023 +0900

    Commit1
```

- 위 첫번째 커밋의 경우 아이디는
d296e52b7bfc15b1b29c5d98133cf52a8175b7a7
 - commit 아이디를 확인한 후 7자리를 떼어 아래와 같이 명령함
(전체 아이디를 복사해도 상관없음)
 - git checkout [커밋아이디 7자리]
 - 예시) git checkout d296e52

Checkout 실습

- Checkout 방법
 - 아래와 같이 "HEAD is not at ..." 가 뜨면 잘 이동한 것임

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git checkout d296e52
Note: switching to 'd296e52'.

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 d296e52 Commit1
```

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 ((d296e52...))
$ git log
commit d296e52b7bfc15b1b29c5d98133cf52a8175b7a7 (HEAD)
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date: Sat Feb 18 18:37:16 2023 +0900

    Commit1
```

Checkout 실습

- Checkout 방법
 - GitFile1.txt 파일을 열어보면 "Contents1" 내용이 들어있음 ("Contents2"가 아니라)
 - 즉, 첫 번째 커밋으로 잘 이동한 것을 확인할 수 있음

Checkout 실습

- 가장 최근 커밋으로 checkout
 - 가장 최근 커밋으로 이동하려면? 커밋 아이디 사용하는 것도 가능하지만 아래와 같은 명령어로 더 간단하게 됨
 - git checkout -

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 ((d296e52...))  
$ git checkout -  
Previous HEAD position was d296e52 Commit1  
Switched to branch 'master'
```

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)  
$ git log  
commit 62c466e6a569abefd5227043cae5ebab87bcf72b (HEAD -> master)  
Author: eslee3209 <eslee3209@sejong.ac.kr>  
Date: Sat Feb 18 18:40:55 2023 +0900  
  
    Commit2  
  
commit d296e52b7bfc15b1b29c5d98133cf52a8175b7a7  
Author: eslee3209 <eslee3209@sejong.ac.kr>  
Date: Sat Feb 18 18:37:16 2023 +0900  
  
    Commit1
```

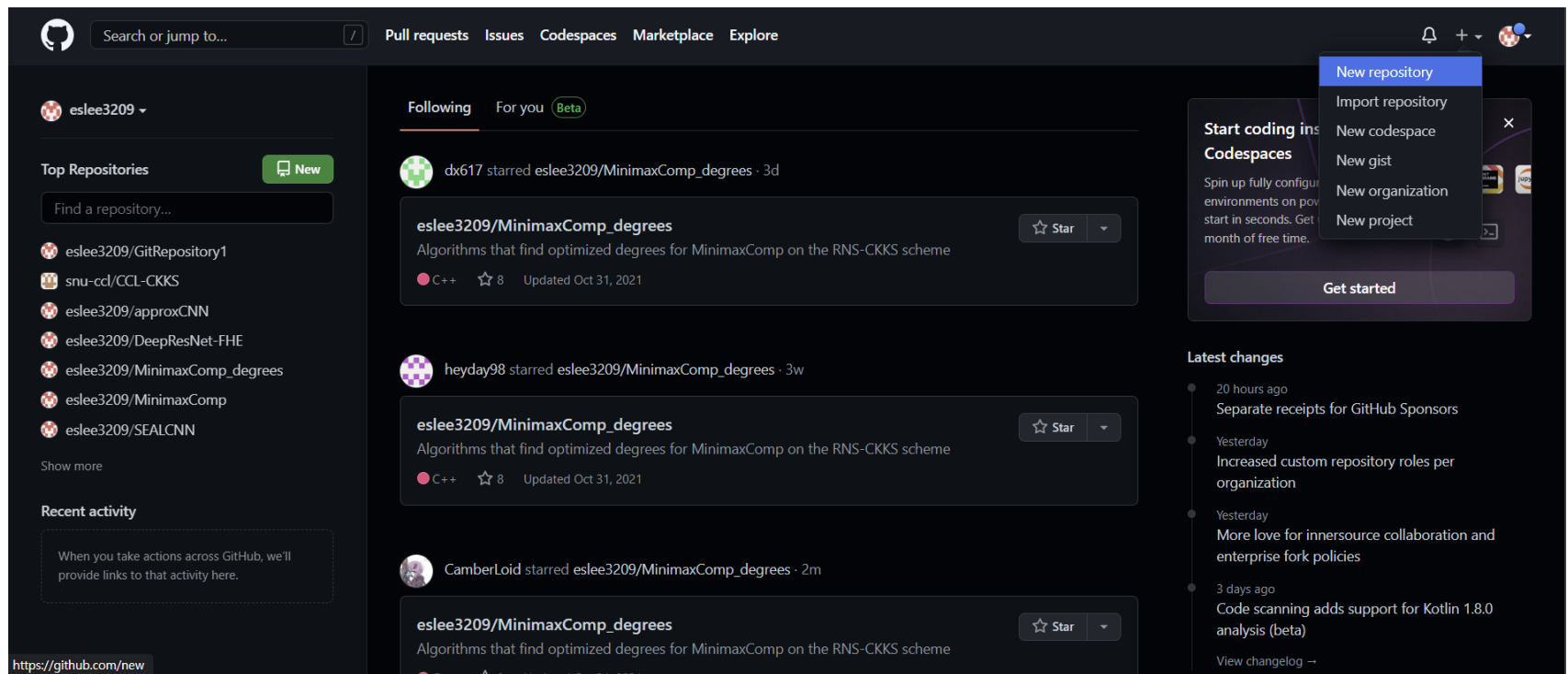
- GitFile1.txt를 열어보면 "Contents2"가 들어있음

원격저장소 실습

- 원격저장소 개념
 - 로컬 저장소만 있어도 혼자서 버전관리할 수는 있음
 - 그러나 다른 개발자와 함께 버전 관리하려면 원격 저장소에 올려야함
 - Github에 원격저장소를 만드는데 이는 github 웹사이트에 협업 프로젝트를 위한 공용 폴더를 만드는 것임
 - Github에서는 이 원격저장소를 레포지토리(repository)라고 함
 - 로컬저장소를 업로드하는 곳이라고도 할 수 있음

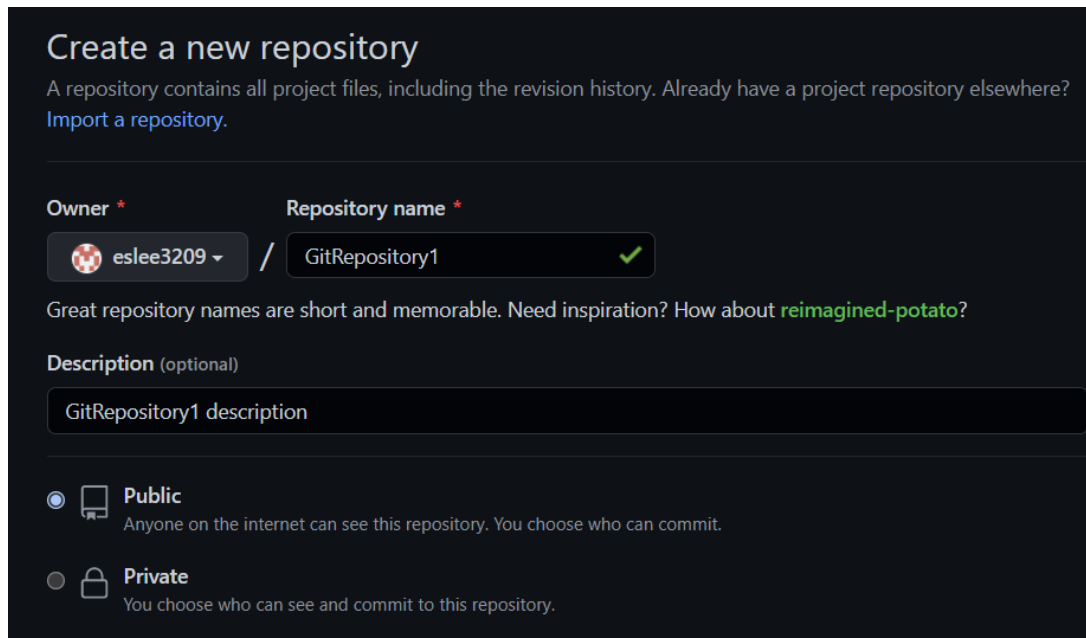
원격저장소 실습

- 원격저장소 생성 방법
 - <https://github.com> 에 접속하고 로그인함
 - 우측 상단 [+] 아이콘을 클릭한 후 [New repository] 메뉴를 선택



원격저장소 실습

- 원격저장소 생성 방법
 - [Repository name]에 원격저장소 이름을 입력함
 - 예시) GitRepository1
 - [Description]에는 원격저장소에 대한 간단한 설명을 적는다.
 - 예시) GitRepository1 description
 - 그 후, [Create repository] 버튼을 클릭함. 그러면 GitRepository1 원격저장소가 생성됨

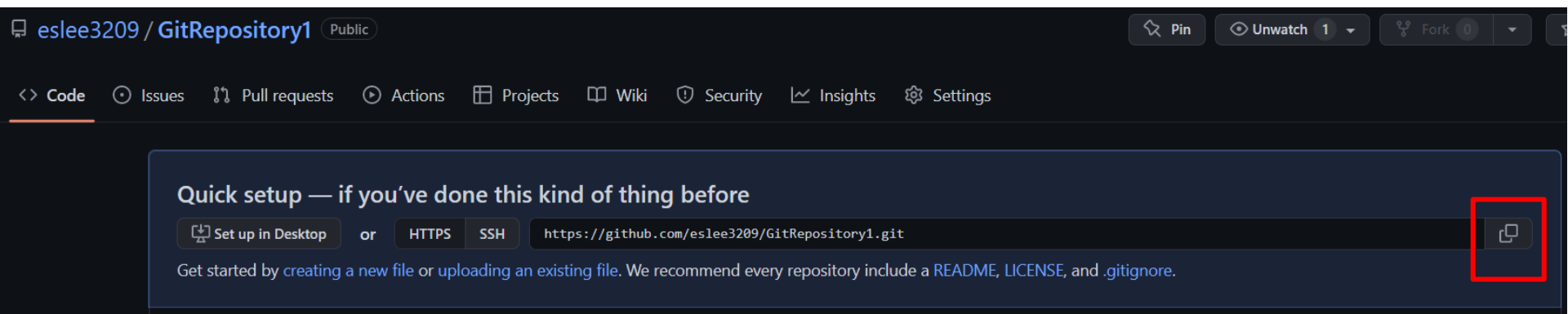


The screenshot shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main input fields: 'Owner' and 'Repository name'. The 'Owner' field is a dropdown menu showing 'eslee3209' with a profile picture icon. The 'Repository name' field contains 'GitRepository1' and has a green checkmark icon to its right. Below these fields, there is a suggestion: 'Great repository names are short and memorable. Need inspiration? How about reimaged-potato?'. Underneath, there is a 'Description (optional)' field with the text 'GitRepository1 description'. At the bottom, there are two radio button options for repository visibility: 'Public' (selected) and 'Private'. The 'Public' option is accompanied by a globe icon and the text 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is accompanied by a lock icon and the text 'You choose who can see and commit to this repository.'

원격저장소 실습

- 원격저장소 주소

- 아래와 같은 형태로 원격저장소 주소가 나옴. 실제 실습 시 "eslee3209" 대신 자신의 username을 넣어주어야함
 - `https://github.com/eslee3209/GitRepository1.git`
- 이 주소를 통해 원격저장소에 접속할 수 있고, 다른 개발자와 함께 작업할 때 이 주소를 알려주면 됨
- 아래 우측 복사 아이콘을 클릭하면 주소를 복사함



원격저장소 실습

- Push란?
 - 로컬저장소에 만들었던 커밋들을 원격저장소에 올리는 것임. 즉, push하는 것임

원격저장소 실습

- Push 방법

- 먼저 로컬저장소에게 원격저장소 주소를 알려주어야 함. 이 과정은 한번만 수행하면 이후 push할 때는 수행할 필요 없음
- Git bash에서 아래 명령어를 수행하여 주소를 알려줌
 - git remote add origin [원격저장소 주소]
 - 예) git remote add origin <https://github.com/eslee3209/GitRepository1.git>

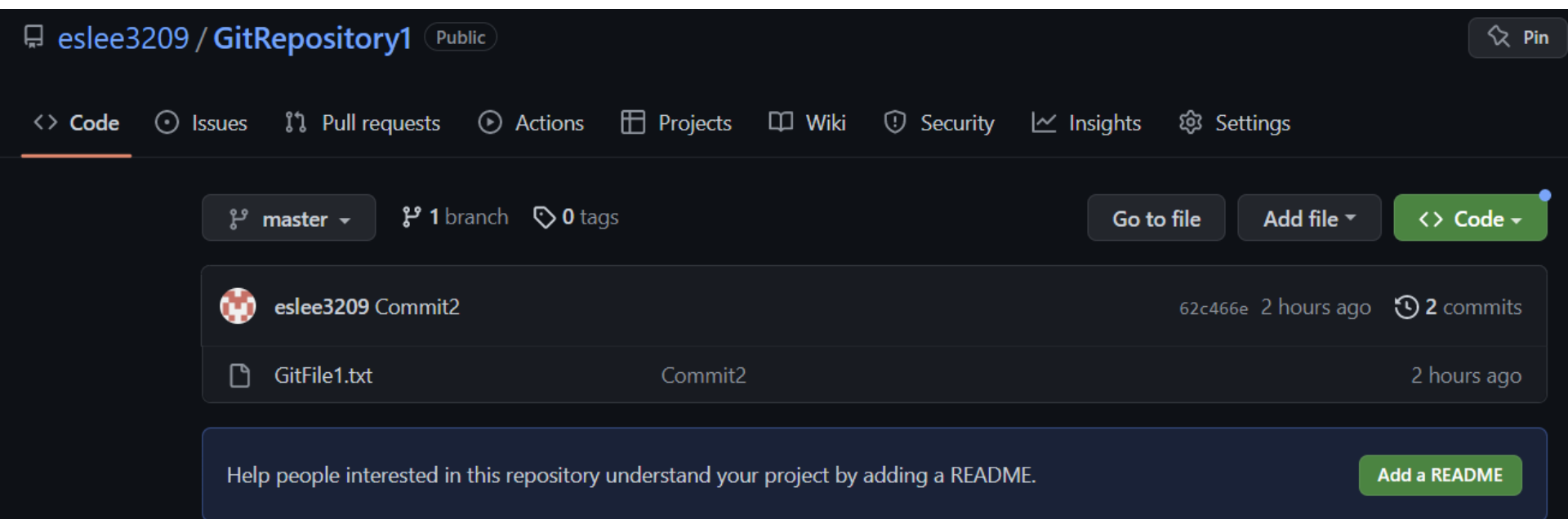
```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git remote add origin https://github.com/eslee3209/GitRepository1.git
```

- 이후 아래 명령어로 push를 수행함. 100% 완료되었다는 텍스트가 뜨면 성공적으로 push된 것임
 - git push origin master

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 434 bytes | 434.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/eslee3209/GitRepository1.git
 * [new branch]      master -> master
```

원격저장소 실습

- Push 결과 확인
 - Github 해당 원격저장소에 들어가면 로컬저장소의 커밋들이 원격저장소에도 잘 반영된 것을 확인할 수 있음

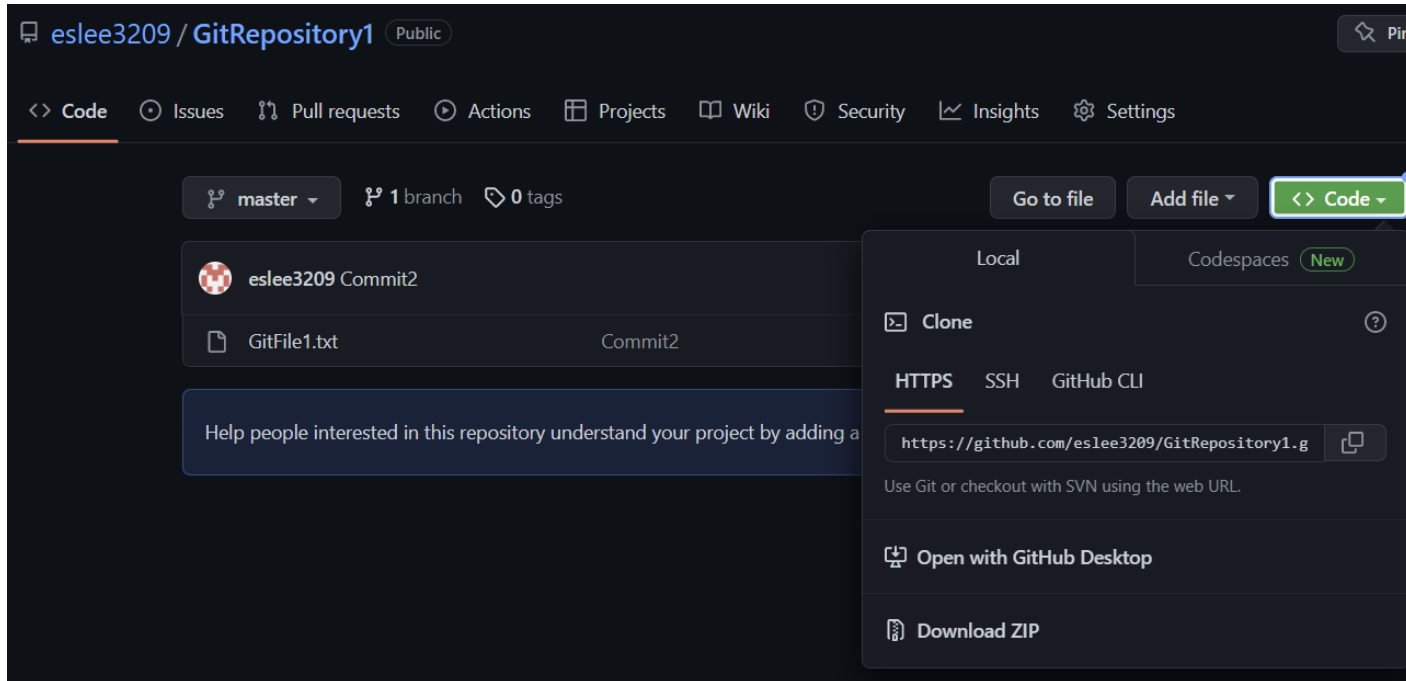


원격저장소 실습

- Clone이란?
 - 원격저장소의 코드와 버전 전체를 내 로컬 컴퓨터로 내려받는 것임
 - 최신 버전 뿐 아니라 이전 버전들과 원격저장소 주소 등이 내 로컬저장소에 저장됨

원격저장소 실습

- Clone 방법
 - 클론할 위치의 폴더를 내 로컬 컴퓨터에 생성함
 - 예) C:\GitDirectory2
 - 그 폴더에서 git bash를 엽
 - Github의 원격저장소에 들어가 [Code] 버튼을 클릭하고 원격저장소 주소를 복사함



원격저장소 실습

- Clone 방법

- 다음 명령어를 입력함.

- git clone [원격저장소 주소] .

- 예) git clone https://github.com/eslee3209/GitRepository1.git .

- 마지막에 한 칸 띄고 마침표를 찍어줌. 현재 폴더에 받으라는 의미임. 점을 찍지 않으면 현재 폴더 안에 GitRepository1 폴더가 생김

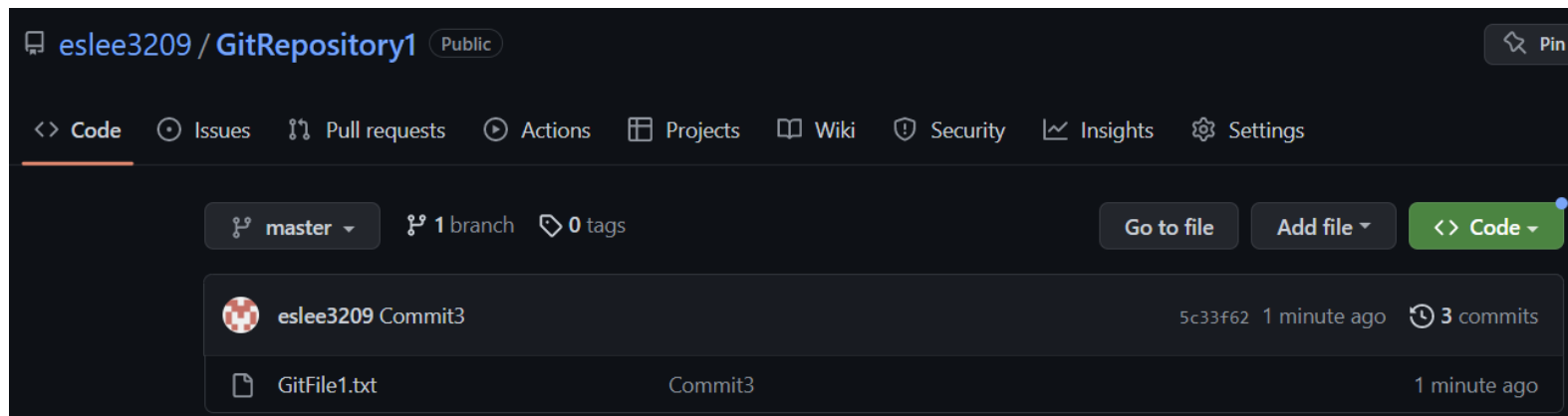
```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory2
$ git clone https://github.com/eslee3209/GitRepository1.git .
Cloning into '.'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

*** [Download Zip] 버튼으로 소스코드를 받을 수도 있지만 원격저장소와 버전 정보가 제외되니 clone하는 것을 권장함

- 클론 결과 해당 폴더(GitDirectory2)에 GitFile1.txt가 생성됨을 알 수 있음. 즉, 정상적으로 클론이 됨

원격저장소 실습

- Push
 - 새로 만든 폴더(GitDirectory2)에서도 push를 진행할 수 있음
 - 파일을 수정함
 - 예) GitFile1.txt에서 내용을 Contents2 -> Contents3로 수정함
 - 다음 명령어들을 입력함
 - `git add GitFile1.txt`
 - `git commit -m "Commit3"`
 - `git push origin master`
 - 이후 github에서 확인해보면 세 번째 커밋이 잘 반영된 것을 알 수 있음



원격저장소 실습

- Pull 개념

- 원격저장소는 최근 커밋 업데이트가 되었지만 로컬저장소에는 커밋이 반영되지 않은 경우가 있음
- 이 경우 pull을 하여 원격저장소의 새로운 커밋을 로컬저장소에 갱신해야함

- Pull 실습

- 로컬저장소 GitDirectory1에 들어감
- 로컬저장소 GitDirectory1에서는 아직 원격저장소의 세 번째 커밋이 반영되어있지 않음
 - 즉, GitFile1.txt 내용이 Contents2 (Contents3이 아니라)로 되어있음
- GitDirectory1 폴더에서 git bash를 열고 아래 명령어를 입력함
 - git pull origin master

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 228 bytes | 16.00 KiB/s, done.
From https://github.com/eslee3209/GitRepository1
* branch      master      -> FETCH_HEAD
62c466e..5c33f62 master    -> origin/master
Updating 62c466e..5c33f62
Fast-forward
 GitFile1.txt | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
```

원격저장소 실습

- Pull 실습
 - 그러면 GitDirectory1의 GitFile1.txt 내용이 "Contents3"으로 바뀌어있음
 - 즉, 원격저장소 최근 커밋을 로컬저장소로 가져오는 pull이 정상적으로 수행된 것을 알 수 있음