

Git & GitHub

- ◆ Git 시작하기

정수아

Contents

01 Git이란 무엇인가

02 Git 프로그램의 종류

03 Git 설치하기

04 Git 환경 설정

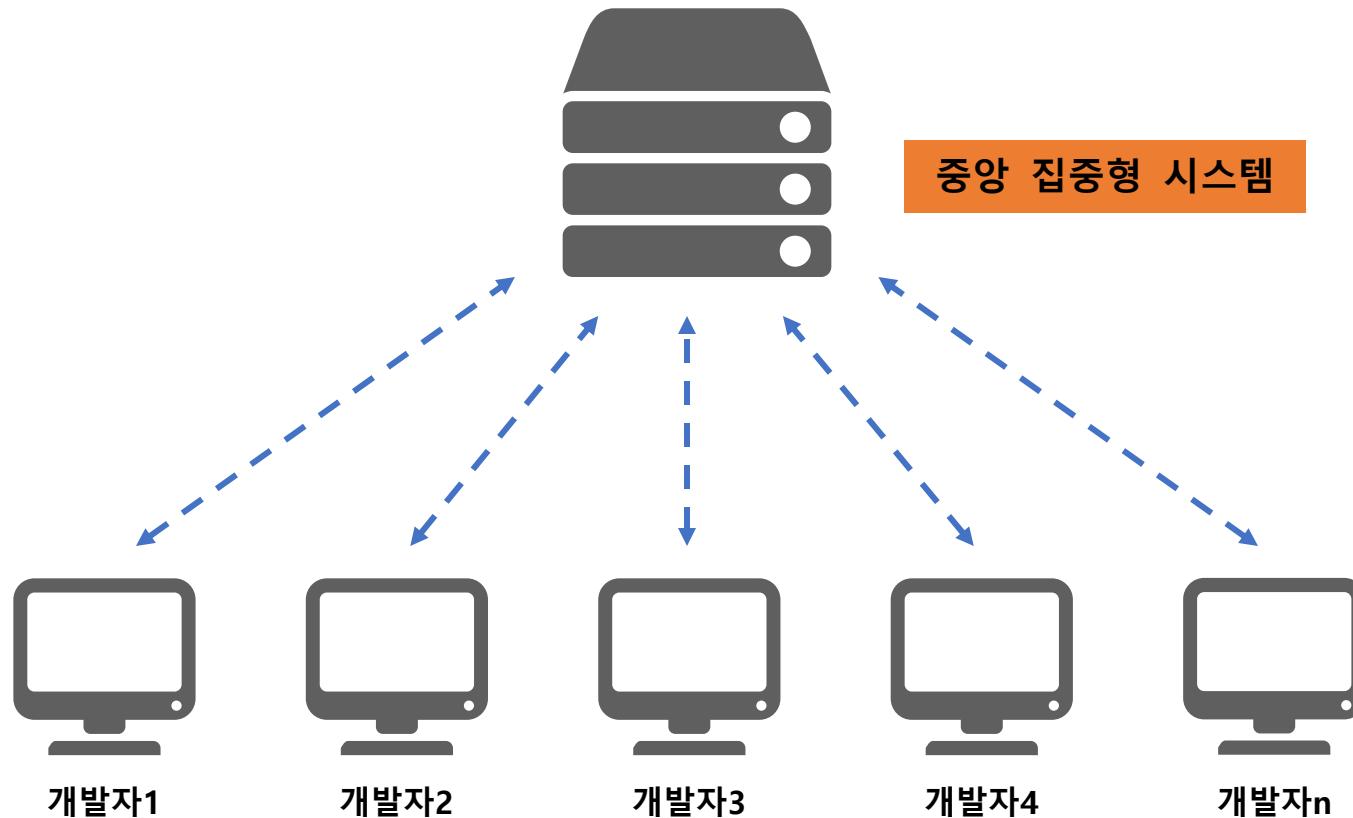
05 기본 리눅스 명령어

06 빔(Vim) 사용하기

01

Git이란 무엇인가

Git의 탄생



Git의 탄생

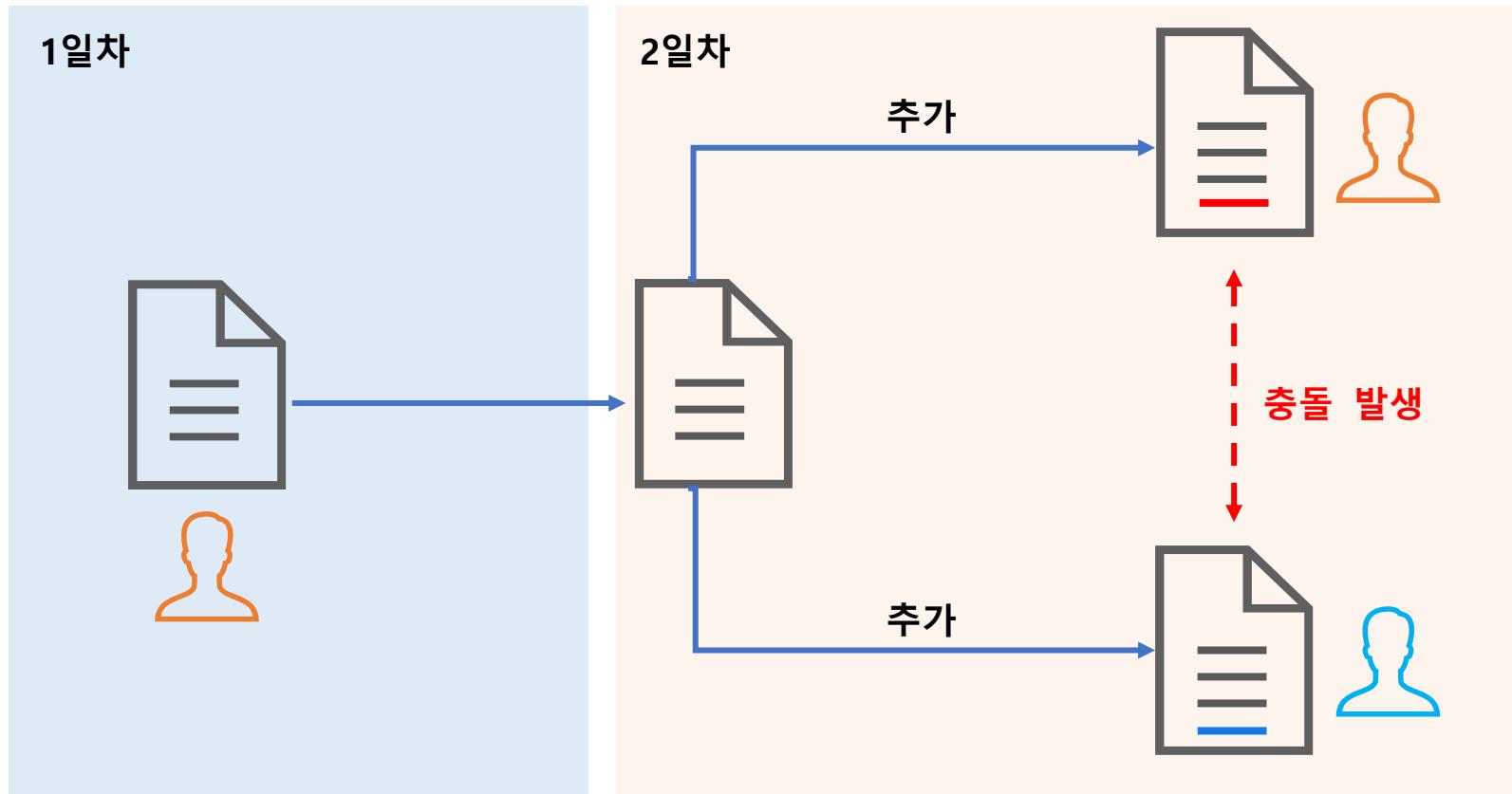
❖ Git

- 2005년 리누스 토르발스가 처음 개발
- 특징
 - 소스 코드의 변경사항을 관리(버전 관리)
 - 분산 버전 관리 시스템
- 개발자들은 Git을 통해 수많은 소스 코드를 효율적으로 관리



버전 관리

❖ 버전 관리의 필요성



버전 관리

❖ 버전 관리

- 코드나 문서의 변경 이력을 기록하고, 이전 상태로 되돌릴 수 있는 시스템
- 개인 또는 팀 프로젝트에서 안전하고 효율적인 작업을 가능하게 함

❖ 특징

- 변경 이력 추적
- 복구 및 롤백
- 협업 지원
- 안전한 백업

Git의 특징

- ❖ 버전 관리(Version Control)
- ❖ 백업(Backup)
- ❖ 협업(Collaboration)

Git의 특징

❖ 버전 관리(Version Control)

- 문서를 작성한 뒤 원본을 남겨두고 수정한 내용을 저장해야 하는 경우
- '다른 이름으로 저장'을 주로 사용



원본



수정



최종



진짜최종

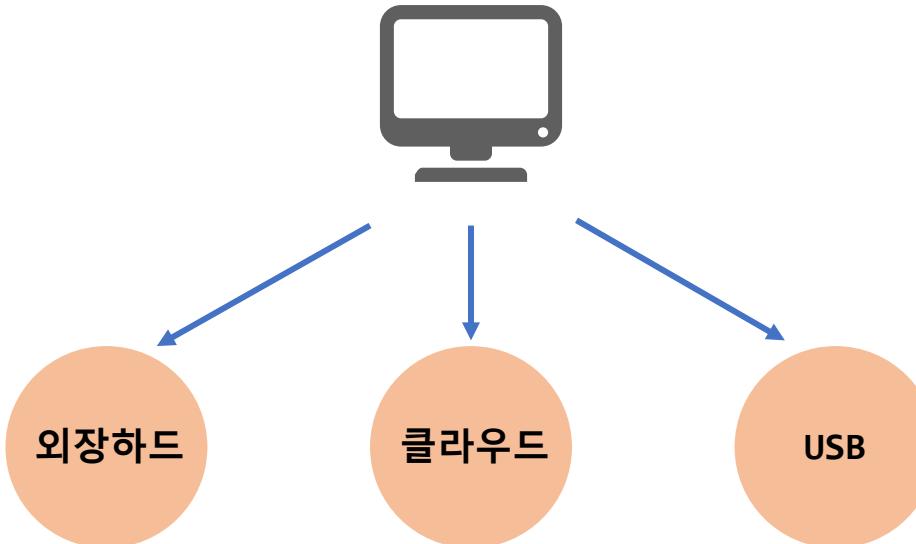
☆ Git을 사용한다면? ☆

파일 이름은 유지하면서 문서를 수정할 때마다 언제 수정했는지,
어떤 것이 변경되었는지 기록 가능

Git의 특징

❖ 백업(Backup)

- 백업이란?
 - 현재 내 컴퓨터에 있는 자료를 다른 곳에 복제



☆ GitHub ☆

Git으로 관리한 파일을 위한 원격 저장소 또는 온라인 저장소

Git의 특징

❖ 협업(Collaboration)

- GitHub와 같은 온라인 서비스를 사용하면 여러 사람이 함께 일할 수 있음



☆ GitHub의 장점 ☆

- 팀원들이 파일을 편하게 주고받으면서 일할 수 있음
- 누가 언제, 어떤 부분을 수정했는지 기록에 남길 수 있음

Git의 특징

❖ 버전 관리(Version Control)

- 문서를 수정할 때마다 언제 수정했는지, 어떤 것을 변경했는지 편하고 구체적으로 기록하기 위한 버전 관리 시스템

❖ 백업(Backup)

- 현재 컴퓨터의 자료를 인터넷 상의 공간(원격 저장소)에 백업
- 주로 GitHub 사용

❖ 협업(Collaboration)

- 원격 저장소를 통해 여러 사람이 함께 일할 수 있음
- 누가, 언제, 어느 부분을 수정했는지 기록 가능

02

Git 프로그램의 종류

Git 프로그램

❖ GitHub 데스크톱(GitHub Desktop)

- GitHub에서 제공하는 프로그램
- 그래픽 사용자 인터페이스(GUI)로 구현
- 사용이 쉬워 누구나 배울 수 있음
- 기본적인 기능만 제공

❖ 토터스Git(TortoiseGit)

- 윈도우 전용 프로그램
- 윈도우 탐색기의 빠른 메뉴에 추가되는 프로그램

❖ 소스트리(Source Tree)

- Git의 기본 기능부터 고급 기능까지 제공
- 사용법은 복잡하지만 익숙해지면 자유롭게 Git을 활용할 수 있음

Git 프로그램

❖ 커맨드 라인 인터페이스(CLI)

- 터미널에 직접 명령을 입력해서 Git을 사용하는 방식
- 리눅스 명령어와 Git 명령어를 알아야 사용 가능
- 대부분의 개발자들은 이 방식으로 Git을 사용

Git 프로그램

❖ 더 많은 Git 프로그램

- <https://git-scm.com/downloads/guis>

GUI Clients

Git comes with built-in GUI tools for committing ([git-gui](#)) and browsing ([gitk](#)), but there are several third-party tools for users looking for platform-specific experience.

If you want to add another GUI tool to this list, just [follow the instructions](#).

All Windows Mac Linux Android iOS

GitHub Desktop
Platforms: Mac, Windows
Price: Free
License: MIT

SourceTree
Platforms: Mac, Windows
Price: Free
License: Proprietary

TortoiseGit
Platforms: Windows
Price: Free
License: GNU GPL

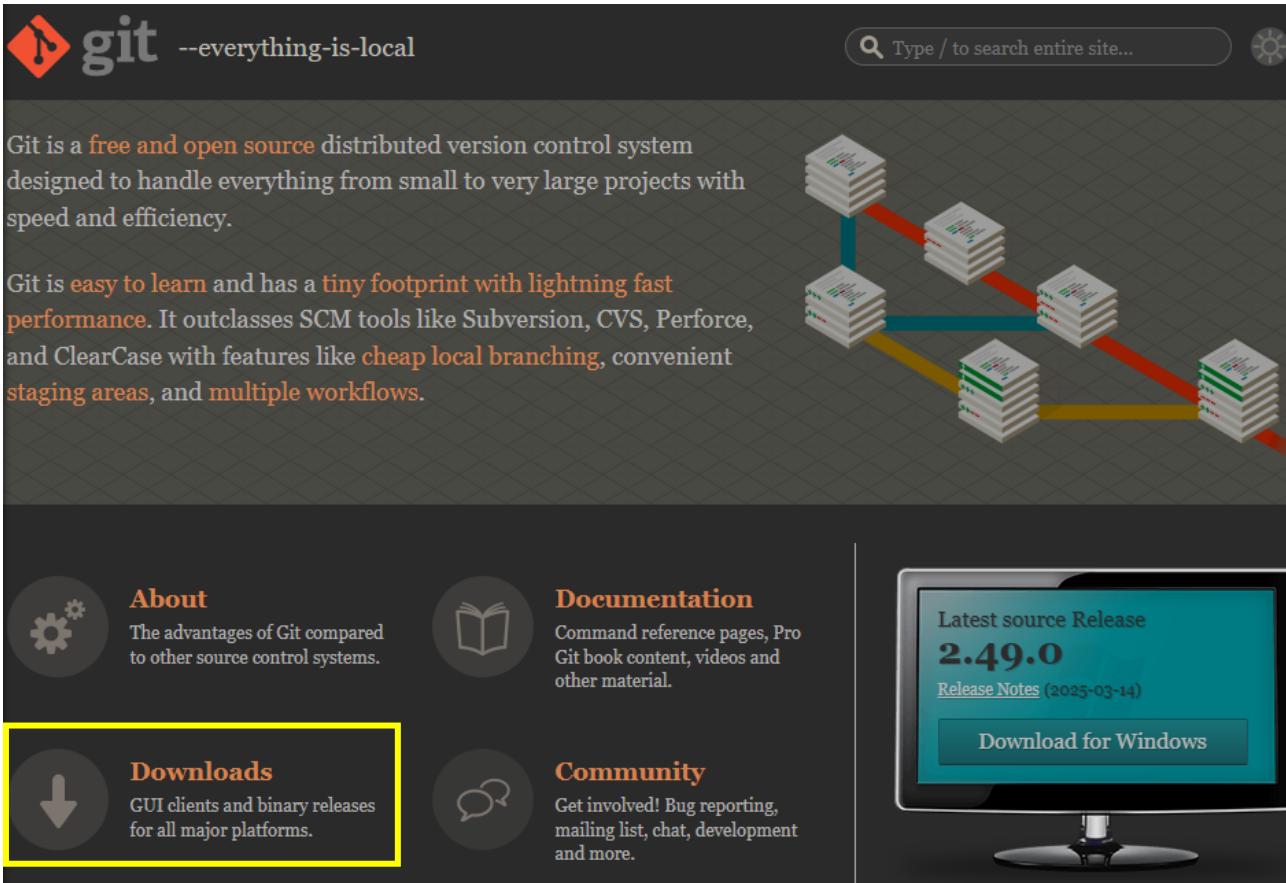
Git Extensions
Platforms: Linux, Mac, Windows
Price: Free
License: GNU GPL

03

Git 설치하기

Git 설치하기 - Windows

❖ <https://git-scm.com>



The screenshot shows the official Git website (<https://git-scm.com>). At the top, there's a navigation bar with the Git logo and the tagline "git --everything-is-local". A search bar and a settings gear icon are also present. The main content area features a dark background with a grid pattern. On the left, there are two columns of text: one about Git's design philosophy and another about its performance and features. To the right of the text is a diagram illustrating Git's distributed nature with multiple repositories connected by bidirectional arrows. Below this, there are four circular icons with text labels: "About", "Documentation", "Downloads", and "Community". The "Downloads" section is highlighted with a yellow border. On the right side, there's a large monitor icon displaying the latest source release information, including the version number "2.49.0" and a "Download for Windows" button.

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint** with **lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, **convenient staging areas**, and **multiple workflows**.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.49.0
[Release Notes](#) (2025-03-14)

[Download for Windows](#)

Git 설치하기

The screenshot shows the official Git website (git-scm.com/) with a dark theme. At the top left is the Git logo and the slogan "distributed-is-the-new-centralized". A search bar and a gear icon are at the top right. On the left sidebar, there are links for About, Documentation, Downloads (which is bolded), GUI Clients, Logos, and Community. A small box on the left sidebar promotes the "Pro Git book". The main content area features a large "Downloads" section with three buttons for macOS, Windows, and Linux/Unix. A yellow box highlights the "Linux/Unix" button. To the right of this is a monitor icon displaying release information for version 2.49.0. Below the download section, there's a note about older releases and a GitHub link. The "GUI Clients" section below has a "View GUI Clients →" link. The "Logos" section on the right has a "View Logos →" link.

git --distributed-is-the-new-centralized

About Documentation Downloads GUI Clients Logos Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloads

macOS Windows Linux/Unix

Older releases are available and the [Git source repository](#) is on GitHub.

GUI Clients

Git comes with built-in GUI tools (`git-gui`, `gitk`), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the web interface.

Latest source Release
2.49.0
[Release Notes](#) (2025-03-14)
[Download for Windows](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

Git 설치하기 - Windows

The screenshot shows the official Git for Windows website. On the left sidebar, there's a navigation menu with links for About, Documentation, Downloads (which includes GUI Clients and Logos), and Community. A callout box highlights the "Pro Git book" information. The main content area features a large heading "Download for Windows". Below it, a paragraph encourages users to click to download the latest version. A yellow box highlights the "Standalone Installer" and "Git for Windows/x64 Setup" links. Further down, it lists "Portable" options and "Using winget tool" instructions, including a command-line example and a note about building from source code.

git --fast-version-control

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Download for Windows

[Click here to download](#) the latest (2.49.0) x64 version of Git for Windows. This is the most recent maintained build. It was released [almost 2 months ago](#), on 2025-03-17.

Other Git for Windows downloads

[Standalone Installer](#)
[Git for Windows/x64 Setup.](#)

[Git for Windows/ARM64 Setup.](#)

[Portable \("thumbdrive edition"\)](#)
[Git for Windows/x64 Portable.](#)

[Git for Windows/ARM64 Portable.](#)

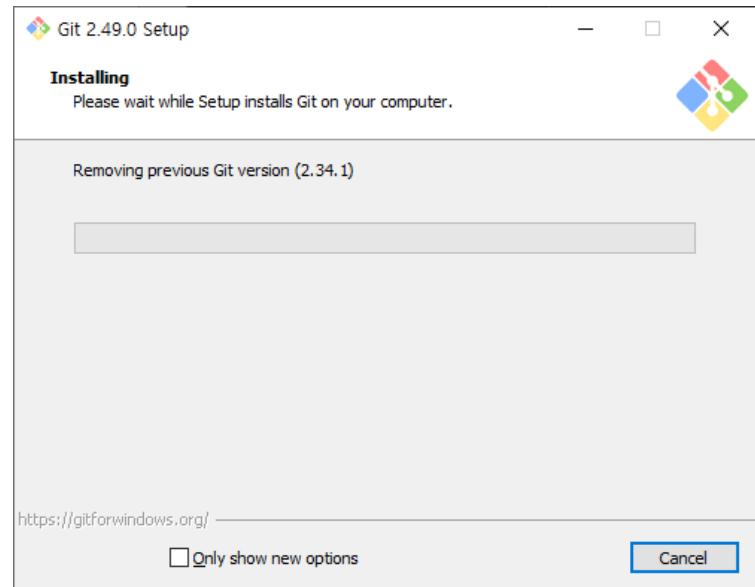
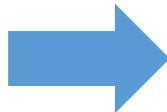
Using winget tool

Install `winget` tool if you don't already have it, then type this command in command prompt or Powershell.

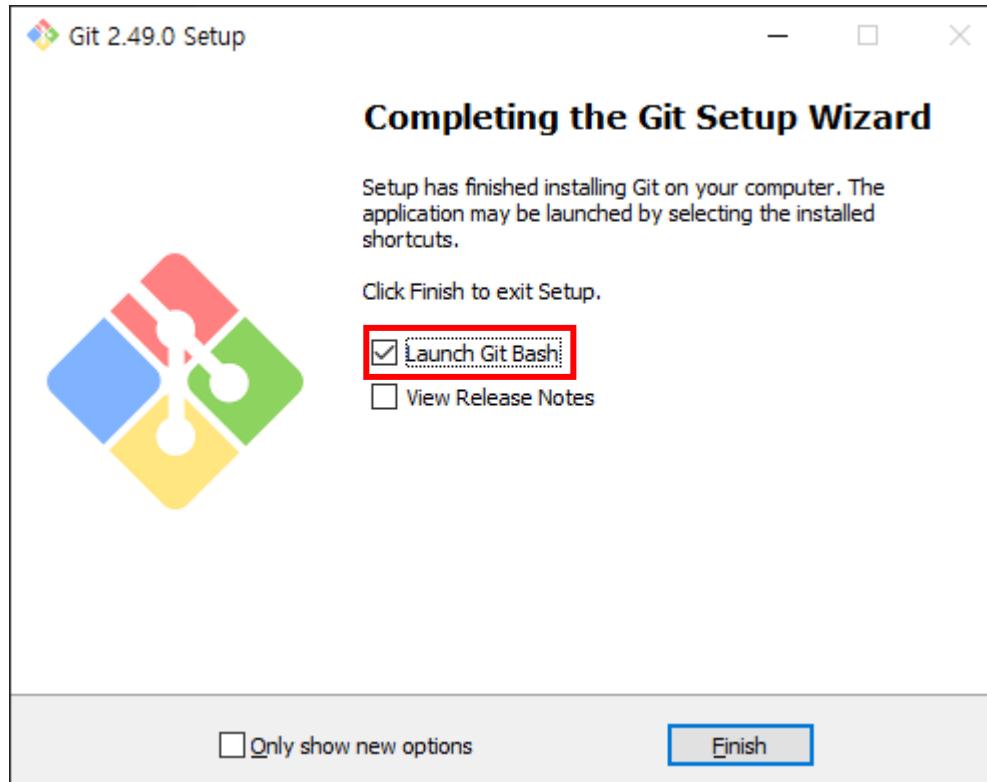
```
winget install --id Git.Git -e --source winget
```

The current source code release is version 2.49.0. If you want the newer version, you can build it from [the source code](#).

Git 설치하기 - Windows

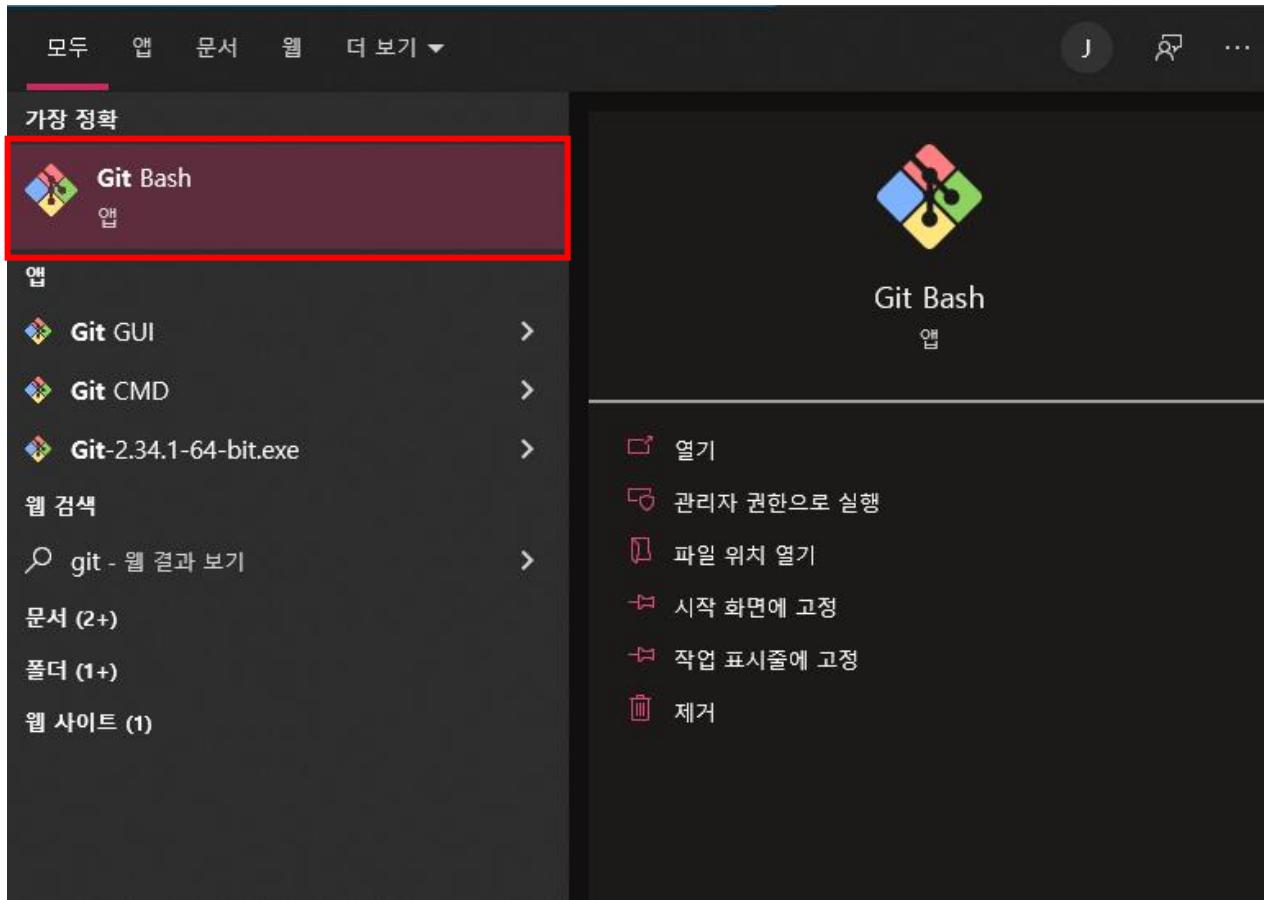


Git 설치하기 - Windows



Git 설치하기 - Windows

- ❖ 윈도우 검색 창에 git 입력 후, [Git Bash] 클릭



Git 설치하기 - macOS

❖ homebrew 설치하기

- <https://brew.sh/>

```
$ /bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

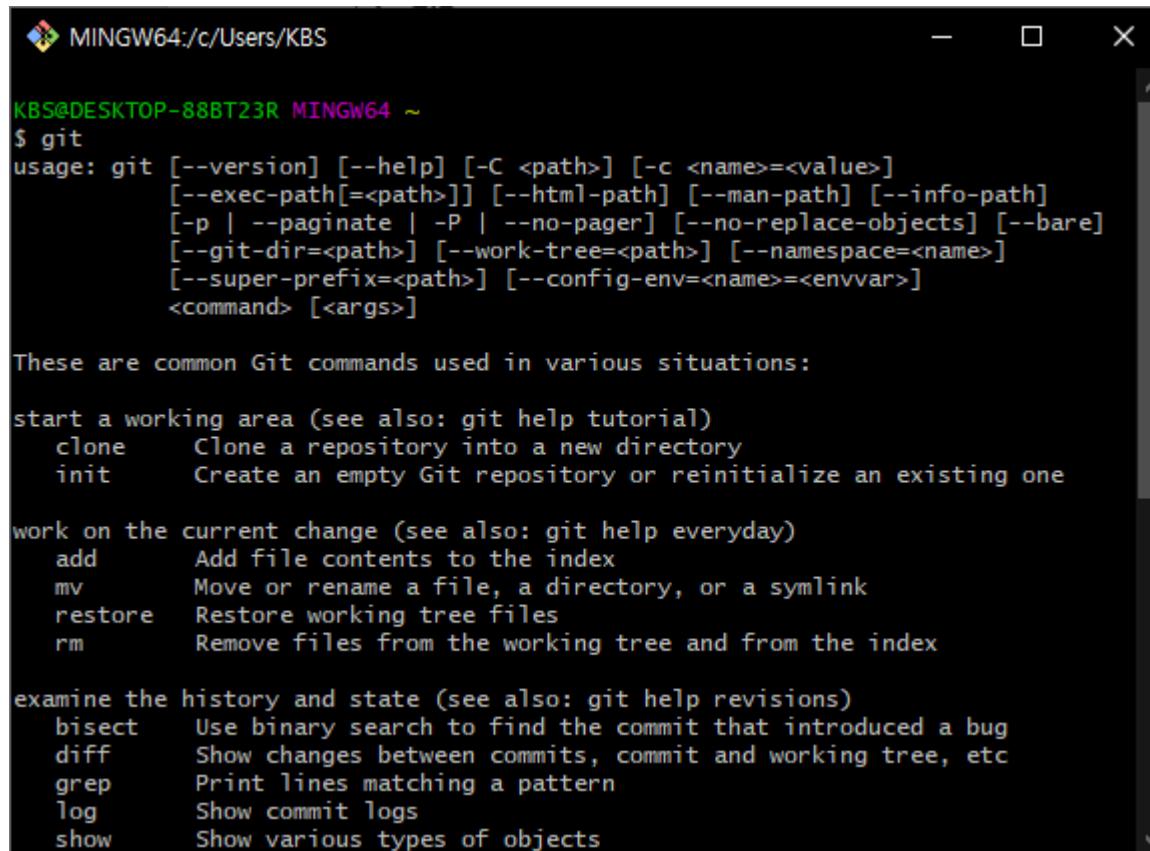
❖ Git 설치하기

```
$ brew install git
```

Git 설치하기

- ❖ Git Bash 화면에 다음 명령어 입력

```
$ git
```



```
MINGW64:/c/Users/KBS
KBS@DESKTOP-88BT23R MINGW64 ~
$ git
usage: git [--version] [--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

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

환경 설정

❖ 사용자 정보 입력

- 사용자 이름

```
$ git config --global user.name "Sooa"
```

- 사용자 이메일

```
$ git config --global user.email "Sooa@gmail.com"
```

05

기본 리눅스 명령어

리눅스 명령어

❖ 현재 디렉터리 위치 경로

```
$ pwd
```



A screenshot of a terminal window titled "MINGW64:/c/Users/KBS". The window shows the command "pwd" being run and its output "/c/Users/KBS". The terminal has a dark theme with white text and a black background.

```
MINGW64:/c/Users/KBS
KBS@DESKTOP-88BT23R MINGW64 ~
$ pwd
/c/Users/KBS
```

리눅스 명령어

- ❖ 현재 디렉터리에 안에 있는 파일 또는 디렉터리 확인

```
$ ls
```



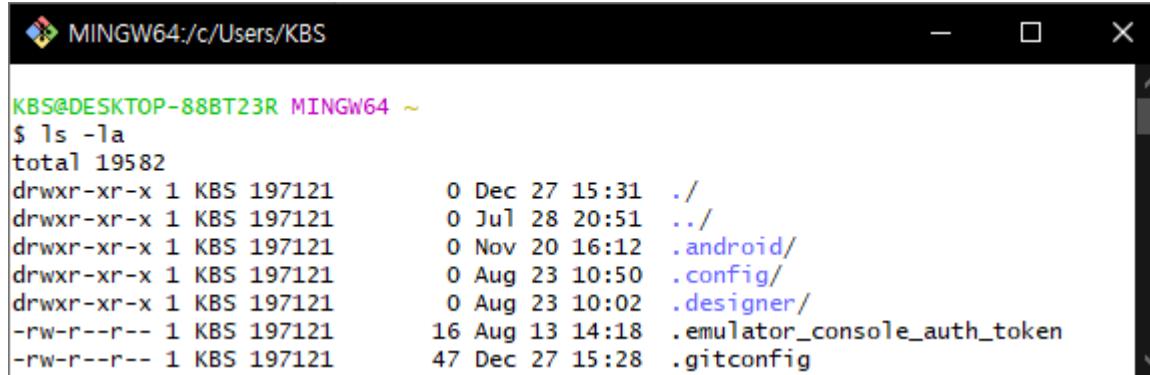
A screenshot of a terminal window titled "MINGW64:/c/Users/KBS". The window shows the command \$ ls followed by a list of directory names: '3D Objects', 'AndroidStudioProjects', 'AppData', 'Application Data', 'Contacts', 'Cookies', 'Desktop', and 'Documents'. The terminal has a dark theme with white text and a black background.

```
MINGW64:/c/Users/KBS
$ ls
'3D Objects'/
AndroidStudioProjects/
AppData/
'Application Data'@
Contacts/
Cookies@
Desktop/
Documents/
```

리눅스 명령어

❖ 파일과 디렉터리 상세 정보 확인

```
$ ls -la
```



The screenshot shows a terminal window titled "MINGW64:/c/Users/KBS". The command \$ ls -la is entered, followed by its output. The output shows a directory listing with detailed file information. The files listed include ./, ../, .android/, .config/, .designer/, .emulator_console_auth_token, and .gitconfig. The listing includes columns for permissions, owner, group, size, date, and name.

```
KBS@DESKTOP-88BT23R MINGW64 ~
$ ls -la
total 19582
drwxr-xr-x 1 KBS 197121      0 Dec 27 15:31 .
drwxr-xr-x 1 KBS 197121      0 Jul 28 20:51 ..
drwxr-xr-x 1 KBS 197121      0 Nov 20 16:12 .android/
drwxr-xr-x 1 KBS 197121      0 Aug 23 10:50 .config/
drwxr-xr-x 1 KBS 197121      0 Aug 23 10:02 .designer/
-rw-r--r-- 1 KBS 197121    16 Aug 13 14:18 .emulator_console_auth_token
-rw-r--r-- 1 KBS 197121   47 Dec 27 15:28 .gitconfig
```

리눅스 명령어

❖ ls 명령어 옵션

옵션	설명
-a	숨김 파일과 디렉터리를 함께 표시
-l	파일이나 디렉터리의 상세 정보를 함께 표시
-r	파일의 정렬 순서를 거꾸로 표시
-t	파일 작성 시간 순으로(내림차순) 표시

리눅스 명령어

❖ 화면 지우기

```
$ clear
```



리눅스 명령어

❖ 디렉터리 이동

```
$ cd
```

- 현재 위치에서 상위 디렉터리로 이동

```
$ cd ..
```

한 번 더 명령어 입력

```
MINGW64:/c/Users
KBS@DESKTOP-88BT23R MINGW64 ~
$ cd ..
KBS@DESKTOP-88BT23R MINGW64 /c/Users
$
```



```
MINGW64:/c
$ cd ..
KBS@DESKTOP-88BT23R MINGW64 /c/Users
$ cd ..
KBS@DESKTOP-88BT23R MINGW64 /c
$
```

리눅스 명령어

❖ 디렉터리 이동

- 현재 위치에서 하위 디렉터리로 이동

```
$ cd 하위_디렉터리_이름
```

- /c/Users 디렉터리로 이동

```
$ cd Users
```



The screenshot shows a terminal window titled "MINGW64:/c/Users". The window contains the following text:

```
KBS@DESKTOP-88BT23R MINGW64 /c
$ cd Users
KBS@DESKTOP-88BT23R MINGW64 /c/Users
$
```

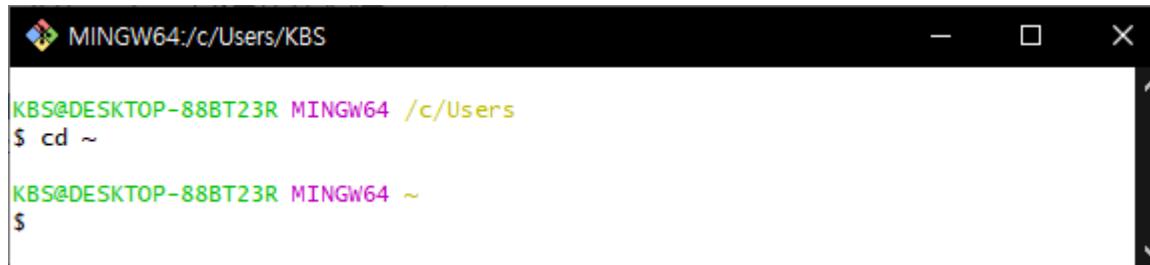
The path "/c/Users" is highlighted with a red rectangle.

리눅스 명령어

❖ 디렉터리 이동

- 홈 디렉터리로 이동

```
$ cd ~
```



A screenshot of a terminal window titled "MINGW64:/c/Users/KBS". The window shows the command \$ cd ~ being run. The terminal output shows the user's name (KBS), the host name (DESKTOP-88BT23R), the shell (MINGW64), the current directory (/c/Users), and the command \$ cd ~. After the command is run, the prompt changes to \$, indicating the user has moved to their home directory.

```
KBS@DESKTOP-88BT23R MINGW64 /c/Users
$ cd ~

KBS@DESKTOP-88BT23R MINGW64 ~
$
```

리눅스 명령어

❖ cd 명령어 옵션

옵션	설명
~	<ul style="list-style-type: none">▪ 현재 사용자의 홈 디렉터리▪ c/Users/계정이름
.	현재 사용자가 작업 중인 디렉터리
..	현재 디렉터리의 상위 디렉터리

리눅스 명령어

❖ 디렉터리 생성

```
$ mkdir 생성할_디렉터리_이름
```

- 홈 디렉터리 안에 있는 Documents 디렉터리에 test 디렉터리 생성

```
$ cd ~/Documents  
$ mkdir test
```

리눅스 명령어

❖ 디렉터리 생성

- test 디렉터리 생성 확인

```
$ ls
```



```
MINGW64:/c/Users/KBS/Documents
KBS@DESKTOP-88BT23R MINGW64 ~/Documents
$ ls
Embarcadero/   'My Videos'@      desktop.ini
HeidiSQL/       'Python Scripts'/
Mobizen/        UltraVNC/
'My Music'@     'Visual Studio 2019'/'    '제작자 지정 Office 서식 파일'/
'My Pictures'@ aa.json           '카카오톡 받은 파일'/
$
```

리눅스 명령어

❖ 디렉터리 삭제

```
$ rm -r 삭제할_디렉터리_이름
```

- 홈 디렉터리 안에 있는 Documents 디렉터리의 test 디렉터리 삭제

```
$ rm -r test  
$ ls
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/Documents'. The user has run the command '\$ rm -r test' to delete the 'test' directory. After the deletion, they run '\$ ls' to list the contents of the 'Documents' folder. The output shows several sub-directories and files: 'Embarcadero', 'HeidiSQL', 'Mobizen', 'My Music', 'My Pictures', 'My Videos', 'Python Scripts', 'UltraVNC', 'Visual Studio 2019', and 'desktop.ini'. There is also a file named 'aa.json'. The Korean text in the listing likely refers to file descriptions or specific file names.

```
KBS@DESKTOP-88BT23R MINGW64 ~/Documents
$ rm -r test

KBS@DESKTOP-88BT23R MINGW64 ~/Documents
$ ls
Embarcadero/      'My Videos'@           desktop.ini
HeidiSQL/         'Python Scripts'@       '사용자 지정 Office 서식 파일'/
Mobizen/          UltraVNC/              '카카오톡 받은 파일'/
'My Music'@       'Visual Studio 2019'/
'My Pictures'@    aa.json
```

06

빔(Vim) 사용하기

빔(Vim) 사용하기

❖ 빔(Vim)이란?

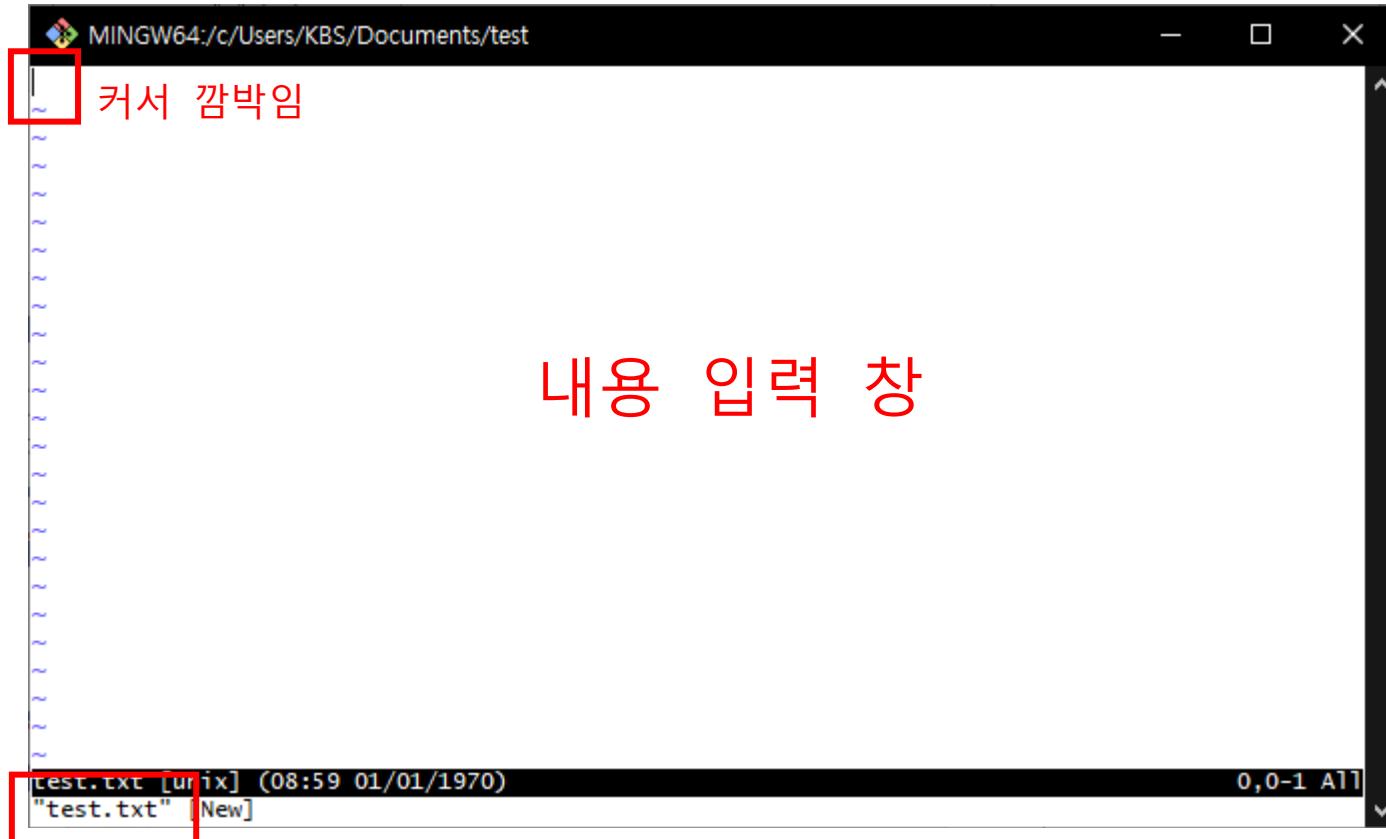
- 터미널에서 사용할 수 있는 대표적인 문서 편집기
- 홈디렉터리/Documents에 test 디렉터리 생성 후 이동

```
$ cd ~/Documents  
$ mkdir test  
$ cd test
```

- 현재 디렉터리(test)에 test.txt 파일 생성

```
$ vim test.txt
```

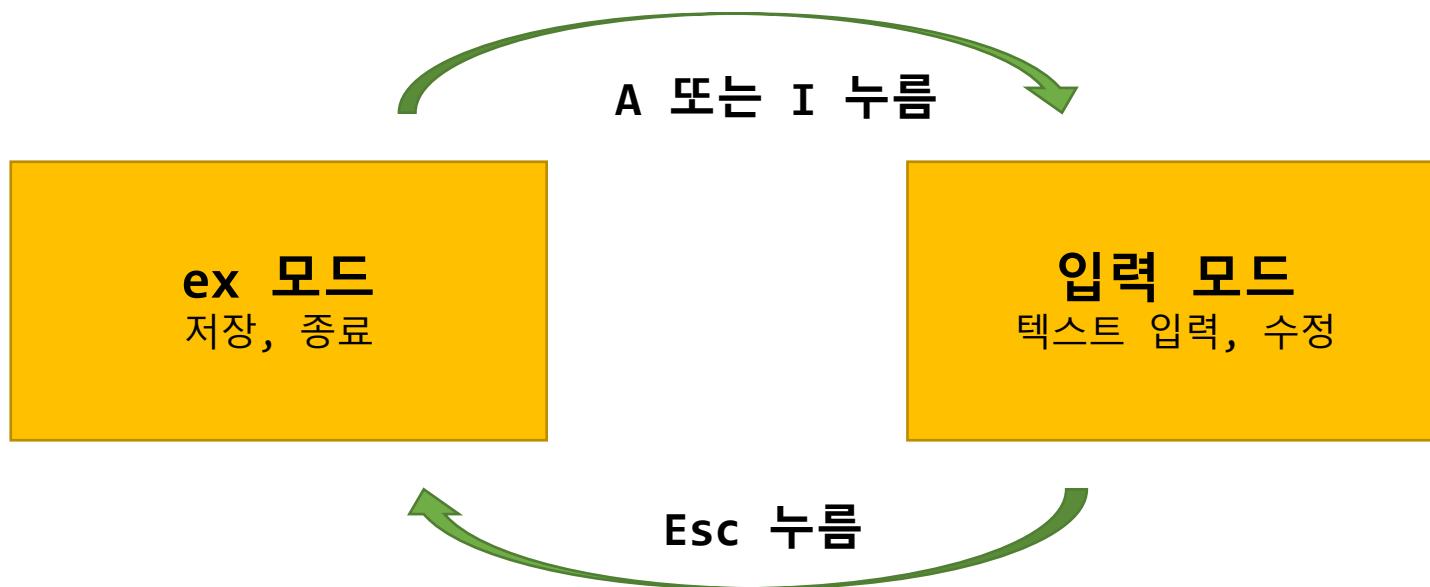
빔(Vim) 사용하기



현재 열려 있는 파일 이름

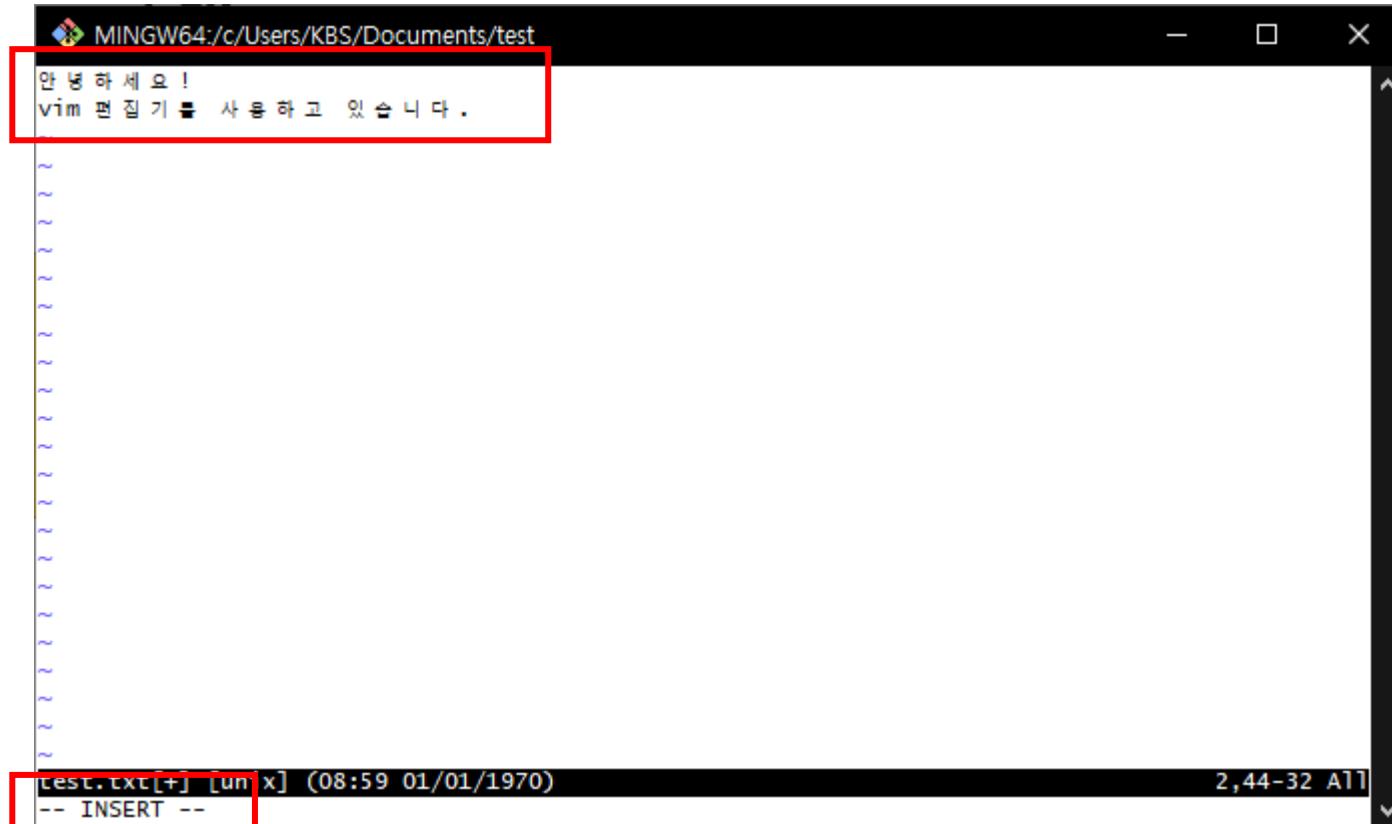
빔(Vim) 사용하기

❖ 입력 모드와 ex 모드



빔(Vim) 사용하기

❖ 텍스트 입력하기



입력 모드

빔(Vim) 사용하기

❖ 내용 저장하기

- ex 모드로 돌아가야 함
 - Esc 키를 누름



빔(Vim) 사용하기

❖ 내용 저장하기

- 콜론(:)을 입력하면 "INSERT"가 있던 자리에 텍스트를 입력할 수 있음
- :wq 입력 후, Enter 키 누름

The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/Documents/test'. The window contains Korean text: '안녕하세요!', 'vim 편집기를 사용하고 있습니다.', and a series of approximately 20 tilde (~) characters. In the bottom status bar, it says 'test.txt[.] [un]x (08:59 01/01/1970)' and '2,43-31 All'. A red box highlights the command ':wq' in the bottom-left corner of the terminal window.

빔(Vim) 사용하기

❖ ex 모드 명령어

옵션	설명
:w 또는 :write	편집 중이던 문서를 저장
:q 또는 :quit	편집기를 종료
:wq	편집 중이던 문서를 저장하고 종료
:q!	문서를 저장하지 않고 편집기를 종료

빔(Vim) 사용하기

❖ 텍스트 문서 내용 확인하기

```
$ cat 파일이름
```

- test.txt 파일 내용 확인

```
$ cat test.txt
```



The screenshot shows a terminal window with a black background and white text. The title bar reads "MINGW64:/c/Users/KBS/Documents/test". The command \$ cat test.txt is entered, followed by the contents of the file: "안녕하세요!" and "vim 편집기를 사용하고 있습니다.". The terminal has standard window controls (minimize, maximize, close) and scroll bars.

```
KBS@DESKTOP-88BT23R MINGW64 ~/Documents/test
$ cat test.txt
안녕하세요!
vim 편집기를 사용하고 있습니다.
```

Git & GitHub

- ◆ Git으로 버전 관리하기

정수아

Contents

01 Git 저장소 만들기

02 버전 만들기

03 커밋 내용 확인하기

04 .gitignore

05 단계마다 파일 상태 알아보기

06 작업 되돌리기

07 특정 커밋으로 되돌리기

01

Git 저장소 만들기

Git 저장소 만들기

- ❖ hello-git 디렉터리 생성 후 이동

```
$ mkdir hello-git  
$ cd hello-git
```



A screenshot of a terminal window titled "MINGW64:/c/Users/KBS/hello-git". The window shows two command-line entries: "\$ mkdir hello-git" and "\$ cd hello-git". The terminal has a dark background with light-colored text. The title bar is black with white text.

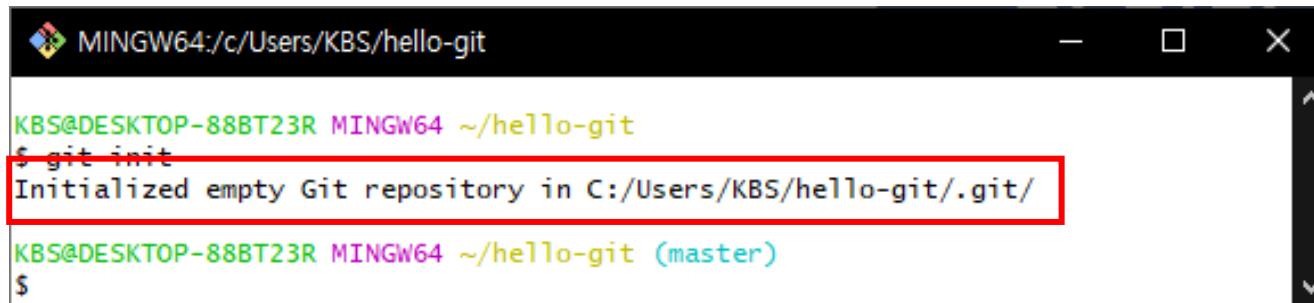
```
KBS@DESKTOP-88BT23R MINGW64 ~  
$ mkdir hello-git  
  
KBS@DESKTOP-88BT23R MINGW64 ~  
$ cd hello-git
```

Git 저장소 만들기

❖ Git 초기화 하기

- 디렉터리를 Git 저장소로 사용할 수 있도록 만들어 줌
- 초기화 후 .git 디렉터리(숨김 폴더)가 생성
 - Git을 사용하면서 생성되는 버전 정보가 저장될 저장소

```
$ git init
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The command \$ git init is entered, followed by the output 'Initialized empty Git repository in C:/Users/KBS/hello-git/.git/'. The output line is highlighted with a red box.

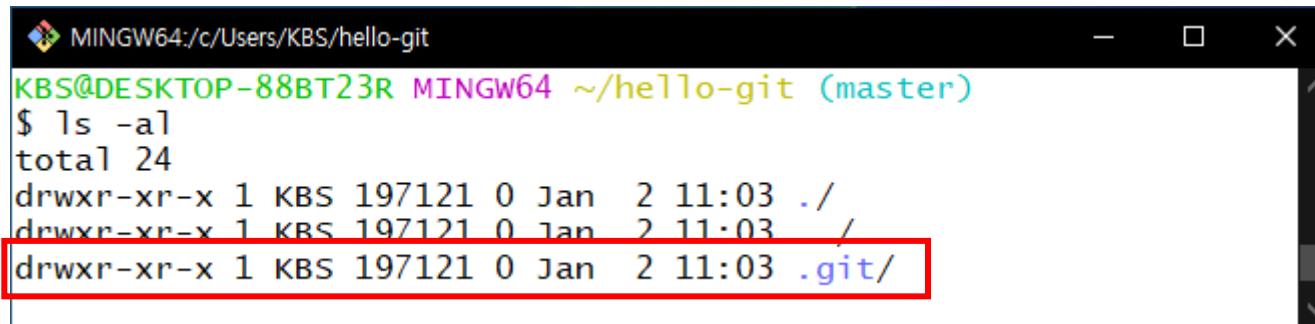
```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git
$ git init
Initialized empty Git repository in C:/Users/KBS/hello-git/.git/
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$
```

Git 저장소 만들기

❖ Git 초기화 하기

- 디렉터리 내부 확인

```
$ ls -al
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ ls -al
total 24
drwxr-xr-x 1 KBS 197121 0 Jan  2 11:03 .
drwxr-xr-x 1 KBS 197121 0 Jan  2 11:03 /
drwxr-xr-x 1 KBS 197121 0 Jan  2 11:03 .git/
```

- .git 디렉터리
 - Git을 사용하면서 버전이 저장될 저장소

02

버전 만들기

버전 만들기

❖ 버전이란?

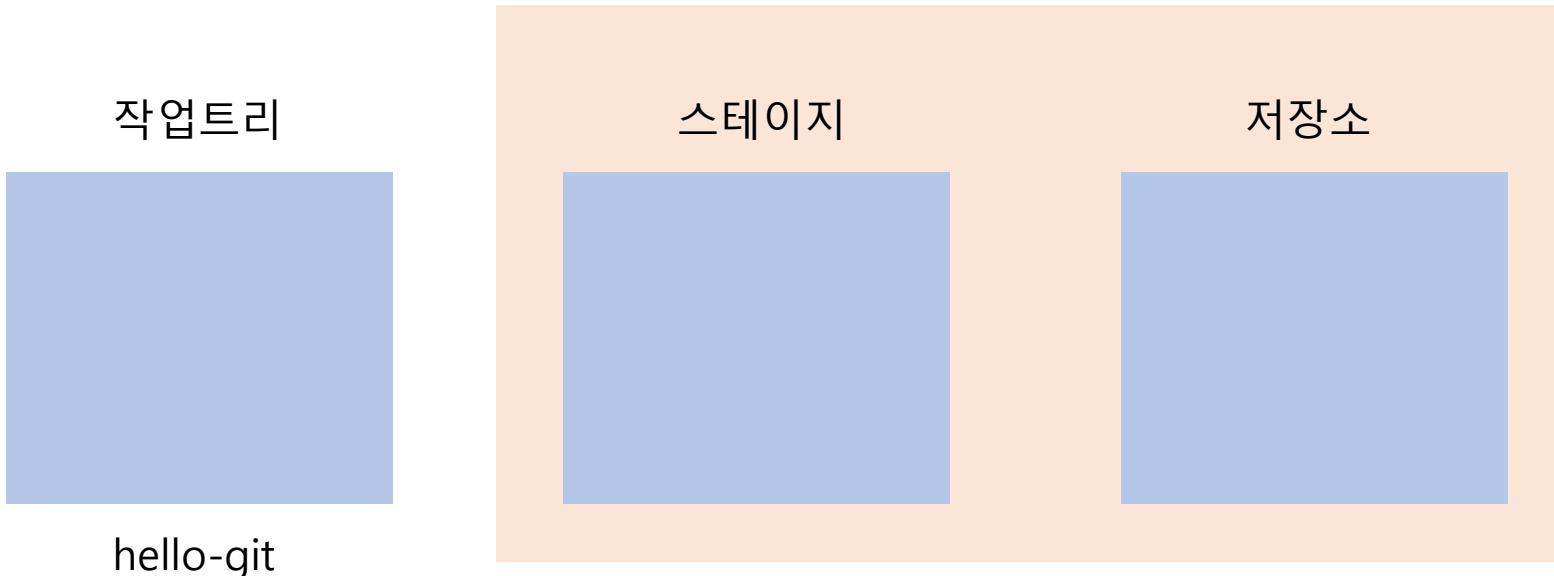
- Git에서 문서를 수정하고 저장할 때마다 생기는 것
- 특징
 - 버전마다 변경 시점과 변경 내용을 확인할 수 있음
 - 이전 버전으로 되돌아갈 수 있음

Git은 어떻게 파일 이름은 그대로 유지하면서 수정 내역을 기록할까?

버전 만들기

❖ 스테이지와 커밋

.git 디렉터리



버전 만들기

❖ 작업 트리(working tree)

- 파일 수정, 저장 등의 작업을 하는 디렉터리
- 작업 디렉터리(working directory)라고도 함
- 우리 눈에 보이는 디렉터리를 말함

❖ 스테이지(stage)

- 버전으로 만들 파일이 대기하는 곳
- 눈에 보이지 않음

❖ 저장소(repository)

- 스테이지에서 대기하고 있던 파일들을 버전으로 만들어 저장하는 장소
- 눈에 보이지 않음

버전 만들기

❖ 버전 생성 과정

1) 작업 트리에서 파일을 수정하고 저장



2) Git으로 버전 관리를 하기 위해 스테이지에 등록



버전 만들기

❖ 버전 생성 과정

- 3) 버전을 만들기 위해 '커밋(commit)' 명령을 실행



- 4) 스테이지에 있던 파일을 저장소에 저장(V1)



버전 만들기

❖ 버전 생성 과정

5) 작업 트리에서 파일을 수정하고 저장



6) Git으로 버전 관리를 하기 위해 스테이지에 등록



버전 만들기

❖ 버전 생성 과정

- 7) 버전을 만들기 위해 '커밋(commit)' 명령을 실행



- 8) 스테이지에 있던 파일을 저장소에 새로운 버전으로 저장(V2)

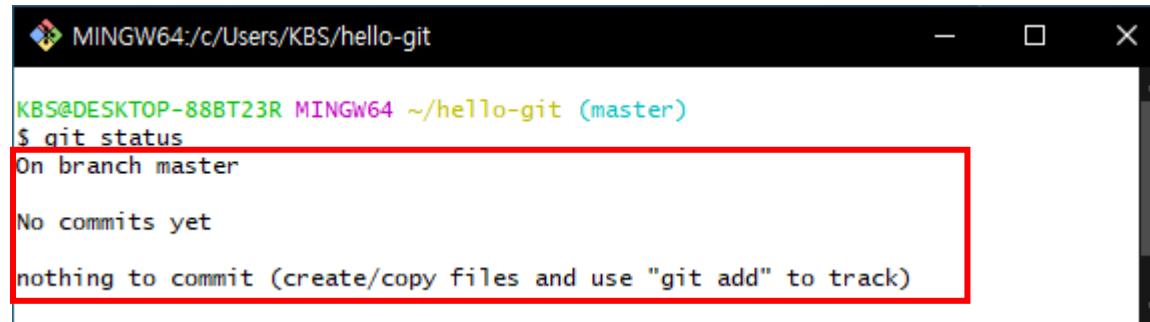


[실습] 버전 만들기

❖ 작업 트리에서 빔으로 문서 수정하기

- hello-git 디렉터리로 이동 후, Git 상태 확인

```
$ git status
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The window contains the following text:

```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
① On branch master
② No commits yet
③ nothing to commit (create/copy files and use "git add" to track)
```

Three lines of the output are highlighted with a red box and numbered 1, 2, and 3.

- ① 현재 master 브랜치에 있음
- ② 아직 커밋한 파일이 없음
- ③ 현재 커밋할 파일이 없음

[실습] 버전 만들기

❖ 작업 트리에서 빔으로 문서 수정하기

- 새로운 파일(hello.txt) 생성

```
$ vim hello.txt
```

- 입력 모드(A 또는 I)로 변경 후, 내용 입력



The screenshot shows a terminal window titled 'MINGW64:/c/Users/sophia/hello-git'. The window contains the following text:

```
안녕하세요!
정수아입니다.

~
```

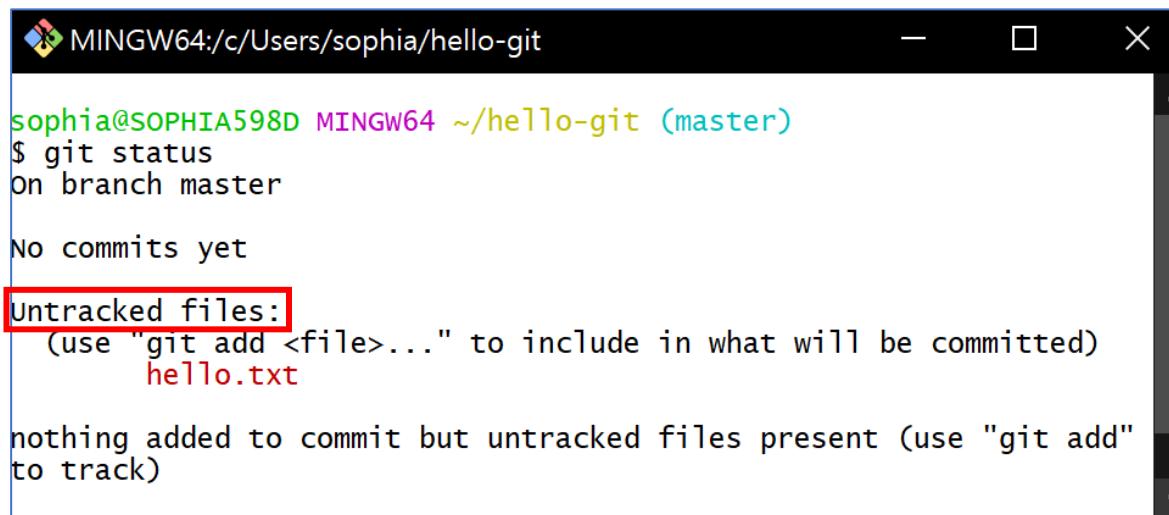
The status bar at the bottom of the terminal window displays the file name 'hello.txt[+]' and its status '(unix) (08:59 01/01/1970)'. It also shows the current line count '2,1 All' and the mode indicator '-- INSERT --'.

[실습] 버전 만들기

❖ 작업 트리에서 빔으로 문서 수정하기

- ex 모드(Esc)로 변경 후, 문서 저장(:wq)
- Git 상태 확인

```
$ git status
```



```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git status
on branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    hello.txt

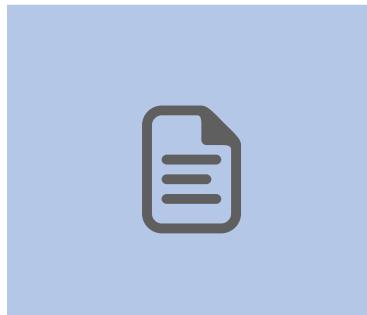
nothing added to commit but untracked files present (use "git add"
to track)
```

[실습] 버전 만들기

❖ Untracked files

- Git에서 아직 한번도 관리하지 않은 파일을 의미
- ex) hello.txt

작업트리



hello.txt

스테이지



저장소



[실습] 버전 만들기

❖ 수정한 파일 스테이징하기



[실습] 버전 만들기

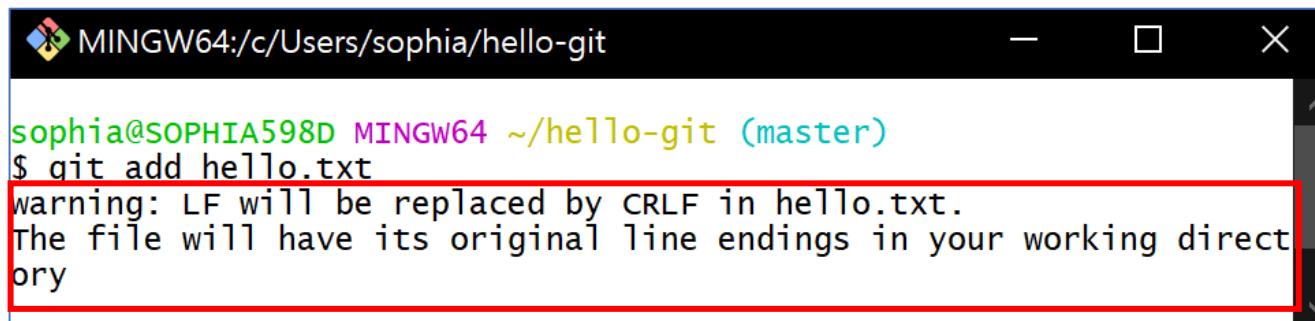
❖ 수정한 파일 스테이징하기

- 스테이지에 수정한 파일 추가

```
$ git add 파일명
```

- 예) 스테이지에 hello.txt를 추가

```
$ git add hello.txt
```



The screenshot shows a terminal window with a black background and white text. The title bar reads "MINGW64:/c/Users/sophia/hello-git". The command entered was "\$ git add hello.txt". The terminal output includes a warning message: "warning: LF will be replaced by CRLF in hello.txt. The file will have its original line endings in your working directory". A red rectangular box highlights this warning message.

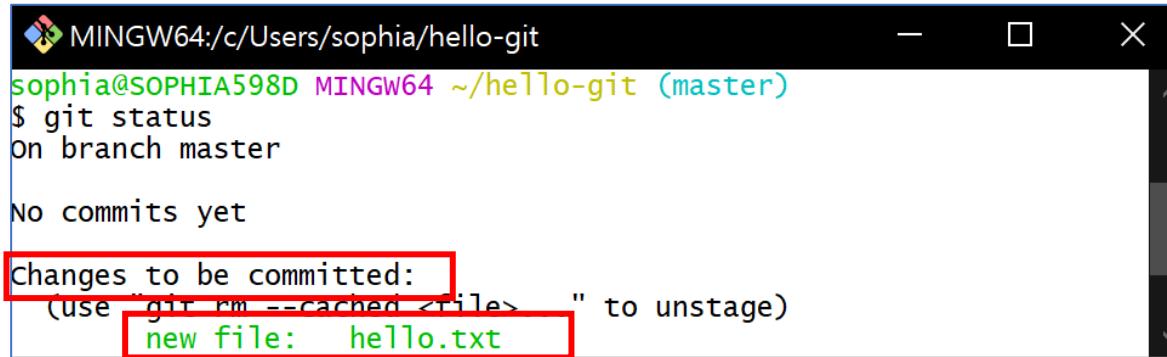
```
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git add hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
```

[실습] 버전 만들기

❖ 수정한 파일 스테이징하기

- Git 상태 확인

```
$ git status
```



```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git status
on branch master

No commits yet

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

- 새 파일 hello.txt를 커밋할 예정임을 의미

[실습] 버전 만들기

❖ 스테이지에 올라온 파일 커밋하기



[실습] 버전 만들기

❖ 스테이지에 올라온 파일 커밋하기

- 파일 커밋

```
$ git commit -m "커밋메시지"
```

- 커밋 메시지

- 커밋할 때 해당 버전에 어떤 변경 사항이 있었는지 확인하기 위한 메시지를 기록
 - 영어로 작성하는 것이 좋음

[실습] 버전 만들기

❖ 스테이지에 올라온 파일 커밋하기

- 예) 커밋 메시지에 "message1" 작성

```
$ git commit -m "message1"
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/sophia/hello-git'. The command '\$ git commit -m "message1"' is entered, followed by its output: '[master (root-commit) d4a20d5] message1'. A red box highlights the line '1 file changed, 2 insertions(+)'.

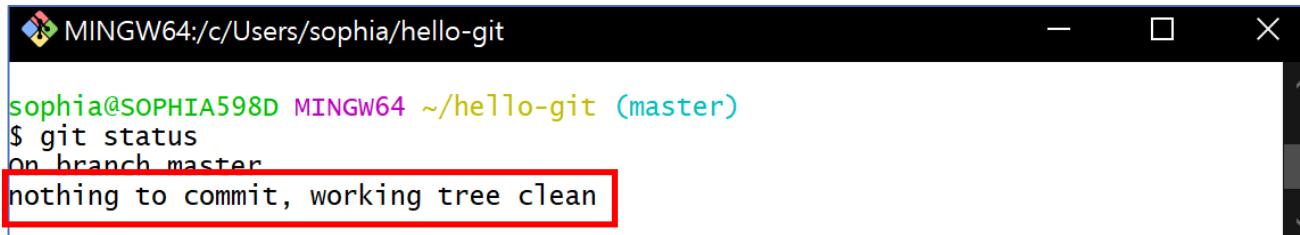
```
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git commit -m "message1"
[master (root-commit) d4a20d5] message1
1 file changed, 2 insertions(+)
create mode 100644 hello.txt
```

- 파일 1개가 변경되었으며, 파일에 2개의 내용이 추가됨

[실습] 버전 만들기

- ❖ 스테이지에 올라온 파일 커밋하기
 - Git 상태 확인

```
$ git status
```



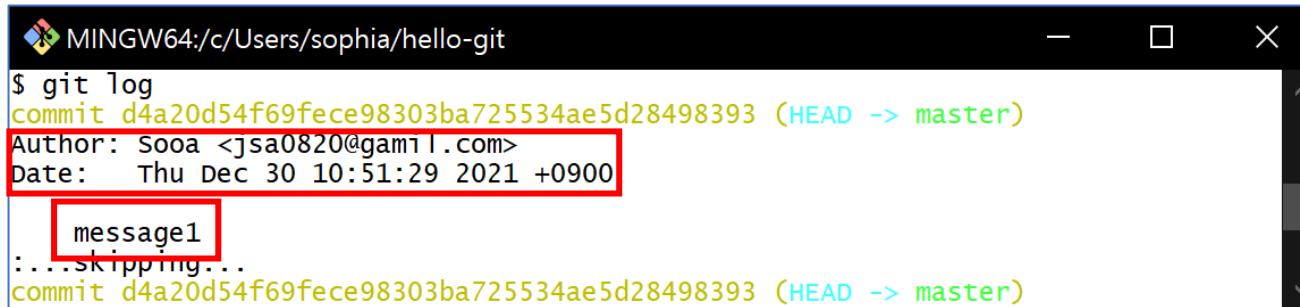
```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- nothing to commit
 - 버전으로 만들 파일이 없음
- working tree clean
 - 작업 트리에도 수정사항이 없음

[실습] 버전 만들기

❖ 커밋한 버전 확인하기

```
$ git log
```



```
MINGW64:/c/Users/sophia/hello-git
$ git log
commit d4a20d54f69fece98303ba725534ae5d28498393 (HEAD -> master)
Author: Sooa <jsa0820@gmail.com>
Date:   Thu Dec 30 10:51:29 2021 +0900

    message1
:...skipping...
commit d4a20d54f69fece98303ba725534ae5d28498393 (HEAD -> master)
```

- Author
 - 커밋을 만든 사람
- Date
 - 커밋 시간 및 커밋 메시지

[실습] 버전 만들기

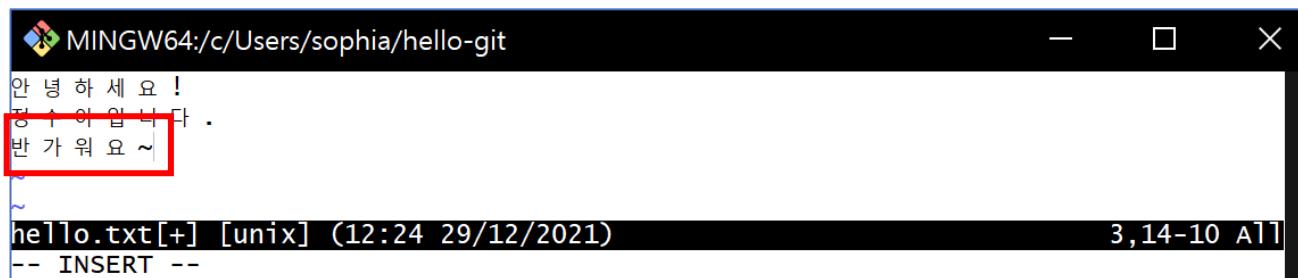
❖ 스테이징과 커밋 한번에 처리하기

- 한 번 이상 커밋한 파일을 다시 커밋할 때만 사용 가능

```
$ git commit -am "커밋메시지"
```

- hello.txt 파일 수정

```
$ vim hello.txt
```



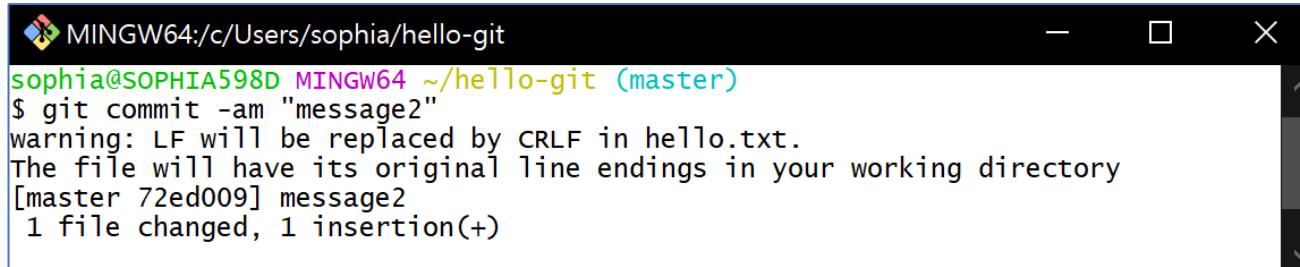
```
MINGW64:/c/Users/sophia/hello-git
안녕하세요!
정수아입니다.
반가워요~
```

hello.txt[+] [unix] (12:24 29/12/2021) 3,14-10 All
-- INSERT --

[실습] 버전 만들기

❖ 스테이징과 커밋 한번에 처리하기

```
$ git commit -am "message2"
```



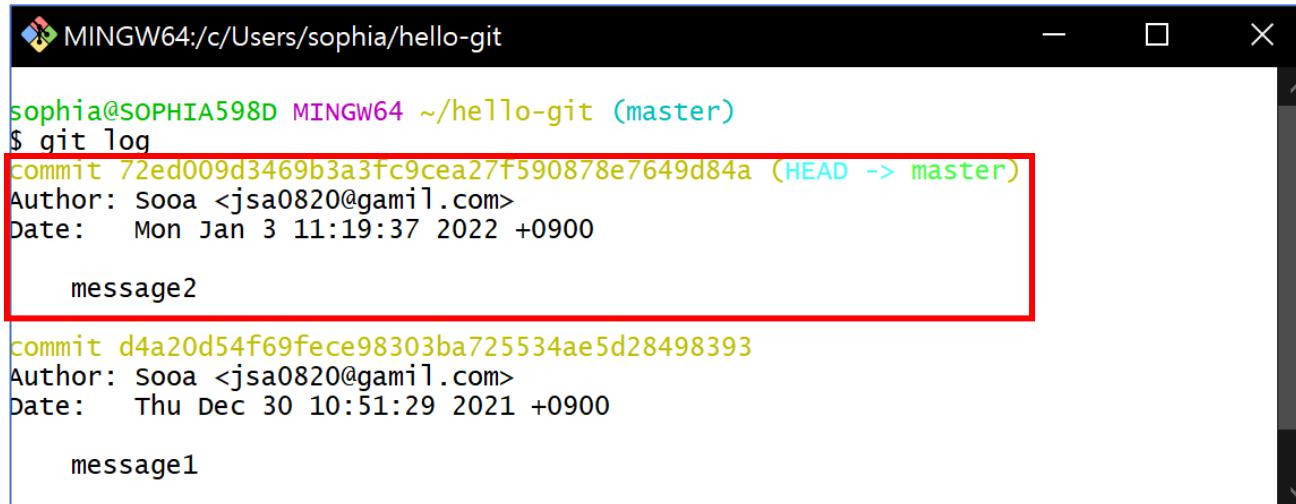
The screenshot shows a terminal window titled 'MINGW64:/c/Users/sophia/hello-git'. The command \$ git commit -am "message2" is entered, followed by a warning about line endings and a summary of the commit.

```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git commit -am "message2"
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
[master 72ed009] message2
 1 file changed, 1 insertion(+)
```

[실습] 버전 만들기

❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git log
commit 72ed009d3469b3a3fc9cea27f590878e7649d84a (HEAD -> master)
Author: Sooa <jsa0820@gmail.com>
Date:   Mon Jan 3 11:19:37 2022 +0900

    message2

commit d4a20d54f69fece98303ba725534ae5d28498393
Author: Sooa <jsa0820@gmail.com>
Date:   Thu Dec 30 10:51:29 2021 +0900

    message1
```

03

커밋 내용 확인하기

커밋 내용 확인하기

❖ 커밋 기록 확인하기

```
$ git log
```

커밋 해시
작성자
버전 생성 날짜

→ commit 72ed009d3469b3a3fc9cea27f590878e7649d84a (HEAD -> master) 최신 버전
→ Author: Sooa <jsa0820@gmail.com>
→ Date: Mon Jan 3 11:19:37 2022 +0900

message2 ← 커밋 메시지

commit d4a20d54f69fce98303ba725534ae5d28498393
Author: Sooa <jsa0820@gmail.com>
Date: Thu Dec 30 10:51:29 2021 +0900

message1

커밋 로그

```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git log
commit 72ed009d3469b3a3fc9cea27f590878e7649d84a (HEAD -> master) 최신 버전
Author: Sooa <jsa0820@gmail.com>
Date: Mon Jan 3 11:19:37 2022 +0900

    message2 ← 커밋 메시지

commit d4a20d54f69fce98303ba725534ae5d28498393
Author: Sooa <jsa0820@gmail.com>
Date: Thu Dec 30 10:51:29 2021 +0900

    message1
```

커밋 내용 확인하기

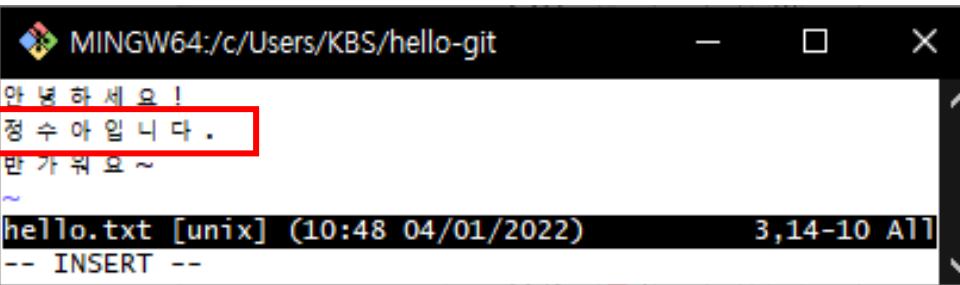
❖ 변경 사항 확인하기

```
$ git diff
```

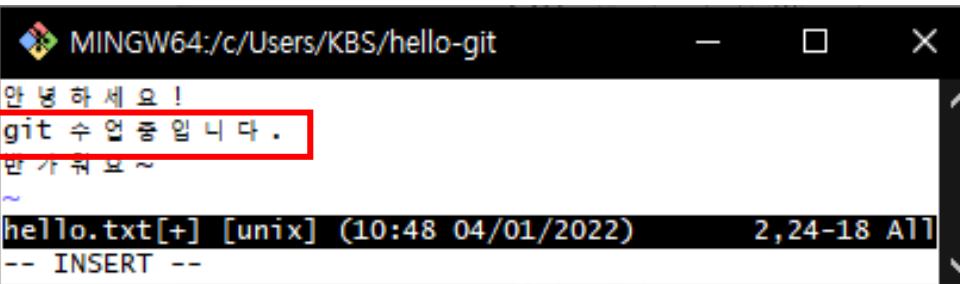
- 특징
 - 작업 트리와 스테이지에 있는 파일의 차이
 - 스테이지와 저장소에 있는 최신 커밋의 차이
 - 특정 두 커밋 간의 차이
 - 특정 두 브랜치 간의 차이

[실습] 변경 사항 확인하기

❖ hello.txt 파일 수정하기



```
MINGW64:/c/Users/KBS/hello-git
안녕하세요!
정수아입니다.
반가워요~
~  
hello.txt [unix] (10:48 04/01/2022) 3,14-10 All
-- INSERT --
```



```
MINGW64:/c/Users/KBS/hello-git
안녕하세요!
git 수업 중입니다.
반가워요~
~  
hello.txt[+] [unix] (10:48 04/01/2022) 2,24-18 All
-- INSERT --
```

[실습] 변경 사항 확인하기

❖ Git 상태 확인하기

```
$ git status
```



```
$ 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:   hello.txt

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

- 아직 스테이징 상태가 아님
- hello.txt 파일이 수정되었음

[실습] 변경 사항 확인하기

❖ 변경 사항 확인하기

- 방금 수정한 파일과 스테이지에 있는 파일 비교

```
$ git diff
```

```
warning: LF will be replaced by CRLF in hello.txt.  
The file will have its original line endings in your working directory  
diff --git a/hello.txt b/hello.txt  
index fbc2450..09cfalc 100644  
--- a/hello.txt  
+++ b/hello.txt  
@@ -1,3 +1,3 @@  
① -정수아인니.  
② +git수업중입니다.  
~  
~  
(END)
```

- ① - 최신 버전과 비교 시, 삭제 되었음을 의미
- ② + 최신 버전과 비교 시, 추가 되었음을 의미

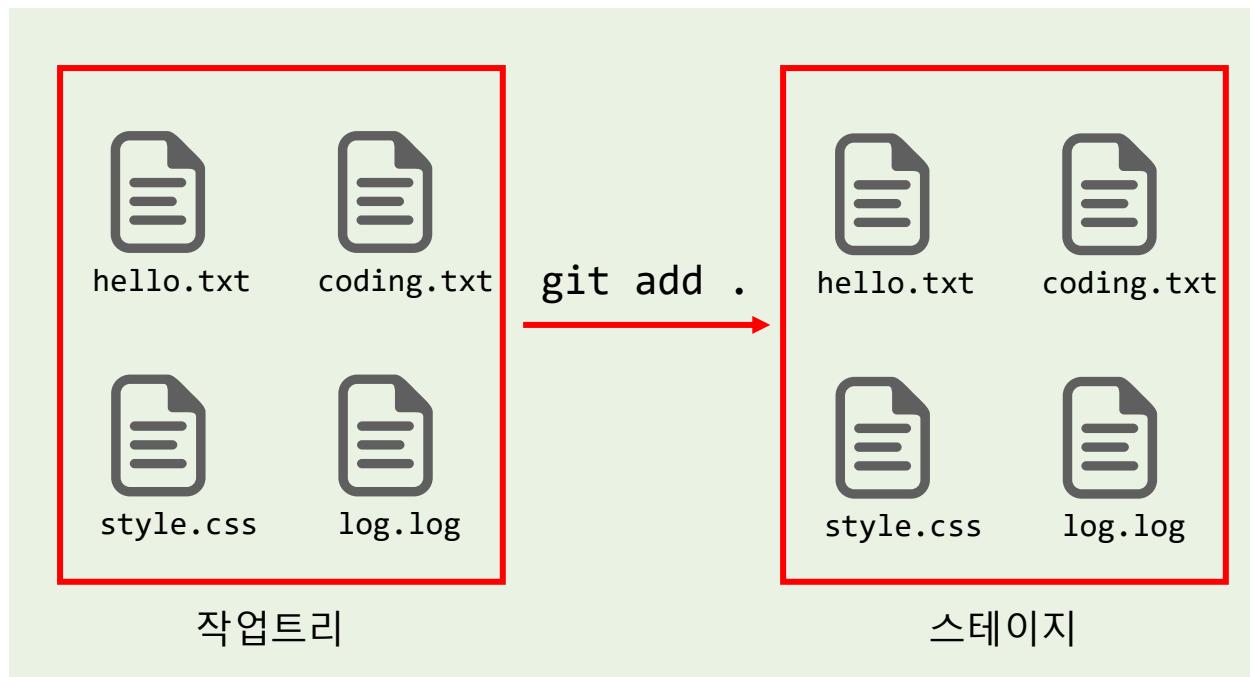
04

.gitignore

.gitignore

❖ .gitignore 파일

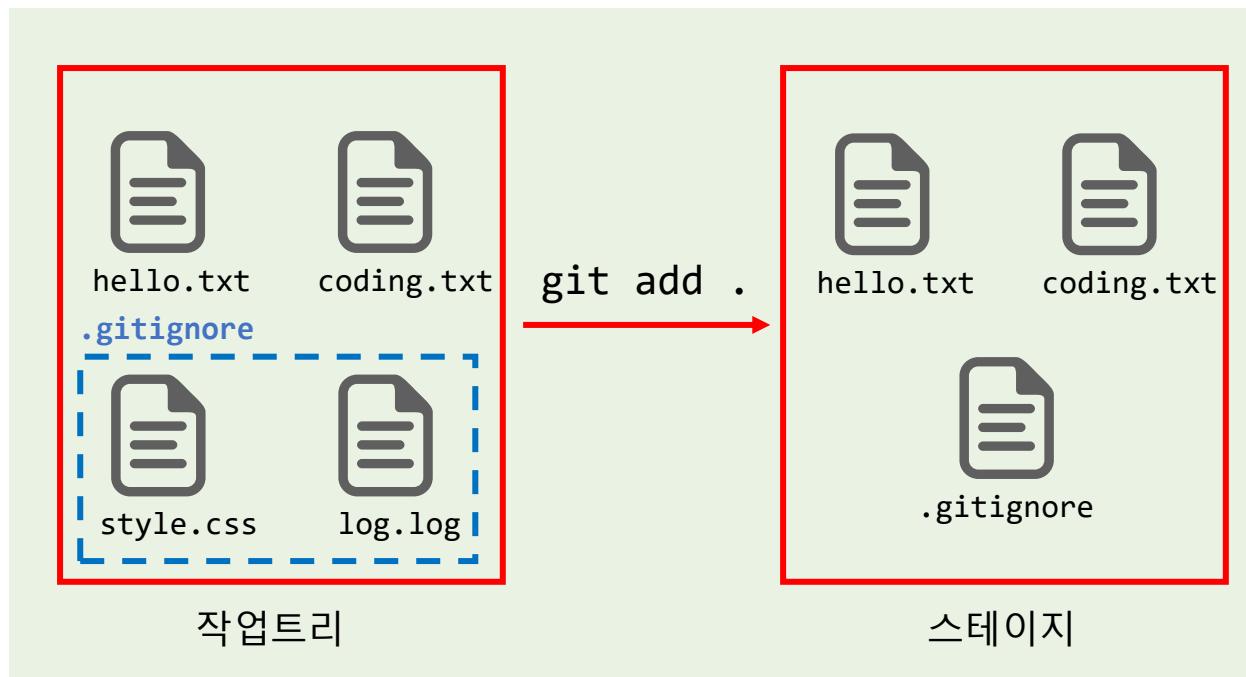
- git에서 관리할 필요가 없는 파일 또는 디렉터리를 작성
- 작성한 파일 또는 디렉터리는 git add 실행 시, 스테이징 안됨



.gitignore

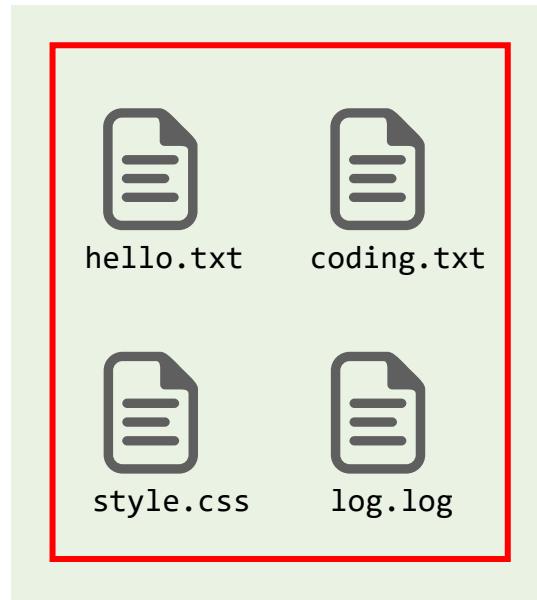
❖ .gitignore 파일

- git에서 관리할 필요가 없는 파일 또는 디렉터리를 작성
- 작성한 파일 또는 디렉터리는 git add 실행 시, 스테이징 안됨



[실습] .gitignore

- ❖ 작업트리에 파일 생성



[실습] .gitignore

❖ .gitignore 파일 생성

```
$ vim .gitignore
```

❖ 제외할 파일 또는 디렉터리 작성



The screenshot shows a terminal window titled "MINGW64:c/Users/KBS/Desktop/git-test". Inside the terminal, the command "\$ vim .gitignore" has been run. The .gitignore file contains two entries: "style.css" and "*.log", which are highlighted with a red rectangle. The bottom status bar of the terminal shows ".gitignore[+]" and "(08:59 01/01/1970)".

[실습] .gitignore

- ❖ 작업트리에 있는 전체 파일을 스테이지에 추가

```
$ git add .
```

- ❖ Git 상태 확인하기

```
$ git status
```



```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:  .gitignore
    new file:  coding.txt
    modified: hello.txt
```

.gitignore

❖ 어떤 파일을 제외할까?

- 보안상 위험성이 있는 파일
- 프로젝트와 관련 없는 파일
- 용량이 너무 커서 제외해야 하는 파일

.gitignore

❖ .gitignore 파일 작성 방법

- 특정 파일 제외

```
style.css
```

- 현재 경로에 있는 특정 파일만 제외

```
/style.css
```

- 특정 디렉터리 안의 모든 파일 제외

```
folder/
```

.gitignore

❖ .gitignore 파일 작성 방법

- 특정 디렉터리의 특정 파일 제외

```
folder/style.css
```

- 특정 확장자 파일 모두 제외

```
*.css
```

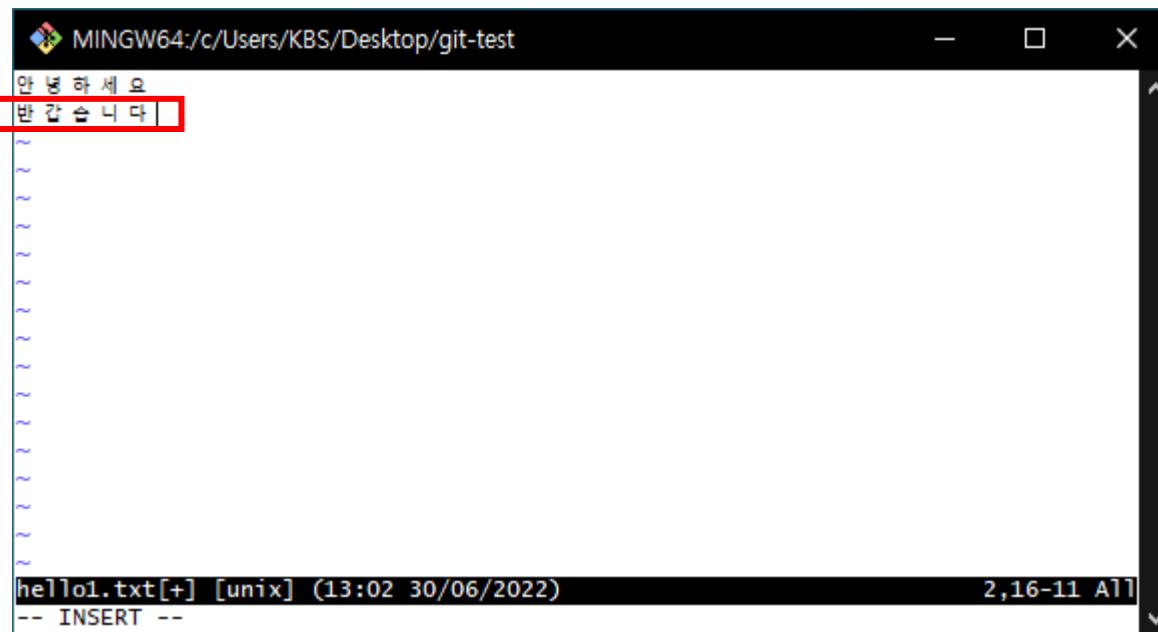
05

단계마다 파일 상태
알아보기

tracked 파일 vs untracked 파일

❖ tracked 파일

- Git이 버전 관리를 하고 있는 파일
- hello.txt 파일 수정

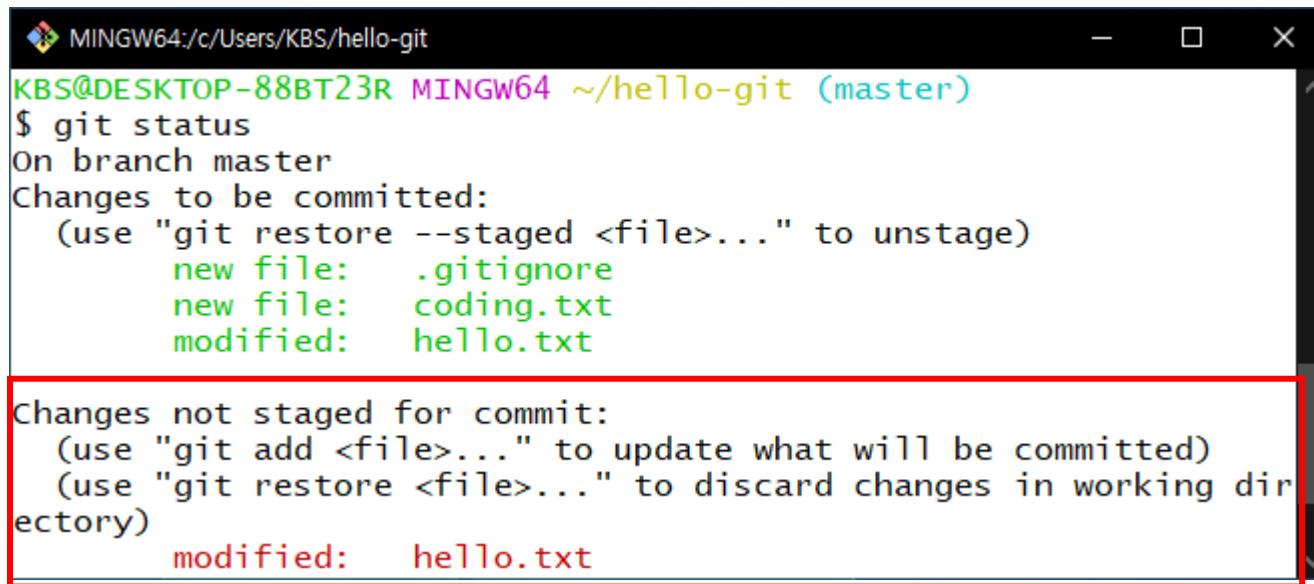


The screenshot shows a terminal window titled "MINGW64:/c/Users/KBS/Desktop/git-test". Inside the window, there is a single line of text: "안녕하세요 반갑습니다". The last part of the line, "반갑습니다", is highlighted with a red rectangular box. At the bottom of the terminal, the status bar displays "hello1.txt[+]" and "(13:02 30/06/2022)". The bottom right corner of the status bar shows "2,16-11 All". The bottom left corner shows "-- INSERT --".

tracked 파일 vs untracked 파일

❖ tracked 파일

```
$ git status
```



The screenshot shows a terminal window on a Windows system (MINGW64) with the following output:

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   coding.txt
    modified:  hello.txt

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:  hello.txt
```

A red box highlights the "Changes not staged for commit" section, specifically the line "modified: hello.txt".

tracked 파일 vs untracked 파일

❖ untracked 파일

- Git이 버전 관리를 하지 않는 파일

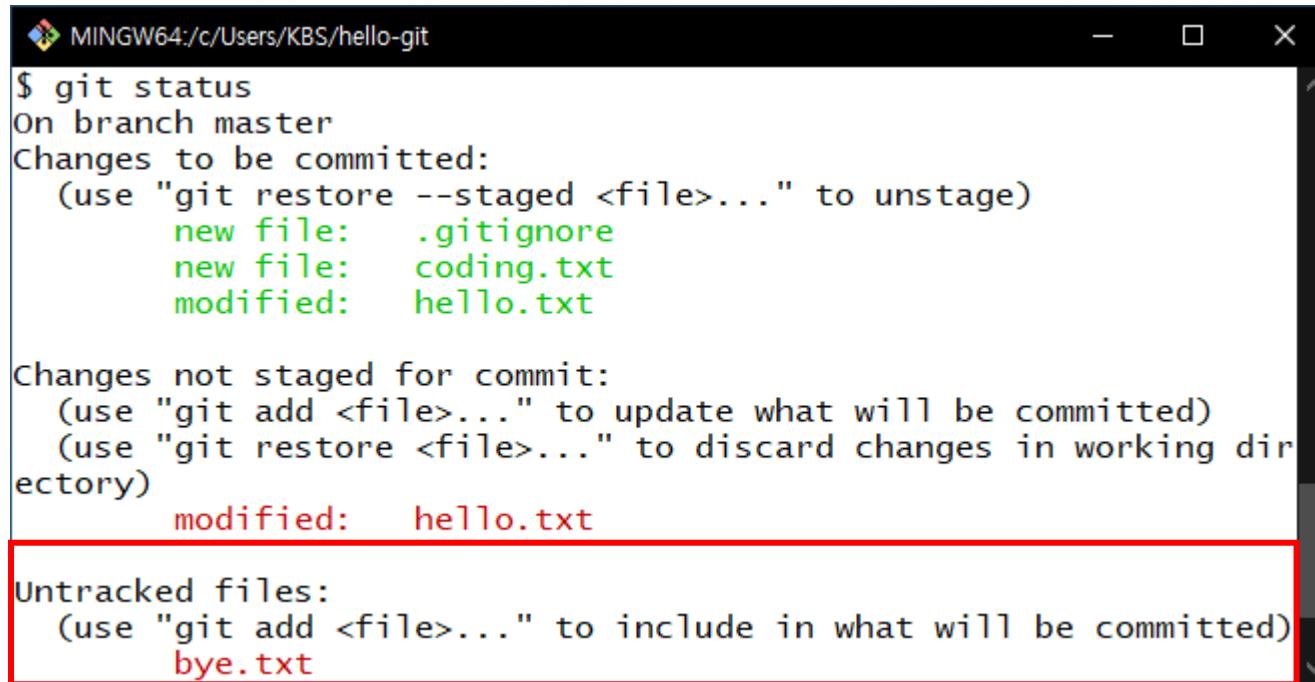
```
$ vim bye.txt
```

```
MINGW64:/c/Users/KBS/hello-git
1
2
3
4
~
~
~
~
~
~
~
~
~
~
~
by e.txt[+] [unix] (11:25 02/01/2023) 3,1 All
-- INSERT --
```

tracked 파일 vs untracked 파일

❖ untracked 파일

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   coding.txt
    modified:   hello.txt

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:   hello.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    bye.txt
```

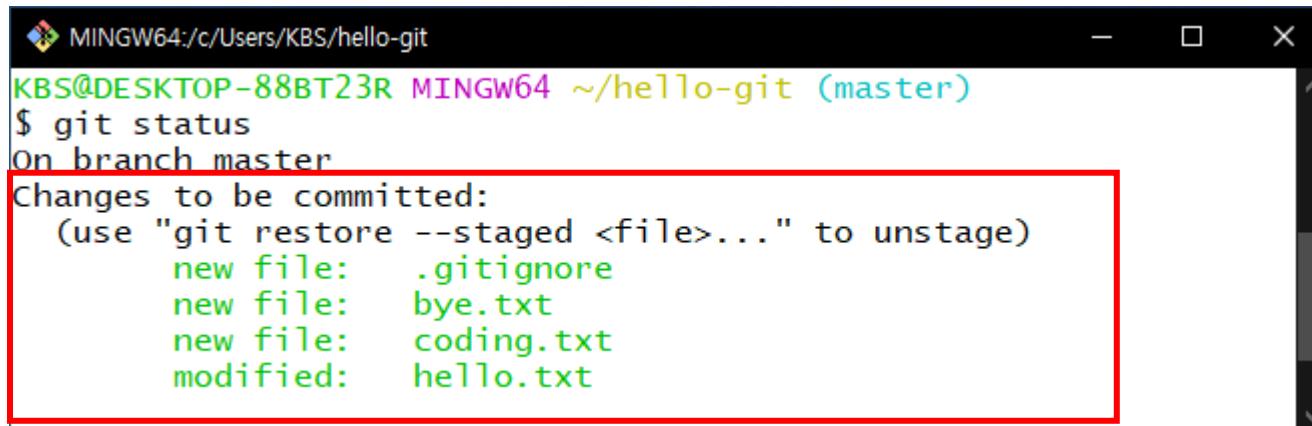
tracked 파일 vs untracked 파일

❖ 스테이지에 추가하기

```
$ git add .
```

❖ Git 상태 확인하기

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
  new file:  .gitignore
  new file:  bye.txt
  new file:  coding.txt
  modified: hello.txt
```

tracked 파일 vs untracked 파일

❖ 스테이지에 올라온 파일 커밋하기

```
$ git commit -m "message3"
```

❖ 커밋 기록 확인하기

```
$ git log
```



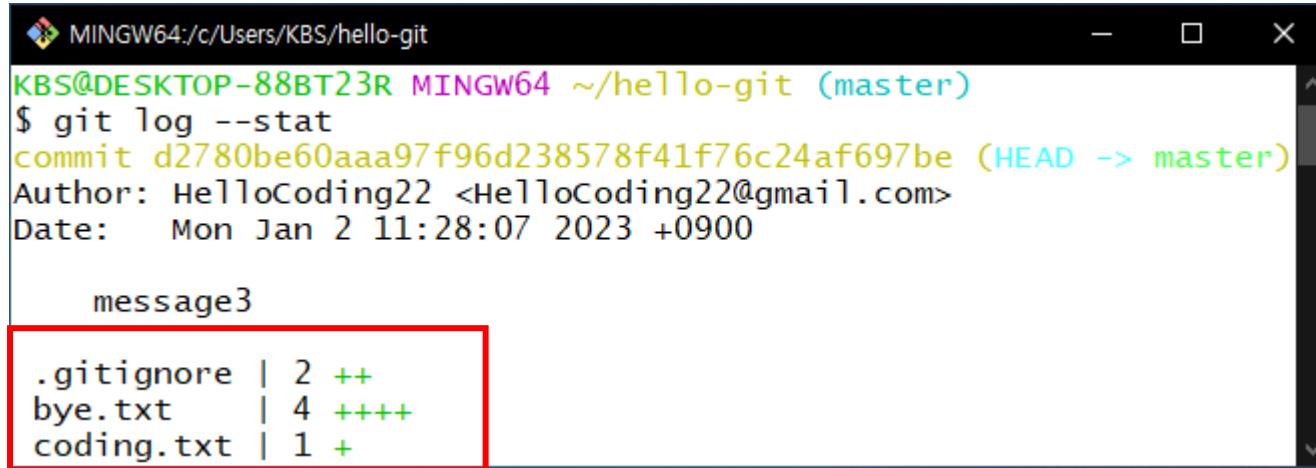
```
MINGW64:/c/Users/KBS/hello-git
commit 6939d47c67d59e71075e9df440be74ab9c8c8a6c (HEAD -> master)
Author: Sooa <jsa0820@gmail.com>
Date:   Tue Jan 4 12:00:16 2022 +0900

    message3
```

tracked 파일 vs untracked 파일

❖ 커밋의 통계 정보 확인하기

```
$ git log --stat
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log --stat
commit d2780be60aaa97f96d238578f41f76c24af697be (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:28:07 2023 +0900

    message3

  .gitignore | 2 ++
  bye.txt     | 4 +++
  coding.txt  | 1 +

```

06

작업 되돌리기

작업 되돌리기

❖ 작업 트리에서 수정한 파일 되돌리기

```
$ git restore 파일명
```

- 파일을 수정한 후, 소스가 정상적으로 작동하지 않는 등의 이유로 수정한 내용을 취소하고 가장 최신 버전으로 되돌려야 하는 경우에 사용

[실습] 작업 트리에서 수정한 파일 되돌리기

❖ bye.txt 파일 수정하기

```
$ vim bye.txt
```



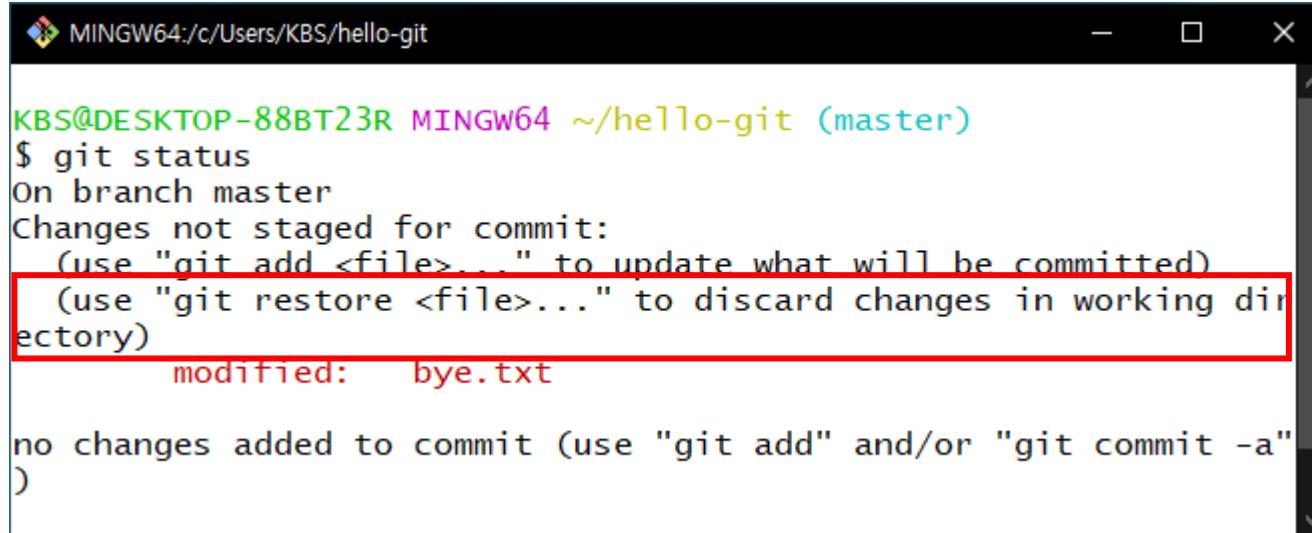
```
MINGW64:/c/Users/KBS/hello-git
1
2
three
4|
~|
```

The screenshot shows a terminal window titled "MINGW64:/c/Users/KBS/hello-git". Inside the window, there is a text file named "bye.txt" with the following content:
1
2
three
4|
~|
~|
~|
~|
~|
The word "three" is highlighted with a red box. The bottom status bar of the terminal shows the file name "bye.txt[+]" and the date and time "(11:26 02/01/2023)". It also displays the current mode "4 , 2 A11" and the status "-- INSERT --".

[실습] 작업 트리에서 수정한 파일 되돌리기

❖ Git 상태 확인하기

```
$ git status
```



The screenshot shows a terminal window titled "MINGW64:/c/Users/KBS/hello-git". The command \$ git status is run, and the output is as follows:

```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (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:   bye.txt

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

The text "(use "git add <file>..." to update what will be committed)" and "(use "git restore <file>..." to discard changes in working directory)" are highlighted with a red rectangle.

[실습] 작업 트리에서 수정한 파일 되돌리기

❖ 작업 트리에서 수정한 파일 되돌리기

```
$ git restore bye.txt
```

❖ bye.txt 파일 확인



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ cat bye.txt
1
2
3
4
```

작업 되돌리기

❖ 스테이징 취소하기

```
$ git restore --staged 파일명
```

- 스테이징 영역에 추가된 변경 사항을 취소하고, 작업 트리에서 수정한 상태로 되돌리는 경우 사용

[실습] 스테이징 취소하기

❖ bye.txt 파일 수정하기

```
$ vim bye.txt
```



The screenshot shows a terminal window titled "MINGW64:/c/Users/KBS/hello-git". The file "bye.txt" is open, displaying the following content:

```
a
b
c
d
~
~
~
~
```

The status bar at the bottom of the terminal window shows the following information:

```
bye.txt[+] [dos] (09:35 03/01/2023)        4,2 A11
-- INSERT --
```

[실습] 스테이징 취소하기

❖ 스테이지에 추가하기

```
$ git add bye.txt
```

❖ Git 상태 확인하기

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   bye.txt
```

[실습] 스테이징 취소하기

작업트리



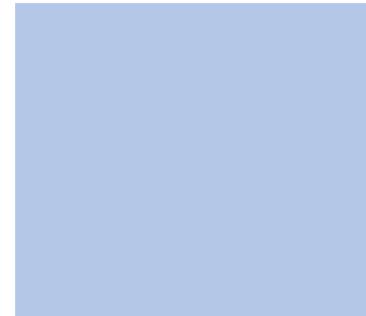
bye.txt

스테이지



bye.txt

저장소



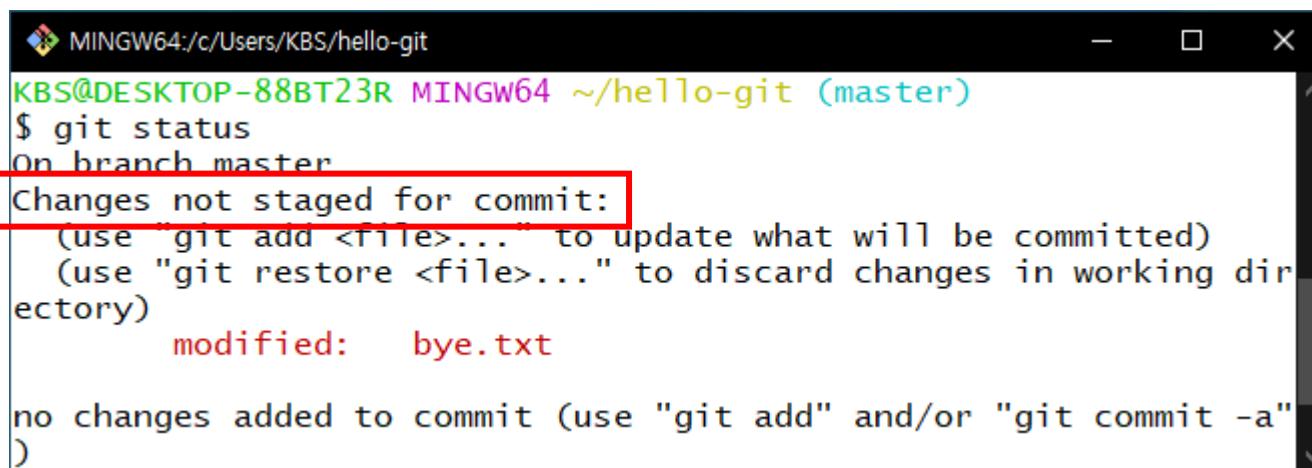
[실습] 스테이징 취소하기

❖ 스테이징 취소하기

```
$ git restore --staged bye.txt
```

❖ Git 상태 확인하기

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (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:   bye.txt

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

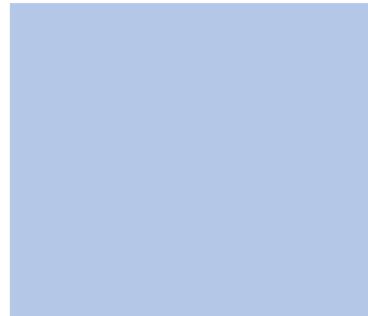
[실습] 스테이징 취소하기

작업트리

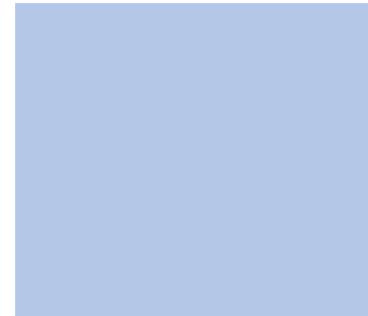


bye.txt

스테이지



저장소



작업 되돌리기

❖ 최신 커밋 취소하기

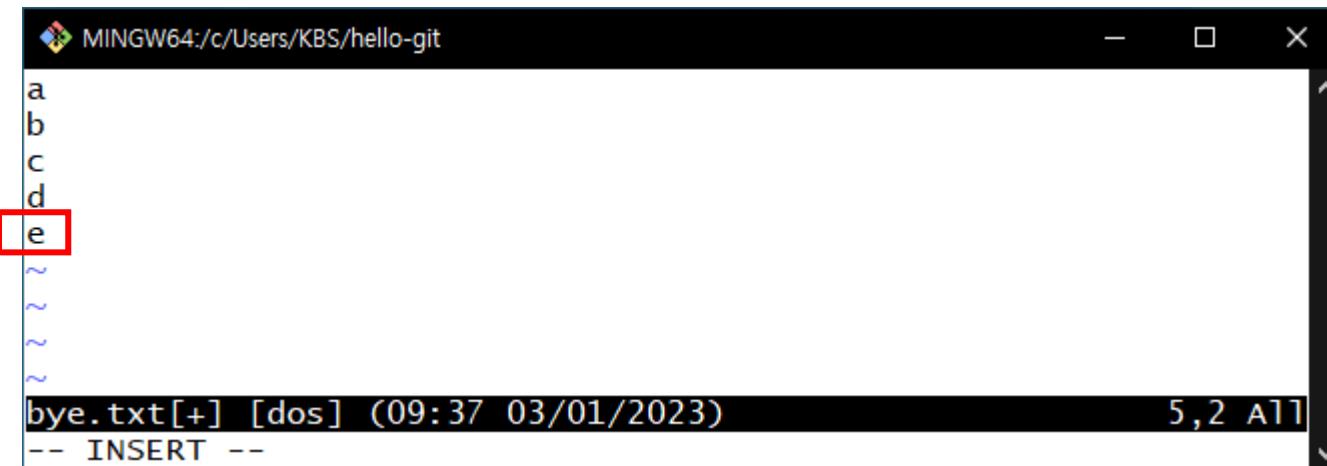
```
$ git reset HEAD^
```

- 수정한 파일을 스테이징하고 커밋까지 했을 때, 가장 마지막에 한 커밋을 취소해야 하는 경우에 사용

[실습] 최신 커밋 취소하기

❖ bye.txt 파일 수정하기

```
$ vim bye.txt
```



```
a  
b  
c  
d  
e
```

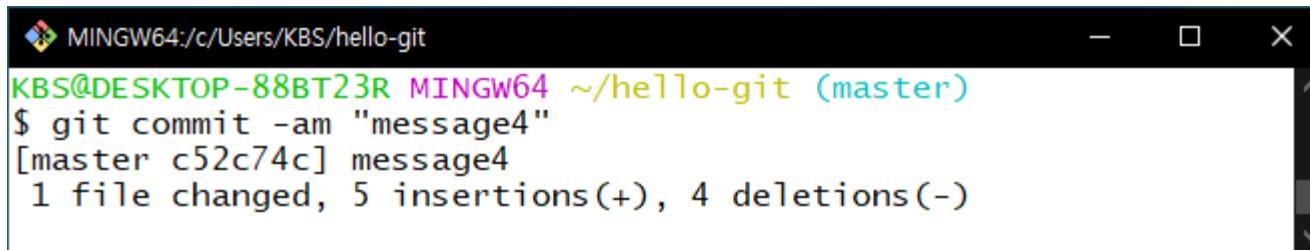
```
~  
~  
~  
~  
~
```

```
bye.txt[+] [dos] (09:37 03/01/2023) 5,2 A11  
-- INSERT --
```

[실습] 최신 커밋 취소하기

❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "message4"
```

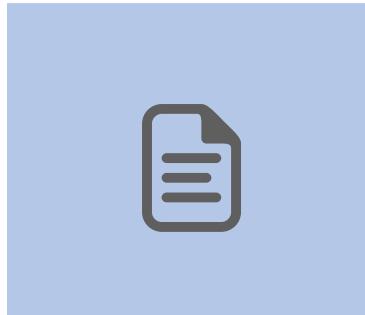


The screenshot shows a terminal window with the following text:

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "message4"
[master c52c74c] message4
 1 file changed, 5 insertions(+), 4 deletions(-)
```

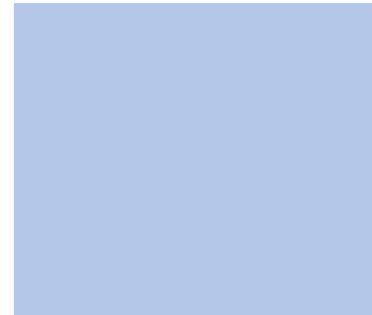
[실습] 최신 커밋 취소하기

작업트리

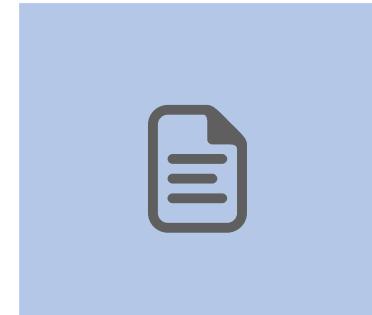


bye.txt

스테이지



저장소



bye.txt

[실습] 최신 커밋 취소하기

❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit c52c74c102241c21fba82644bf2d59ec26760815 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 09:42:00 2023 +0900

    message4

commit d2780be60aaa97f96d238578f41f76c24af697be
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:28:07 2023 +0900

    message3

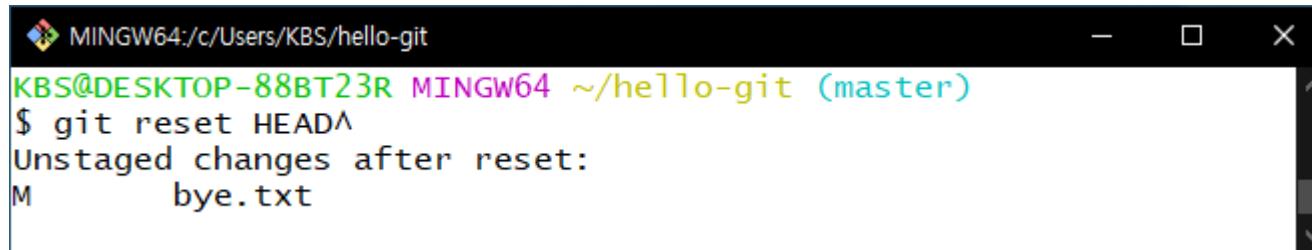
commit 308f2d87e93504a8ac30981895ba87c1bfecb215
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:11:52 2023 +0900

    message2
```

[실습] 최신 커밋 취소하기

❖ 최신 커밋 취소하기

```
$ git reset HEAD^
```

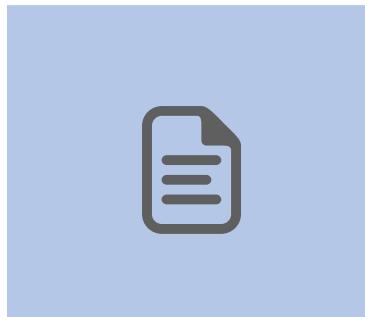


```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git reset HEAD^
Unstaged changes after reset:
M      bye.txt
```

- 최신 커밋이 취소되고, 스테이지에서도 내려감
- 취소한 파일은 작업 트리에 남아 있음

[실습] 최신 커밋 취소하기

작업트리



bye.txt

스테이지



저장소



[실습] 최신 커밋 취소하기

❖ 커밋 기록 확인하기

```
$ git log
```

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit d2780be60aaa97f96d238578f41f76c24af697be (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:28:07 2023 +0900

    message3

commit 308f2d87e93504a8ac30981895ba87c1bfecb215
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:11:52 2023 +0900

    message2

commit 23c544453cdf34e29b441403e7b14d1f2d8940fe
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:10:43 2023 +0900

    message1
```

07

특정 커밋으로 되돌리기

특정 커밋으로 되돌리기(reset)

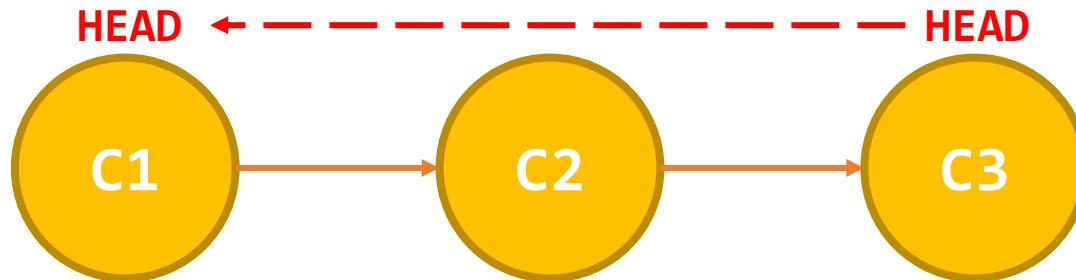
❖ 특정 커밋으로 되돌리기

```
$ git reset --옵션 되돌아갈_커밋해시
```

옵션	설명
--hard	특정 버전의 커밋으로 되돌린 다음 그 이후 버전을 삭제
--soft	커밋 이력은 삭제되지만 변경 내용은 스테이지 상태로 남아있음
--mixed	커밋 이력은 삭제되지만 변경 내용은 작업트리에 남아있음

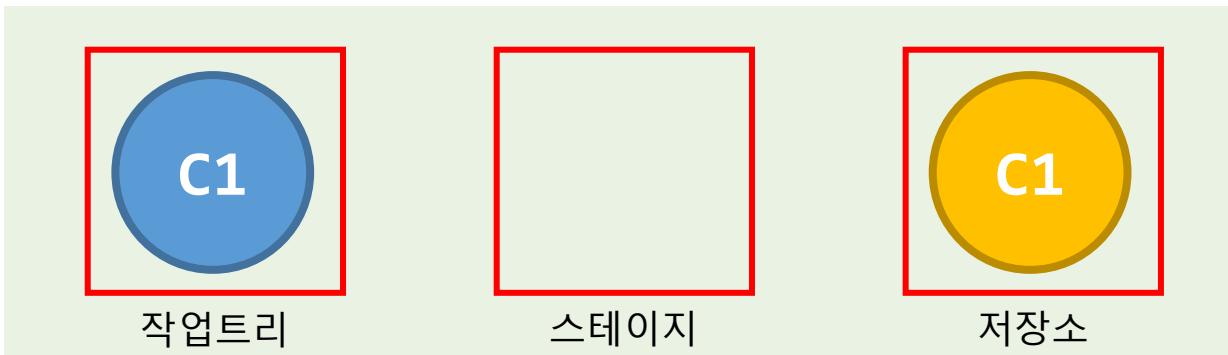
특정 커밋으로 되돌리기(reset)

❖ --hard 옵션



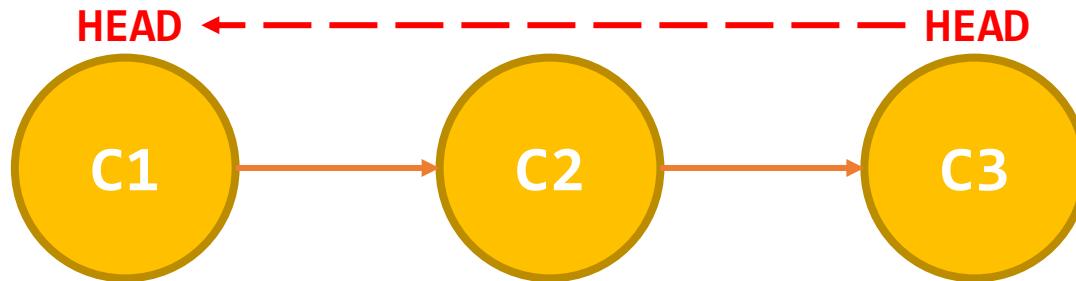
```
$ git reset --hard C1커밋해시
```

- 실행 결과



특정 커밋으로 되돌리기(reset)

❖ --soft 옵션



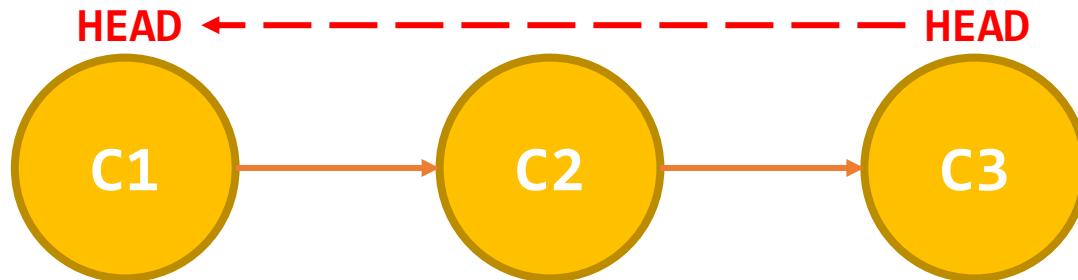
```
$ git reset --soft C1커밋해시
```

- 실행 결과



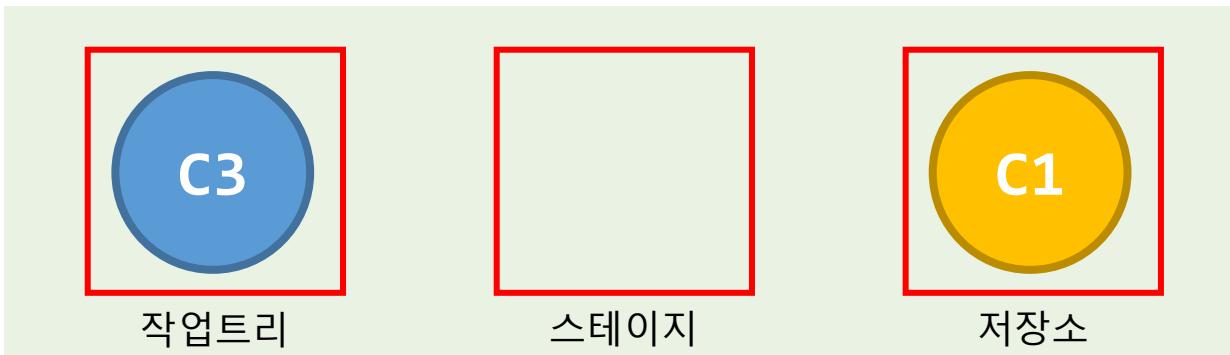
특정 커밋으로 되돌리기(reset)

❖ --mixed 옵션



```
$ git reset --mixed C1커밋해시
```

- 실행 결과



[실습] 특정 커밋으로 되돌리기(reset) --hard

❖ 새로운 파일 생성 및 내용 입력(a)

```
$ vim test.txt
```



❖ 스테이지에 추가 및 커밋 하기

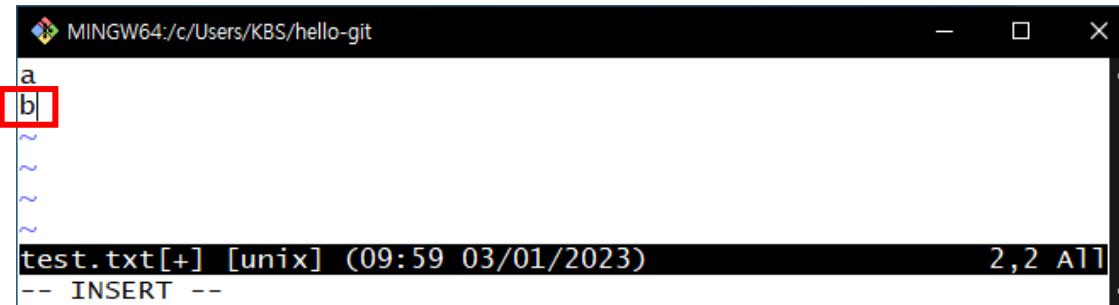
```
$ git add test.txt
```

```
$ git commit -m "test commit 1"
```

[실습] 특정 커밋으로 되돌리기(reset) --hard

❖ test.txt 파일에 내용 추가하기(b)

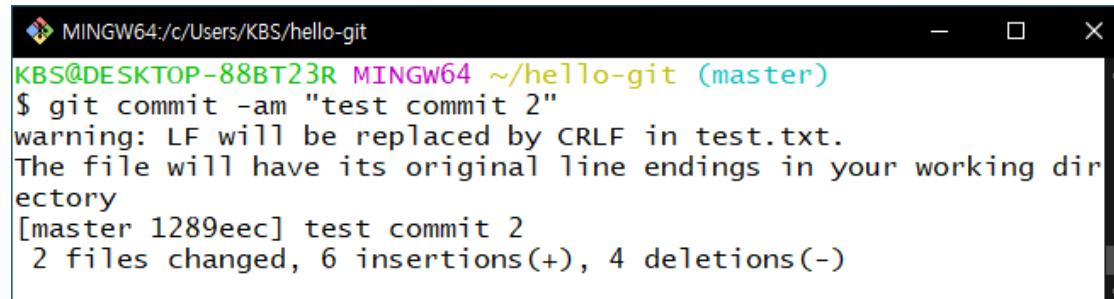
```
$ vim test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
a
b
~
~
~
~
test.txt[+] [unix] (09:59 03/01/2023) 2,2 All
-- INSERT --
```

❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "test commit 2"
```

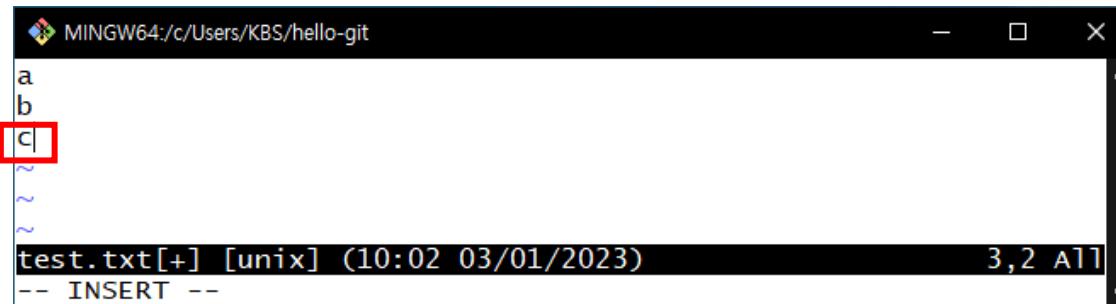


```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "test commit 2"
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory
[master 1289eec] test commit 2
2 files changed, 6 insertions(+), 4 deletions(-)
```

[실습] 특정 커밋으로 되돌리기(reset) --hard

❖ test.txt 파일에 내용 추가하기(c)

```
$ vim test.txt
```

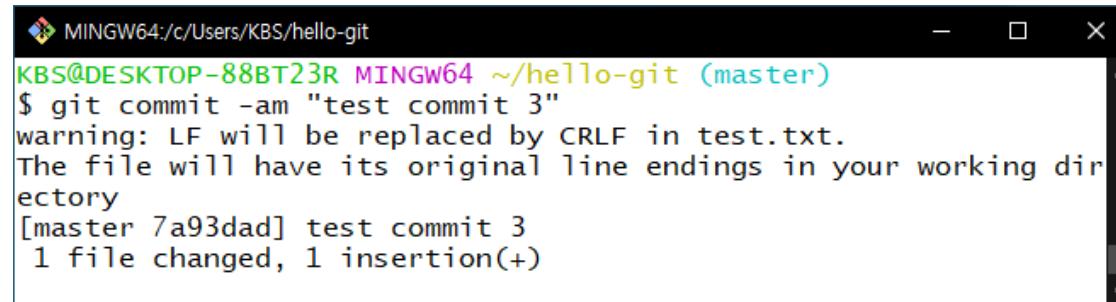


```
MINGW64:/c/Users/KBS/hello-git
a
b
c
~
~
~
```

```
test.txt[+] [unix] (10:02 03/01/2023) 3,2 All
-- INSERT --
```

❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "test commit 3"
```



```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "test commit 3"
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory
[master 7a93dad] test commit 3
 1 file changed, 1 insertion(+)
```

[실습] 특정 커밋으로 되돌리기(reset) --hard

❖ test.txt 파일에 내용 추가하기(d)

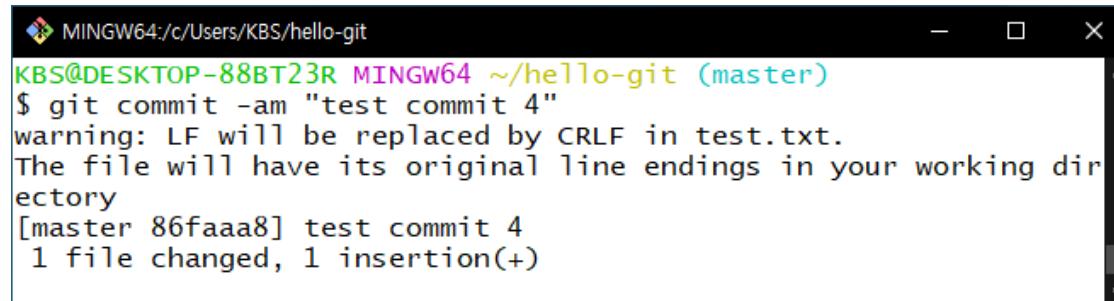
```
$ vim test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
a
b
c
d
~
~
test.txt[+] [unix] (10:02 03/01/2023)        4,1 All
```

❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "test commit 4"
```



```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "test commit 4"
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory
[master 86faaa8] test commit 4
 1 file changed, 1 insertion(+)
```

[실습] 특정 커밋으로 되돌리기(reset) --hard

❖ 커밋 기록 확인하기

```
$ git log
```



```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit 86faaa84646658e10ebea5dbda984b1f75fb8b32 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:04:19 2023 +0900

    test commit 4

commit 7a93dad7d1cede316e5bb24352f745923e1a2bcc
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:02:59 2023 +0900

    test commit 3

commit 1289ee cbd5d4792b55206aeaf78fdf0367de7eb6
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:00:57 2023 +0900

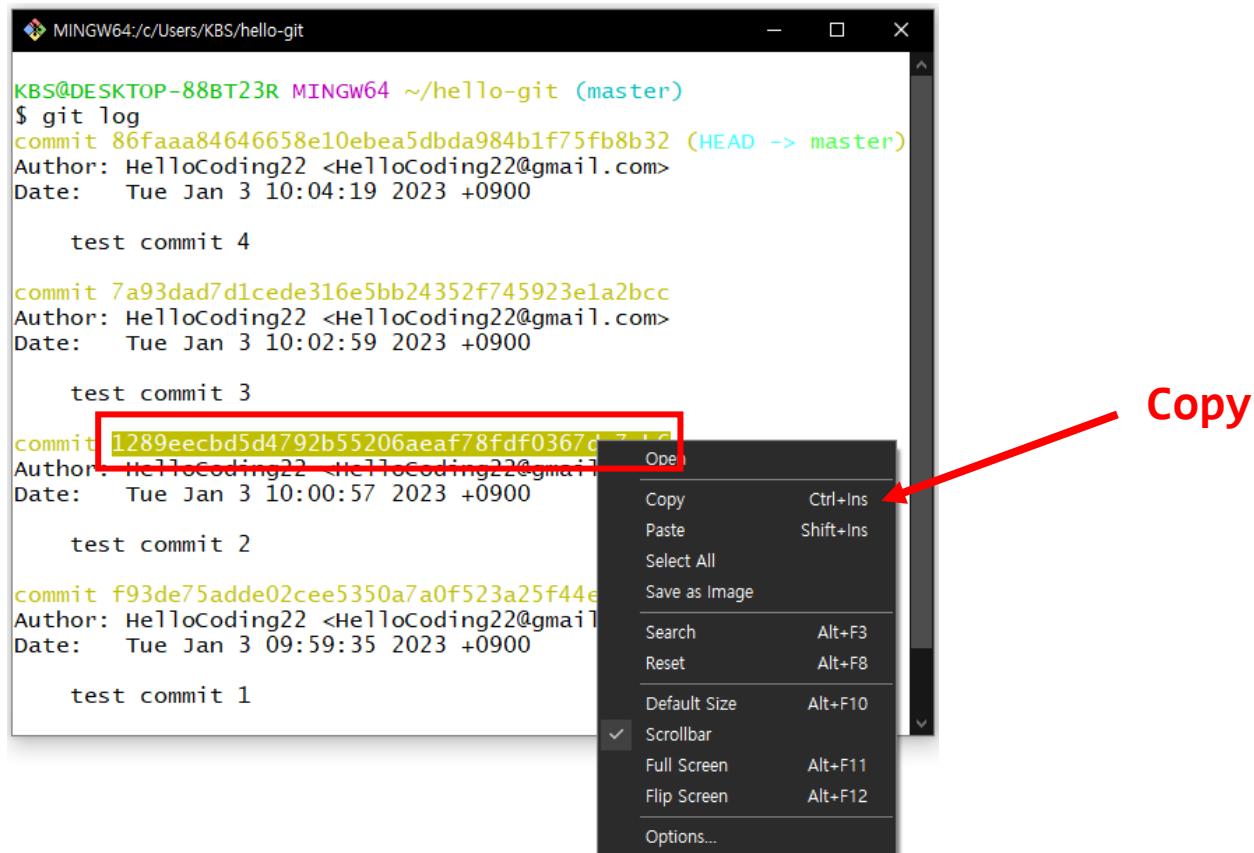
    test commit 2

commit f93de75adde02cee5350a7a0f523a25f44e3a820
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 09:59:35 2023 +0900

    test commit 1
```

[실습] 특정 커밋으로 되돌리기(reset) --hard

- ❖ "test commit 2"를 최신 커밋으로 만들기
 - "test commit 2" 커밋 해시를 복사



[실습] 특정 커밋으로 되돌리기(reset) --hard

- ❖ "test commit 2"를 최신 커밋으로 만들기

```
$ git reset --hard 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The command \$ git reset --hard 1289eecbd5d4792b55206aeaf78fdf0367de7eb6 was run, and the output 'HEAD is now at 1289eec test commit 2' is displayed. The last line of the output is highlighted with a red rectangle.

```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git reset --hard 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
HEAD is now at 1289eec test commit 2
```

[실습] 특정 커밋으로 되돌리기(reset) --hard

❖ 커밋 기록 확인하기

```
$ git log
```



```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:00:57 2023 +0900

    test commit 2

commit f93de75adde02cee5350a7a0f523a25f44e3a820
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 09:59:35 2023 +0900

    test commit 1

commit d2780be60aaa97f96d238578f41f76c24af697be
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:28:07 2023 +0900

    message3
```

[실습] 특정 커밋으로 되돌리기(reset) --hard

❖ test.txt 파일 확인

```
$ cat test.txt
```



```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ cat test.txt
a
b
```

특정 커밋으로 되돌리기(revert)

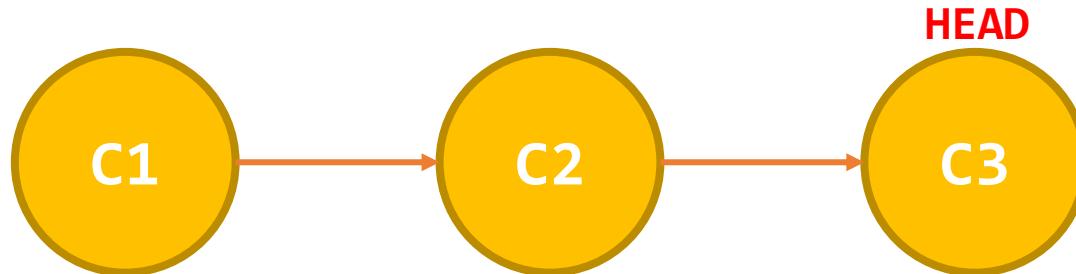
❖ 특정 커밋으로 되돌리기

```
$ git revert 취소할_커밋해시
```

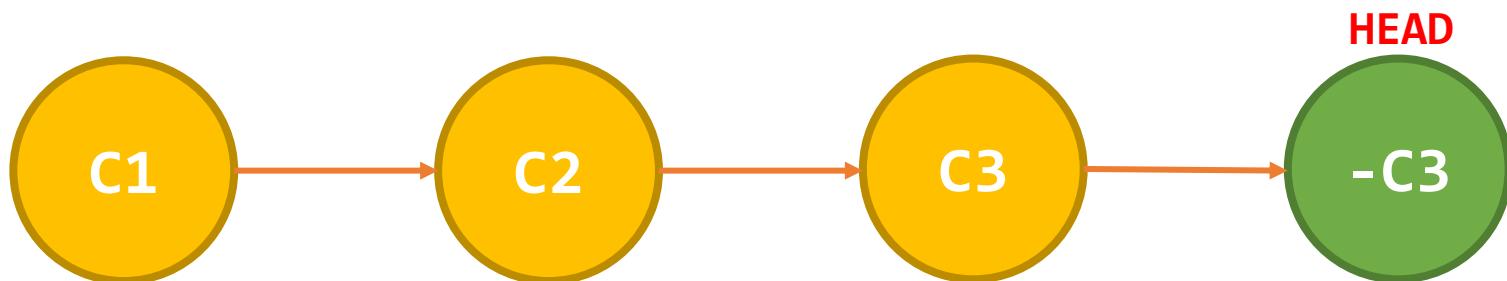
- 기존 커밋을 삭제하지 않고, 특정 커밋으로 되돌리는 새로운 커밋을 추가하는 방식
- 커밋 기록을 변경하지 않고, 안전하게 작업을 취소하는 방법
- 협업 환경에서 주로 사용

특정 커밋으로 되돌리기(revert)

❖ C1 커밋으로 되돌아가기



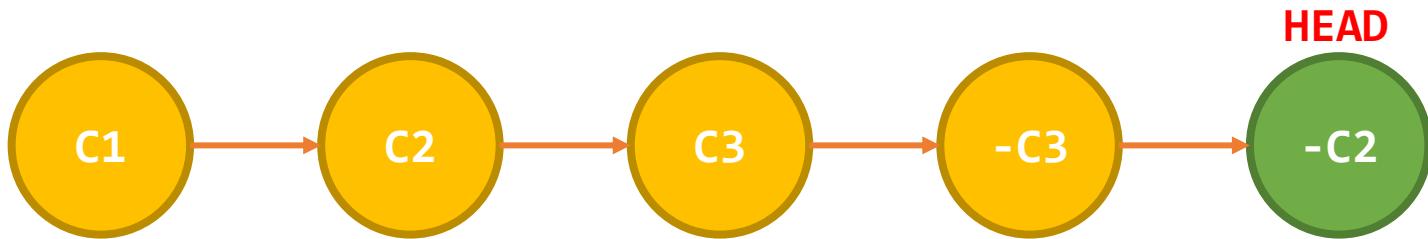
- C3 커밋 취소



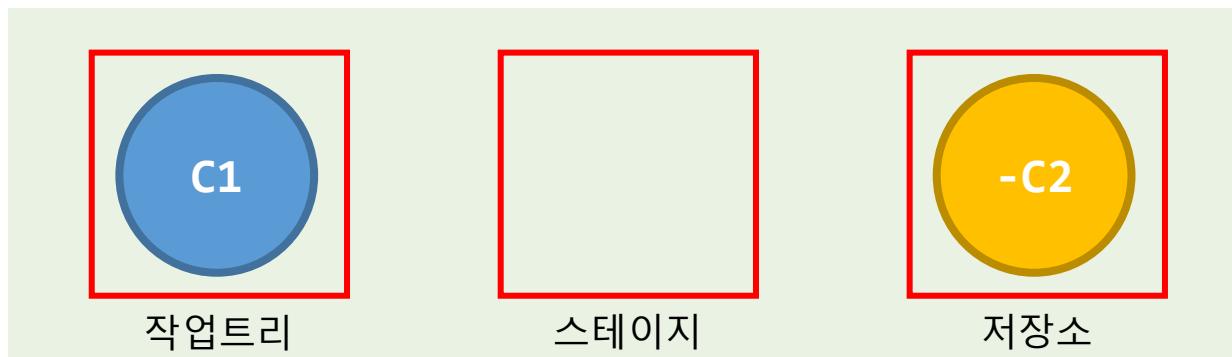
특정 커밋으로 되돌리기(revert)

❖ C1 커밋으로 되돌아가기

- C2 커밋 취소



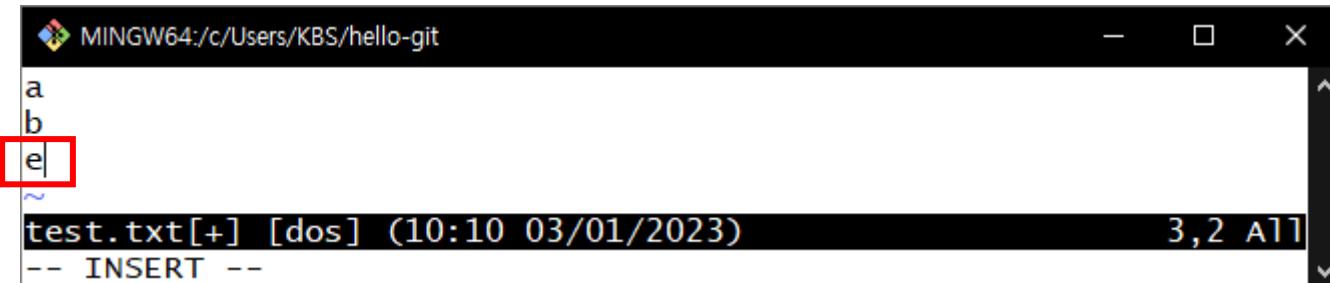
- 취소한 커밋의 기록을 남기면서 C1 커밋으로 돌아갈 수 있음



[실습] 특정 커밋으로 되돌리기(revert)

- ❖ test.txt 파일에 내용 추가하기(e)

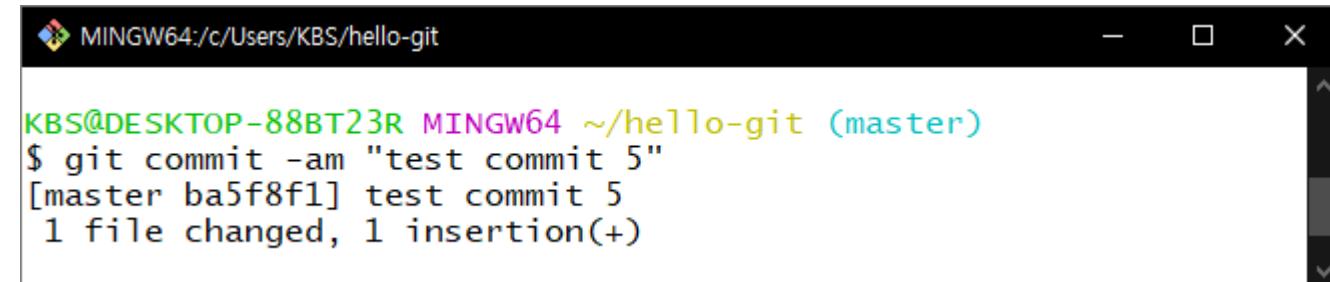
```
$ vim test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
a
b
e
~
test.txt[+] [dos] (10:10 03/01/2023) 3,2 All
-- INSERT --
```

- ❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "test commit 5"
```



```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "test commit 5"
[master ba5f8f1] test commit 5
1 file changed, 1 insertion(+)
```

[실습] 특정 커밋으로 되돌리기(revert)

❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit ba5f8f120021eac6808bc0ee5a360c3192496523 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:26:38 2023 +0900

    test commit 5

commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:00:57 2023 +0900

    test commit 2

commit f93de75adde02cee5350a7a0f523a25f44e3a820
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 09:59:35 2023 +0900

    test commit 1

commit d2780be60aaa97f96d238578f41f76c24af697be
Author: HelloCoding22 <HelloCoding22@gmail.com>
```

[실습] 특정 커밋으로 되돌리기(revert)

- ❖ "test commit 5"를 취소하고 "test commit 2"로 돌아가기
 - "test commit 5"의 커밋 해시를 복사

```
$ git revert ba5f8f120021eac6808bc0ee5a360c3192496523
```

The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The command 'git revert ba5f8f120021eac6808bc0ee5a360c3192496523' has been entered. A red box highlights the commit hash 'ba5f8f120021eac6808bc0ee5a360c3192496523' in the command line. Another red box highlights the message 'Revert "test commit 5"' in the title bar. The terminal output shows the revert process starting with a commit message template and listing modified files.

```
revert한 버전명  
커밋 메시지 입력(옵션)  
Revert "test commit 5"  
커밋 메시지 작성 : 커밋 보류 함  
This reverts commit ba5f8f120021eac6808bc0ee5a360c3192496523.  
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the comm  
it.  
#  
# On branch master  
# Changes to be committed:  
#       modified:   test.txt  
#  
~  
~  
~  
~  
~  
<lo-git/.git/COMMIT_EDITMSG[+] [unix] (10:32 03/01/2023)7,31 All  
-- INSERT --
```

[실습] 특정 커밋으로 되돌리기(revert)

❖ test.txt 파일 확인

```
$ cat test.txt
```



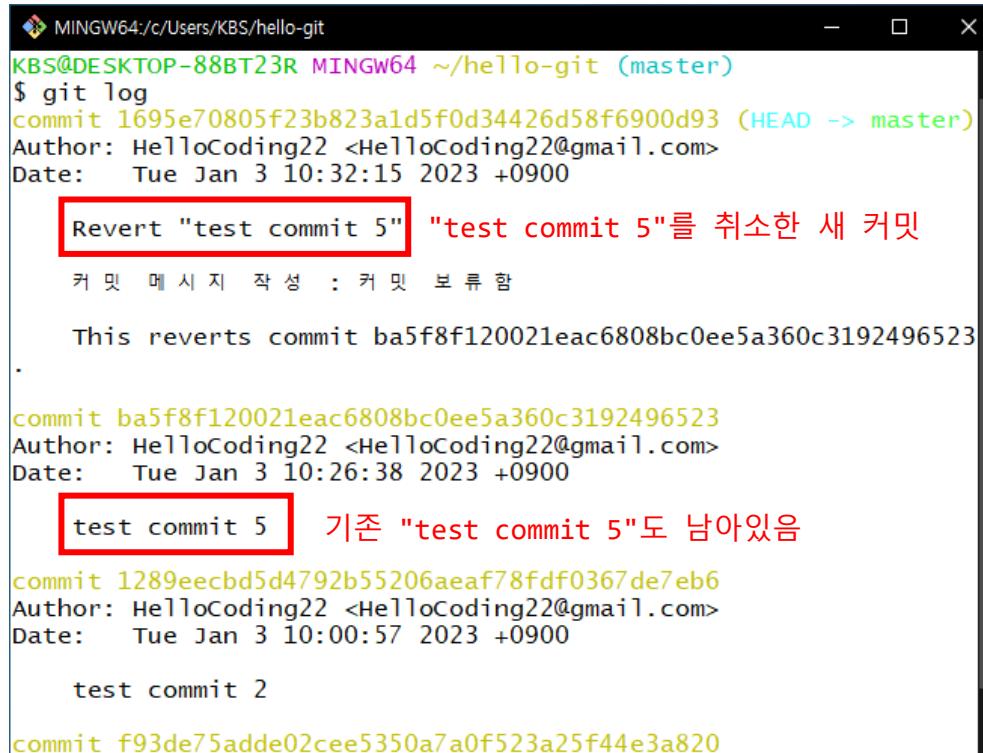
A screenshot of a terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The window shows the command 'KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)' followed by the output of the command '\$ cat test.txt', which displays the letters 'a' and 'b' on separate lines.

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ cat test.txt
a
b
```

[실습] 특정 커밋으로 되돌리기(revert)

❖ 커밋 기록 확인하기

```
$ git log
```



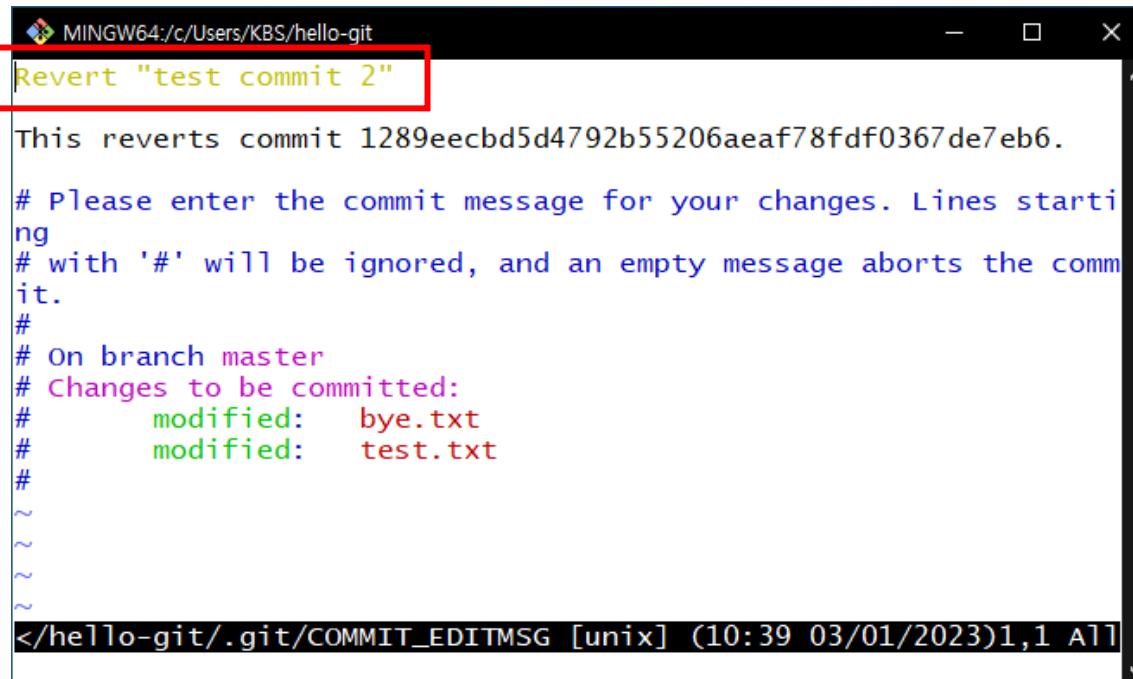
```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit 1695e70805f23b823a1d5f0d34426d58f6900d93 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:32:15 2023 +0900
    Revert "test commit 5" "test commit 5"를 취소한 새 커밋
    커밋 메시지 작성 : 커밋 보류함
    This reverts commit ba5f8f120021eac6808bc0ee5a360c3192496523
.

commit ba5f8f120021eac6808bc0ee5a360c3192496523
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:26:38 2023 +0900
    test commit 5 기존 "test commit 5"도 남아있음
commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:00:57 2023 +0900
    test commit 2
commit f93de75adde02cee5350a7a0f523a25f44e3a820
```

[실습] 특정 커밋으로 되돌리기(revert)

- ❖ "test commit 2"를 취소하고 "test commit 1"로 돌아가기
 - "test commit 2"의 커밋 해시를 복사

```
$ git revert 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The title bar has a red box around the text 'Revert "test commit 2"'. The main area of the terminal contains the following text:

```
This reverts commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6.  
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the comm  
it.  
#  
# On branch master  
# Changes to be committed:  
#     modified:   bye.txt  
#     modified:   test.txt  
#  
~  
~  
~  
~  
~</hello-git/.git/COMMIT_EDITMSG [unix] (10:39 03/01/2023)1,1 All>
```

[실습] 특정 커밋으로 되돌리기(revert)

❖ test.txt 파일 확인

```
$ cat test.txt
```



A screenshot of a terminal window titled "MINGW64:/c/Users/KBS/hello-git". The window shows the command \$ cat test.txt followed by the output "a".

```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ cat test.txt
a
```

[실습] 특정 커밋으로 되돌리기(revert)

❖ 커밋 기록 확인하기

```
$ git log
```

```
commit e72d72fb6bfeb2a8630052490831af46ee2463bd (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:39:36 2023 +0900

    Revert "test commit 2"      test commit 2를 취소한 새 커밋

    This reverts commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
.

commit 1695e70805f23b823a1d5f0d34426d58f6900d93
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:32:15 2023 +0900

    Revert "test commit 5"

    커밋 메시지 작성 : 커밋 보류함

    This reverts commit ba5f8f120021eac6808bc0ee5a360c3192496523
.

commit ba5f8f120021eac6808bc0ee5a360c3192496523
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Tue Jan 3 10:26:38 2023 +0900

:
```

08

정리하기

정리하기

git init	현재 위치에 저장소 생성하기
git status	Git 상태 확인하기
git add test.txt	test.txt 파일을 스테이지에 올리기
git add .	수정한 전체 파일을 스테이징
git commit -m "first commit"	first commit 메시지와 함께 커밋하기
git log	커밋 기록 확인하기
git reset HEAD^	최신 커밋 취소하기
git reset 옵션 커밋해시	지정한 커밋해시로 이동 후, 이후 커밋 취소하기
git revert 커밋해시	지정한 커밋해시의 변경이력을 남기면서 취소하기

Git & GitHub

◆ 브랜치

정수아

Contents

01 브랜치란?

02 초기 작업하기

03 브랜치 만들기

04 브랜치 정보 확인하기

05 브랜치 병합하기

06 정리하기

01

브랜치란?

브랜치란?

❖ 브랜치(Branch)

- 독립적으로 작업을 진행할 수 있도록 해주는 기능
 - 메인 작업 흐름(master, main)에서 나와, 다른 작업을 분리해서 진행할 수 있도록 함
 - 하나의 프로젝트에서 여러 개발자가 서로 다른 작업을 진행 시, 서로의 작업에 영향을 주지 않기 위해 필요
- 특징
 - 분기(branch)와 병합(merge) 기능

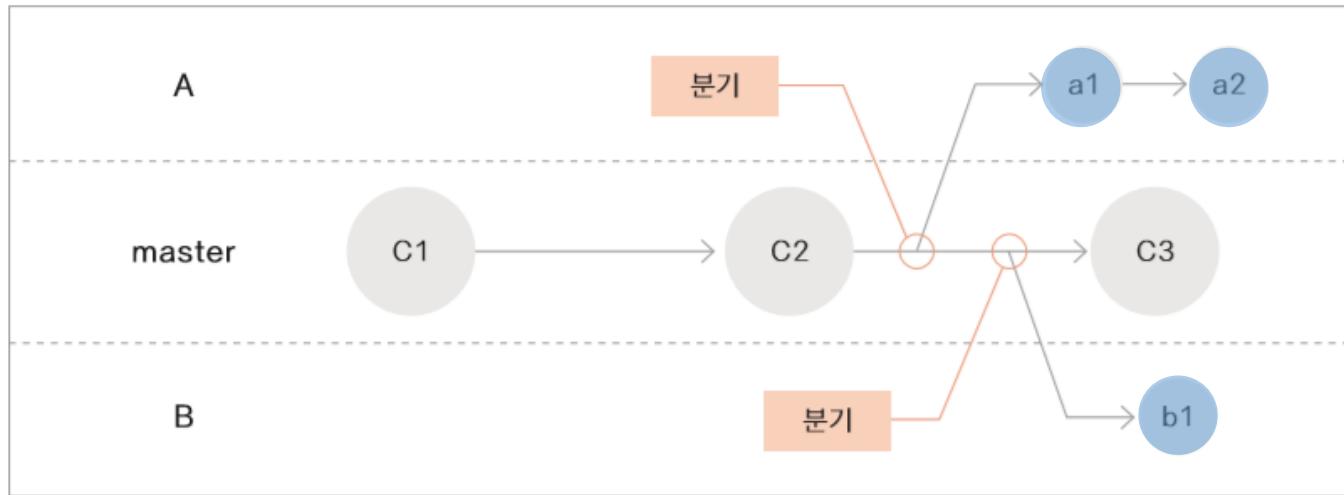
❖ 기준 브랜치

- main 또는 master 브랜치
- Git으로 버전 관리를 시작하면 기본적으로 생성

브랜치 기능

❖ 분기(branch)

- master 브랜치에서 새로운 브랜치(A, B)를 생성

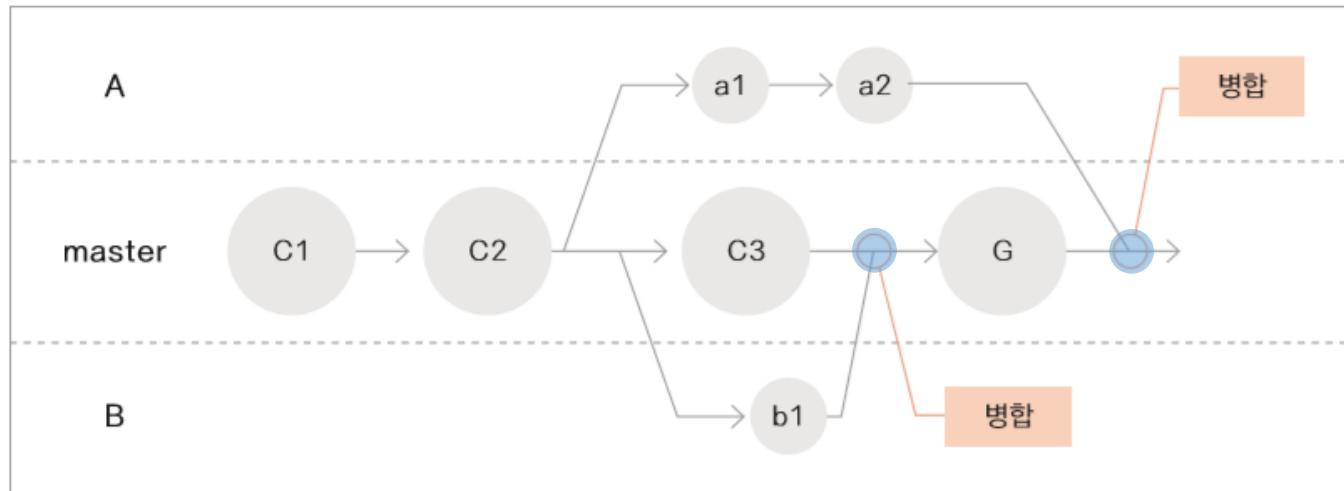


- 분기된 브랜치의 특징
 - 분기되기 이전에 master가 작업한 내용 유지(C1, C2)
 - 새로운 기능 생성 가능(a1, a2)

브랜치 기능

❖ 병합(merge)

- 새로운 브랜치(A, B)의 작업 내용을 master 브랜치에 합침



02

초기 작업하기

초기 작업하기

❖ 디렉터리 생성

```
$ mkdir manual  
$ cd manual
```

❖ git 초기화

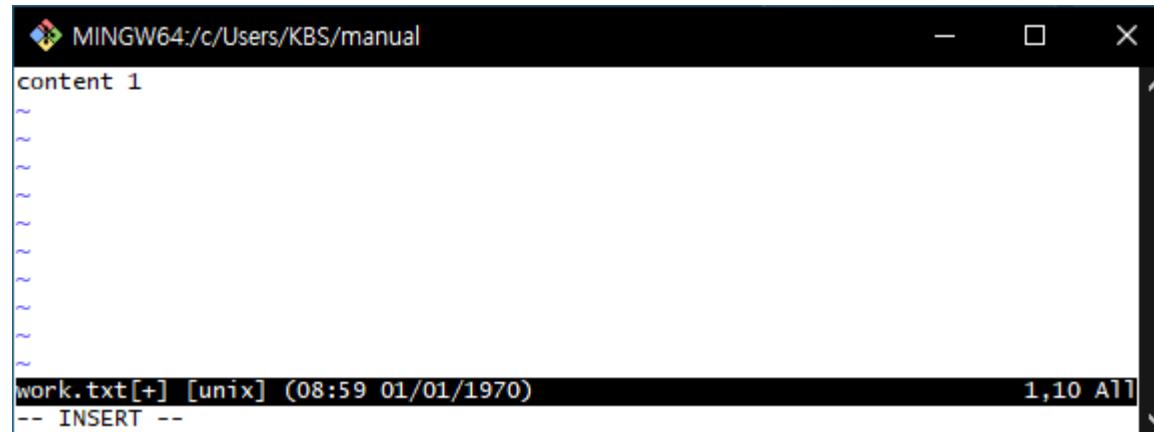
- 초기화 후 .git 디렉터리(숨김 폴더)가 생성

```
$ git init
```

초기 작업하기

❖ 파일 생성 후 내용 입력(content 1)

```
$ vim work.txt
```



A screenshot of a terminal window titled "MINGW64:/c/Users/KBS/manual". The window contains the text "content 1" followed by ten blank lines. The status bar at the bottom shows "work.txt[+]" and "(08:59 01/01/1970)". On the right side of the status bar, there are status icons for line numbers (1,10 All) and mode (-- INSERT --). The terminal has a dark theme with light-colored text.

초기 작업하기

❖ 스테이지에 올리기

```
$ git add work.txt
```

❖ 커밋하기

```
$ git commit -m "work1"
```

❖ 내용 입력 및 커밋하기 두 번 반복

초기 작업하기

❖ 커밋 내역 확인하기

```
$ git log
```



```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (master)
$ git log
commit 901cb57d43d50befa0b8b06573e9db59b1e27a8c [HEAD -> master]
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:51:15 2023 +0900

    work3

commit e480818a23f5b2d93de19ceebc436a08c3fc56ad
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:51:03 2023 +0900

    work2

commit fd2028dc6aa410e41d66e6d1dcb25d015bb60891
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:50:20 2023 +0900

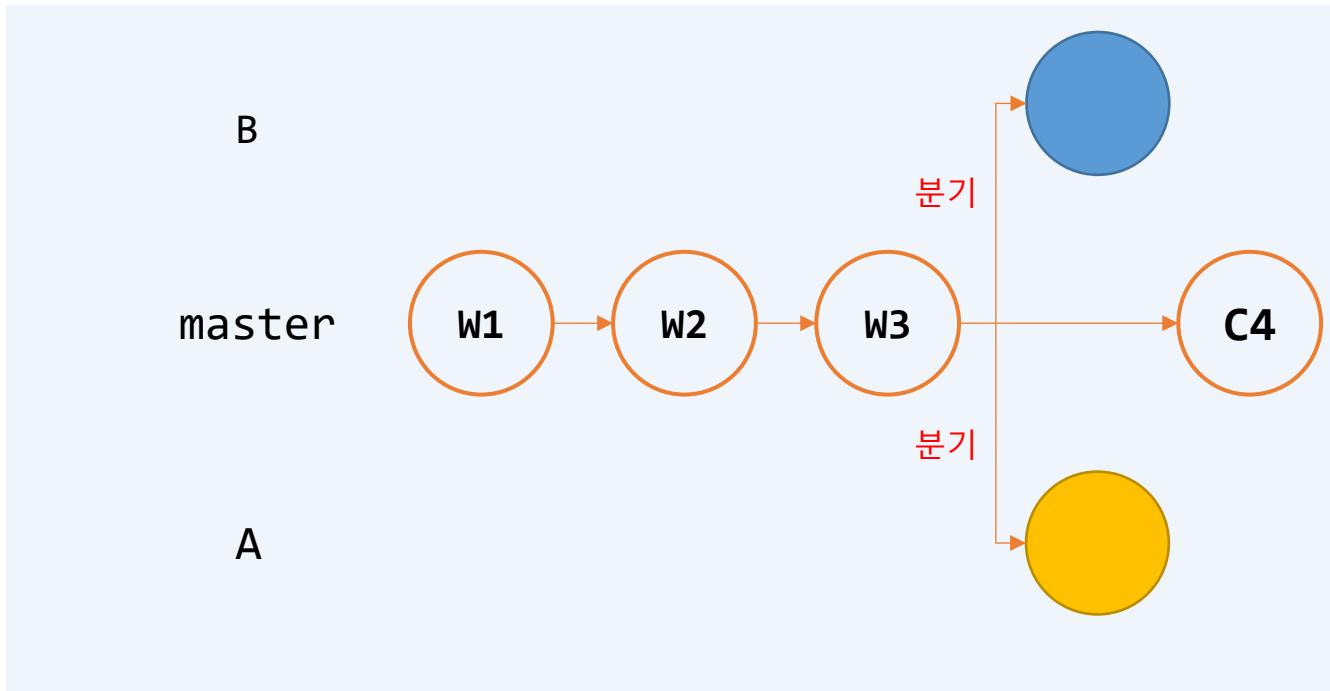
    work1
```

03

브랜치 만들기

브랜치 만들기

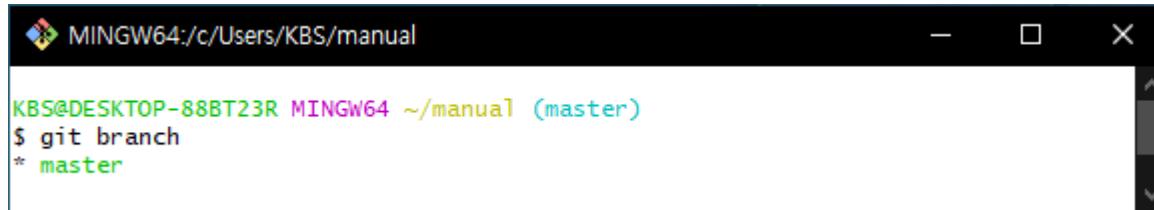
❖ 브랜치 분기



브랜치 만들기

❖ 브랜치 확인

```
$ git branch
```



A screenshot of a terminal window titled "MINGW64:/c/Users/KBS/manual". The window shows the command \$ git branch and its output, which includes the current branch (* master).

```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (master)
$ git branch
* master
```

❖ 새 브랜치 생성

```
$ git branch 브랜치이름
```

브랜치 만들기

❖ A 브랜치 생성

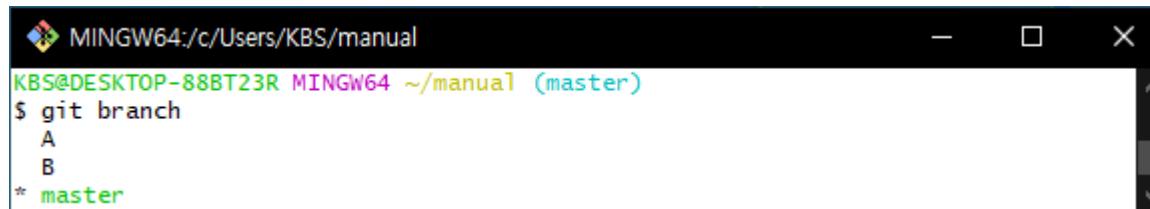
```
$ git branch A
```

❖ B 브랜치 생성

```
$ git branch B
```

❖ 생성된 브랜치 확인

```
$ git branch
```



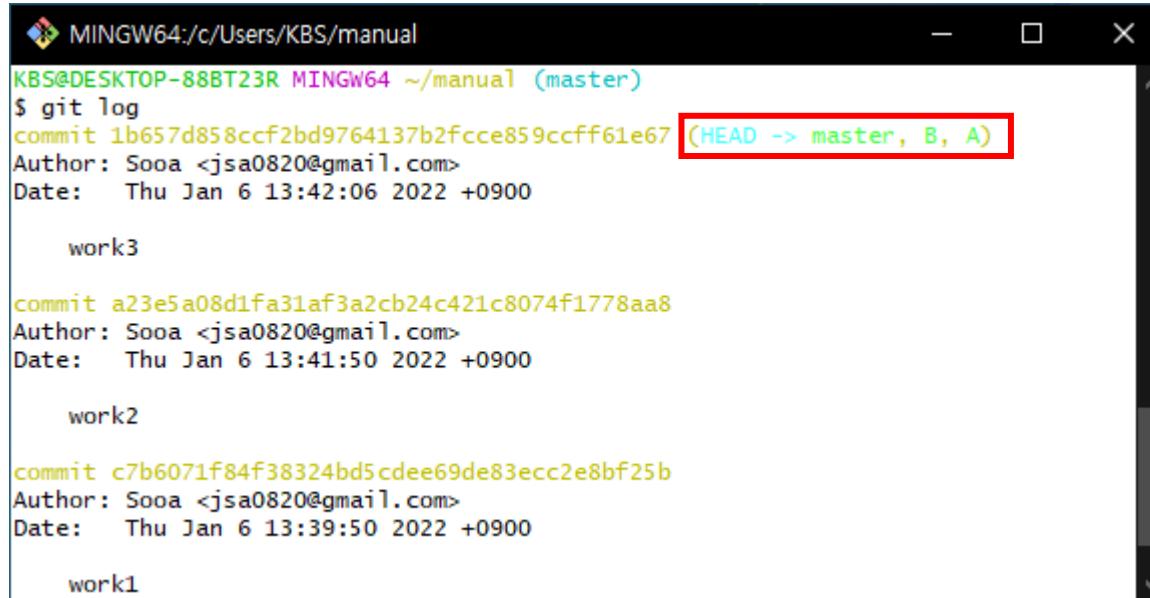
The screenshot shows a terminal window with the following text:

```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (master)
$ git branch
A
B
* master
```

브랜치 만들기

❖ 커밋 내역 확인하기

```
$ git log
```



```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (master)
$ git log
commit 1b657d858ccf2bd9764137b2fcce859ccff61e67 (HEAD -> master, B, A)
Author: Sooa <jsa0820@gmail.com>
Date:   Thu Jan 6 13:42:06 2022 +0900

    work3

commit a23e5a08d1fa31af3a2cb24c421c8074f1778aa8
Author: Sooa <jsa0820@gmail.com>
Date:   Thu Jan 6 13:41:50 2022 +0900

    work2

commit c7b6071f84f38324bd5cdee69de83ecc2e8bf25b
Author: Sooa <jsa0820@gmail.com>
Date:   Thu Jan 6 13:39:50 2022 +0900

    work1
```

브랜치 만들기

❖ 새로운 내용 추가 및 커밋하기

- work.txt 파일에 content4 추가 후 커밋(C4)

```
$ git commit -am "master commit content 4"
```

브랜치 만들기

❖ 커밋 내역 확인하기

```
$ git log
```

```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (master)
$ git log
commit 19caa7e78e79f5089a456b33495f689cd660f975 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:57:05 2023 +0900

    master commit content 4

commit 901cb57d43d50befa0b8b06573e9db59b1e27a8c (B, A)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:51:15 2023 +0900

    work3

commit e480818a23f5b2d93de19ceebc436a08c3fc56ad
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:51:03 2023 +0900

    work2

commit fd2028dc6aa410e41d66e6d1dcb25d015bb60891
```

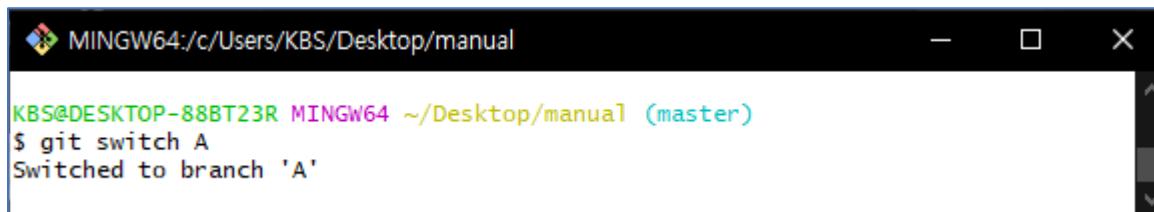
브랜치 변경 후 작업하기

❖ 브랜치 이동하기

```
$ git switch 브랜치이름
```

❖ A 브랜치로 이동하기

```
$ git switch A
```



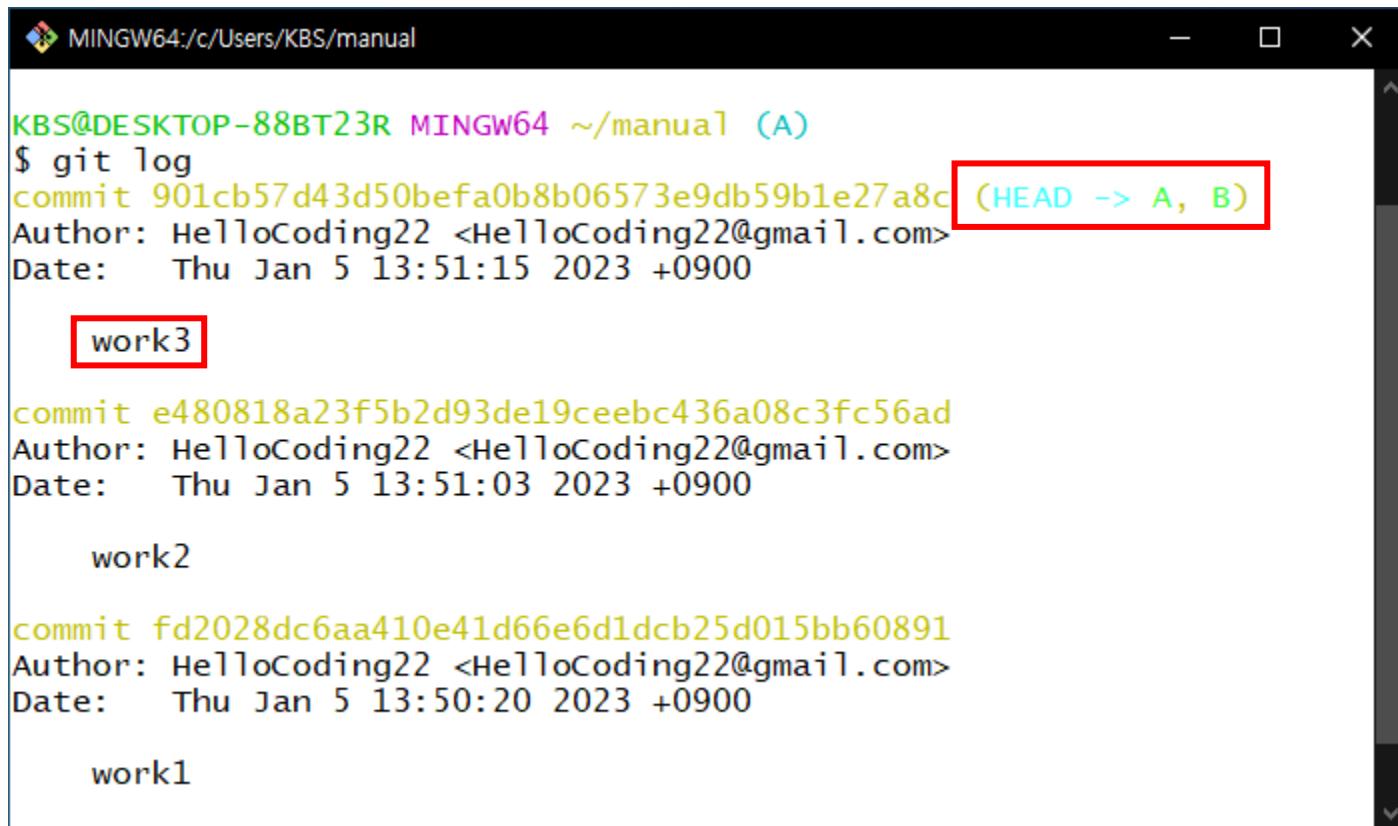
A screenshot of a terminal window titled 'MINGW64:/c/Users/KBS/Desktop/manual'. The window shows the command \$ git switch A and its output 'Switched to branch 'A''. The terminal has a dark theme with light-colored text.

```
MINGW64:/c/Users/KBS/Desktop/manual
KBS@DESKTOP-88BT23R MINGW64 ~/Desktop/manual (master)
$ git switch A
Switched to branch 'A'
```

브랜치 변경 후 작업하기

❖ 커밋 내역 확인하기 - 현재 브랜치 : A

```
$ git log
```



```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (A)
$ git log
commit 901cb57d43d50befa0b8b06573e9db59b1e27a8c (HEAD -> A, B)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:51:15 2023 +0900

    work3

commit e480818a23f5b2d93de19ceebc436a08c3fc56ad
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:51:03 2023 +0900

    work2

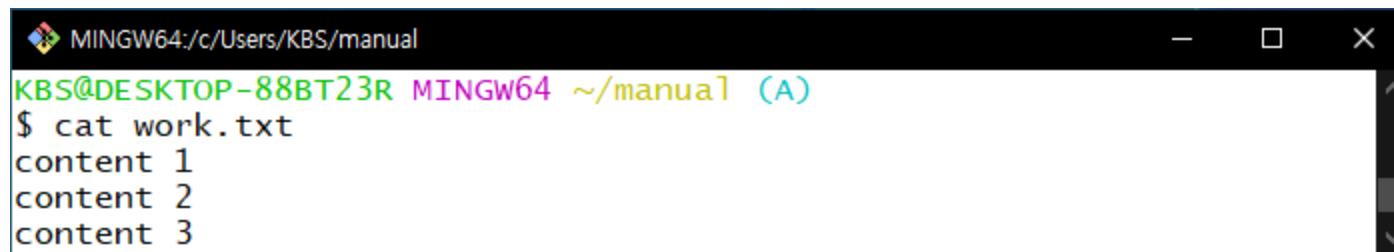
commit fd2028dc6aa410e41d66e6d1dcb25d015bb60891
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Thu Jan 5 13:50:20 2023 +0900

    work1
```

브랜치 변경 후 작업하기

- ❖ work.txt 파일 확인하기 - 현재 브랜치 : A

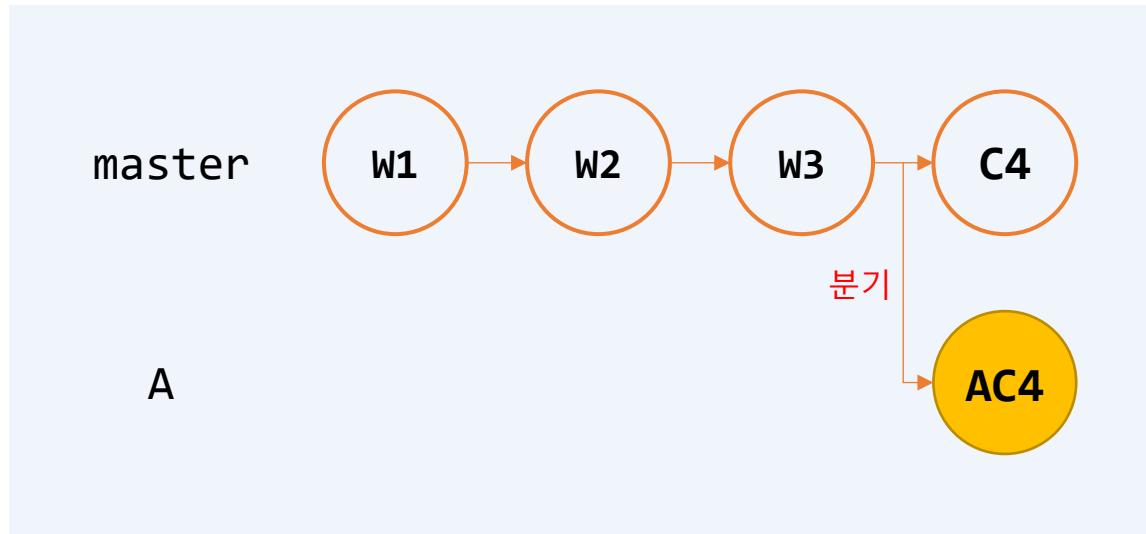
```
$ cat work.txt
```



```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (A)
$ cat work.txt
content 1
content 2
content 3
```

브랜치 변경 후 작업하기

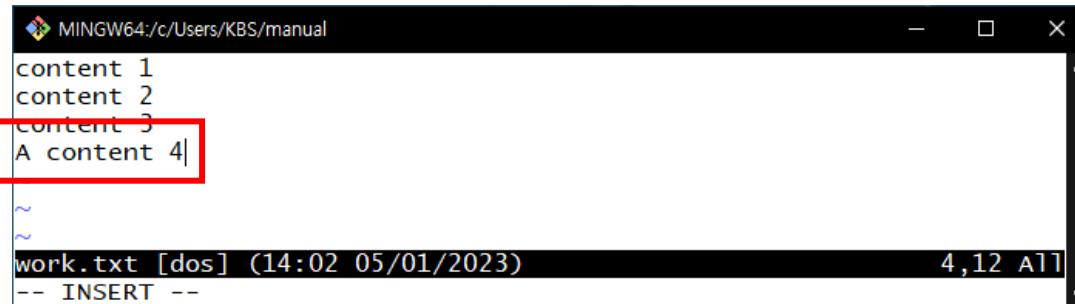
- ❖ 새로운 커밋 생성하기 - 현재 브랜치 : A



브랜치 변경 후 작업하기

- ❖ A 브랜치의 work.txt 파일에 내용(A content 4) 추가

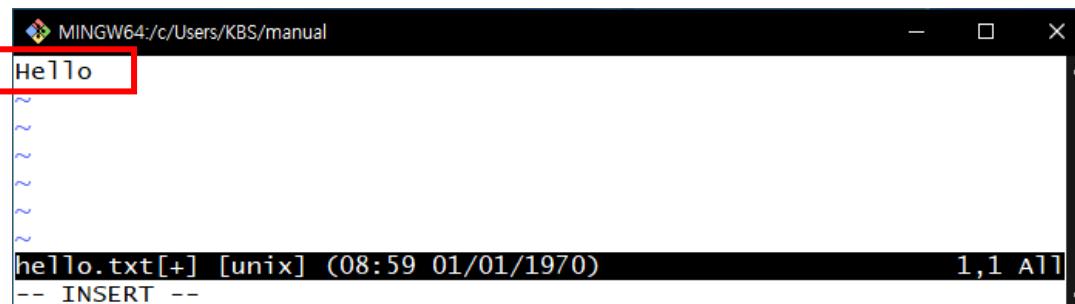
```
$ vim work.txt
```



```
MINGW64:/c/Users/KBS/manual
content 1
content 2
content 3
A content 4

~
~
work.txt [dos] (14:02 05/01/2023) 4,12 All
-- INSERT --
```

- ❖ 새 파일(hello.txt) 생성 및 내용(Hello) 추가



```
MINGW64:/c/Users/KBS/manual
Hello

~
~
~
~
~
hello.txt[+] [unix] (08:59 01/01/1970) 1,1 All
-- INSERT --
```

브랜치 변경 후 작업하기

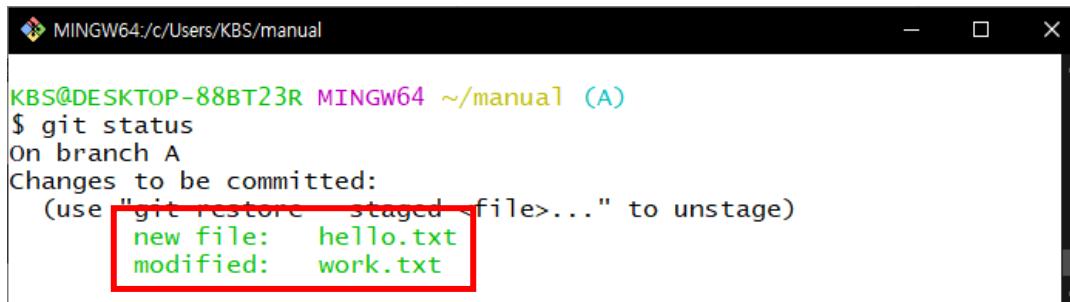
❖ 수정된 파일을 스테이지에 올리기

```
$ git add work.txt  
$ git add hello.txt
```

또는

```
$ git add .
```

- git add 명령어 뒤에 마침표(.)를 붙이면 현재 저장소에서 수정된 파일이 한번에 스테이지에 올라감

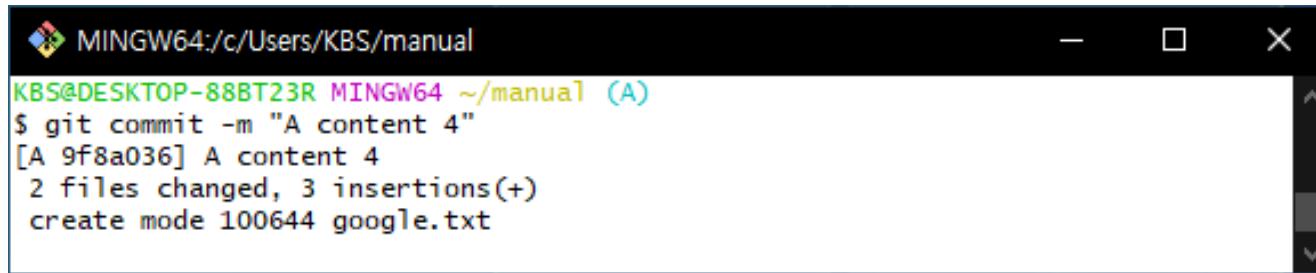


```
MINGW64:c/Users/KBS/manual  
KBS@DESKTOP-88BT23R MINGW64 ~/manual (A)  
$ git status  
On branch A  
Changes to be committed:  
(use "git restore --staged <file>..." to unstage)  
  new file:   hello.txt  
  modified:  work.txt
```

브랜치 변경 후 작업하기

❖ 커밋하기

```
$ git commit -m "A content 4"
```



A screenshot of a terminal window titled 'MINGW64:/c/Users/KBS/manual'. The window shows the command \$ git commit -m "A content 4" followed by its output: [A 9f8a036] A content 4, 2 files changed, 3 insertions(+), and create mode 100644 google.txt.

```
KBS@DESKTOP-88BT23R MINGW64 ~/manual (A)
$ git commit -m "A content 4"
[A 9f8a036] A content 4
2 files changed, 3 insertions(+)
create mode 100644 google.txt
```

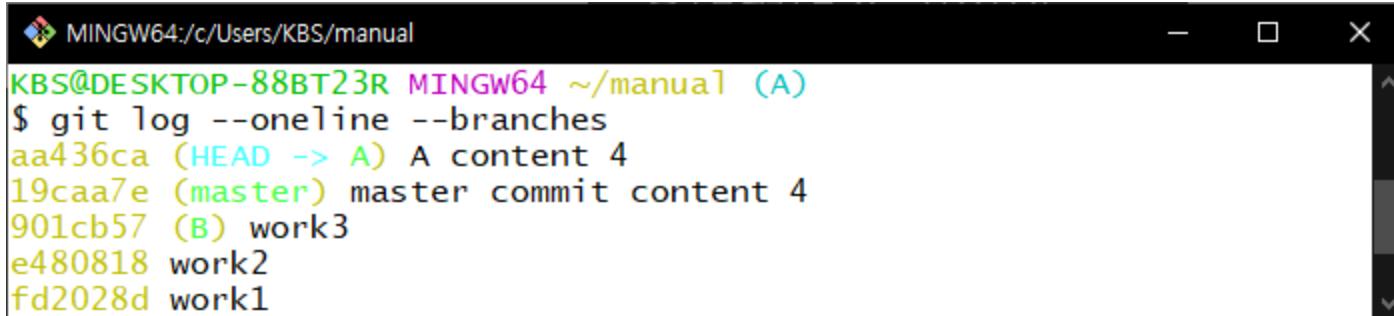
04

브랜치 정보 확인하기

브랜치 정보 확인하기

❖ 브랜치마다 커밋 로그 확인하기

```
$ git log --oneline --branches
```



The screenshot shows a terminal window with the following text:

```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (A)
$ git log --oneline --branches
aa436ca (HEAD -> A) A content 4
19caa7e (master) master commit content 4
901cb57 (B) work3
e480818 work2
fd2028d work1
```

- A 브랜치 : A content 4
- master 브랜치 : master commit content 4
- B 브랜치 : work3

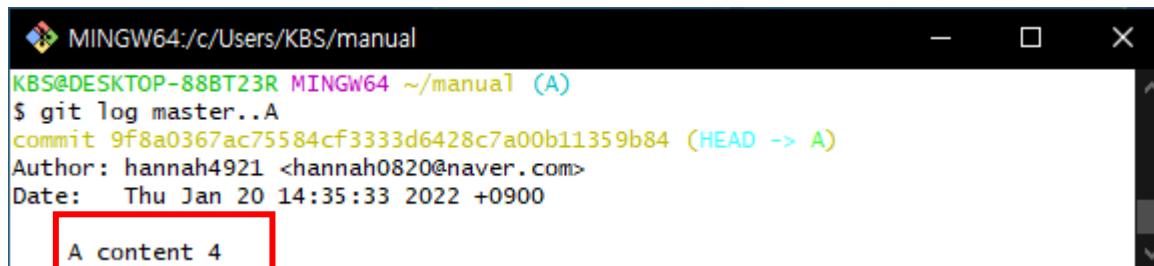
브랜치 정보 확인하기

❖ 브랜치 사이의 차이점 확인하기

```
$ git log 기준_브랜치..비교_브랜치
```

- 예) master 브랜치와 A 브랜치를 비교

```
$ git log master..A
```



```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (A)
$ git log master..A
commit 9f8a0367ac75584cf3333d6428c7a00b11359b84 (HEAD -> A)
Author: hannah4921 <hannah0820@naver.com>
Date:   Thu Jan 20 14:35:33 2022 +0900
    A content 4
```

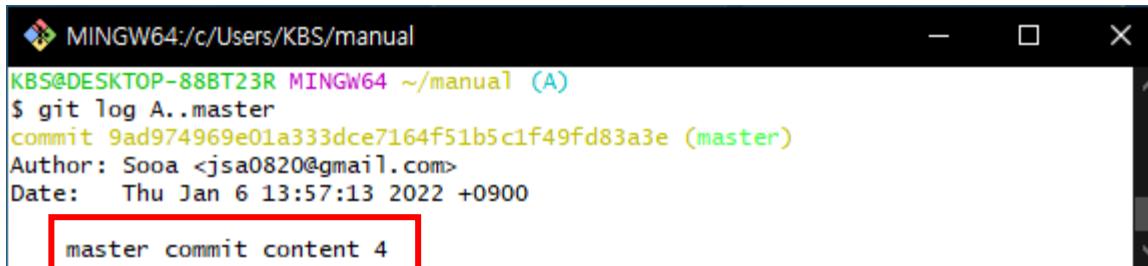
- master 브랜치에는 없고, A 브랜치에는 있는 커밋을 보여줌

브랜치 정보 확인하기

❖ 브랜치 사이의 차이점 확인하기

- 예) master 브랜치와 A 브랜치를 비교

```
$ git log A..master
```



```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (A)
$ git log A..master
commit 9ad974969e01a333dce7164f51b5c1f49fd83a3e (master)
Author: Sooa <jsa0820@gmail.com>
Date:   Thu Jan 6 13:57:13 2022 +0900

    master commit content 4
```

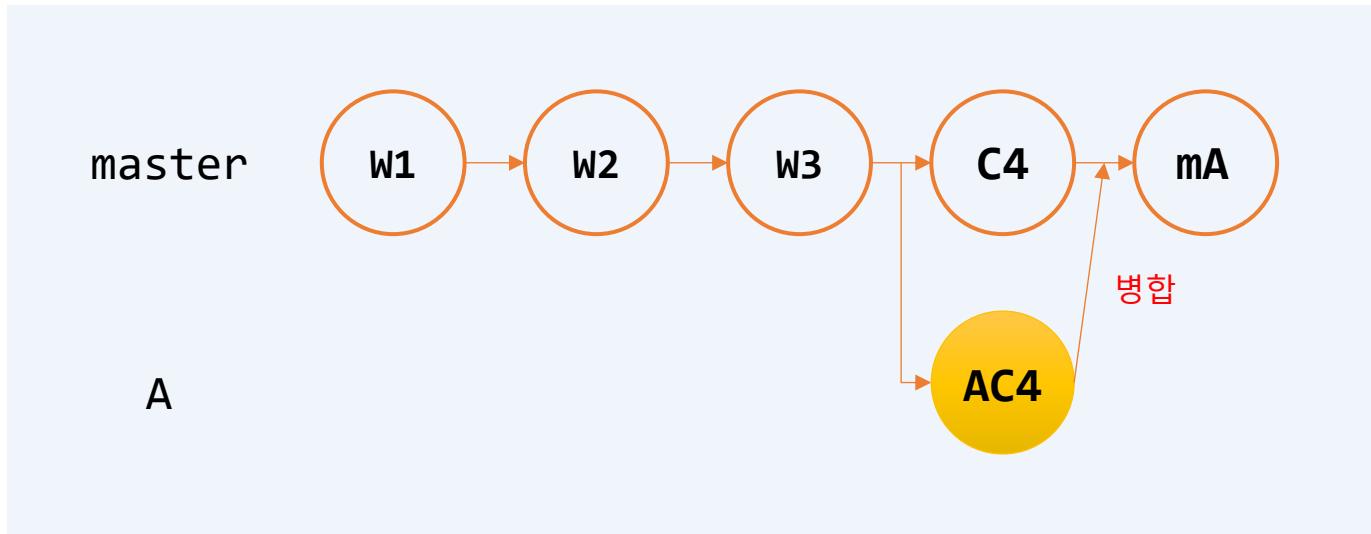
- A 브랜치에는 없고, master 브랜치에는 있는 커밋을 보여줌

05

브랜치 병합하기

브랜치 병합하기

❖ 브랜치 병합



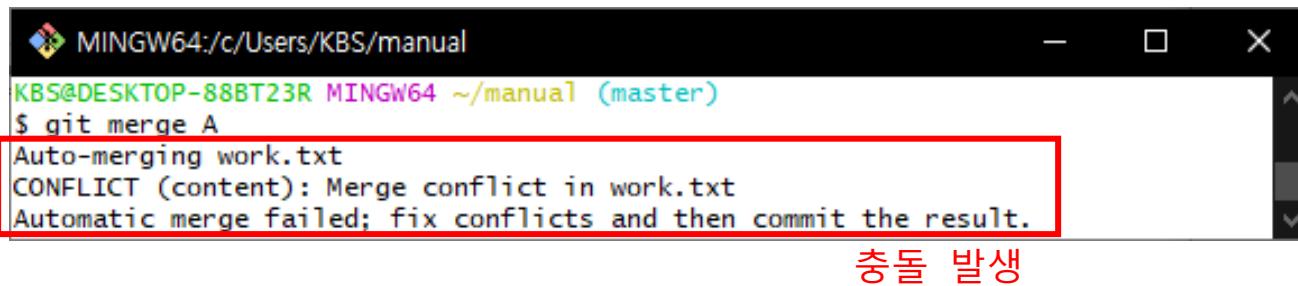
브랜치 병합하기

❖ master 브랜치로 이동하기

```
$ git switch master
```

❖ master 브랜치에 A 브랜치 병합하기

```
$ git merge A
```



The screenshot shows a terminal window with the following text:

```
MINGW64:/c/Users/KBS/manual
KBS@DESKTOP-88BT23R MINGW64 ~/manual (master)
$ git merge A
Auto-merging work.txt
CONFLICT (content): Merge conflict in work.txt
Automatic merge failed; fix conflicts and then commit the result.
```

A red box highlights the last three lines of the output, which indicate a merge conflict in the 'work.txt' file. To the right of the terminal window, the Korean text "충돌 발생" (Conflict Occurred) is displayed in red.

브랜치 병합하기

❖ work.txt 파일 내용 확인

```
content 1
```

```
content 2
```

```
content 3
```

```
<<<<<< HEAD
```

```
content 4 현재 브랜치에서 수정한 내용
```

```
=====
```

```
A content 4 병합할 브랜치에서 수정한 내용
```

```
>>>>> A
```

브랜치 병합하기

❖ 파일 내용을 원하는 대로 수정하고 저장

- <<<<< HEAD, =====, >>>>> A는 삭제

```
content 1
```

```
content 2
```

```
content 3
```

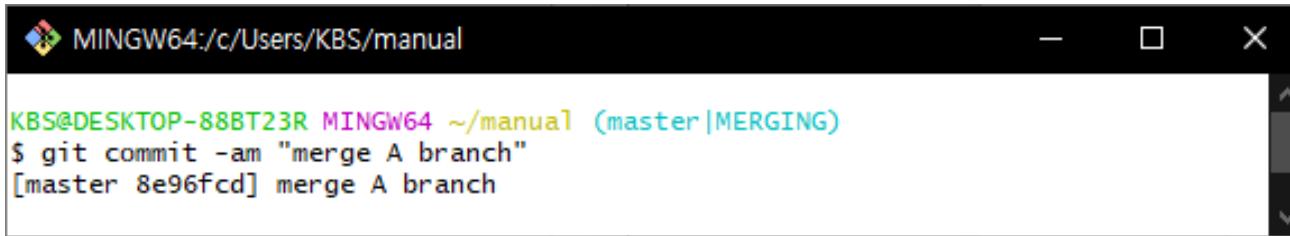
```
content 4
```

```
A content 4
```

브랜치 병합하기

- ❖ work.txt 파일 스테이징 및 커밋하기(mA)

```
$ git commit -am "merge A branch"
```



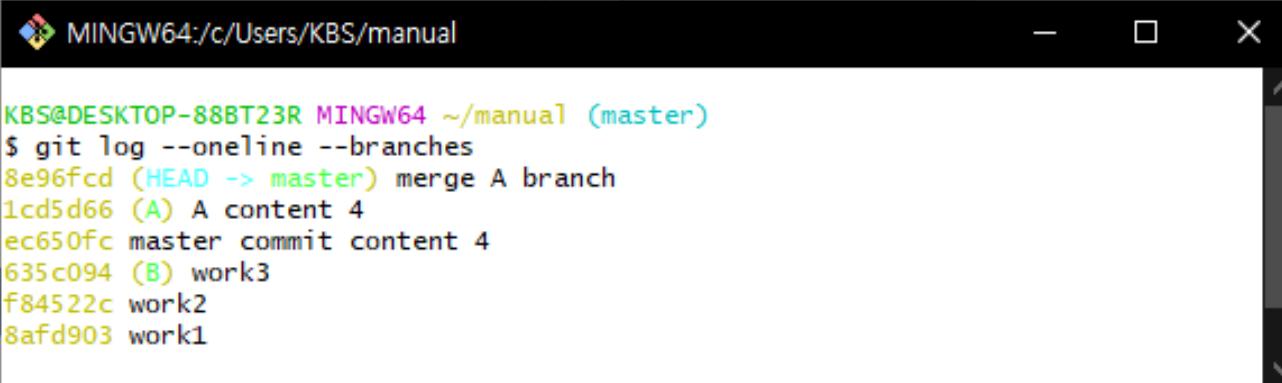
The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/manual'. The command '\$ git commit -am "merge A branch"' is entered, and the output shows it was successful, creating a new commit '[master 8e96fc] merge A branch'.

```
KBS@DESKTOP-88BT23R MINGW64 ~/manual (master |MERGING)
$ git commit -am "merge A branch"
[master 8e96fc] merge A branch
```

브랜치 병합하기

❖ 로그 확인하기

```
$ git log --oneline --branches
```



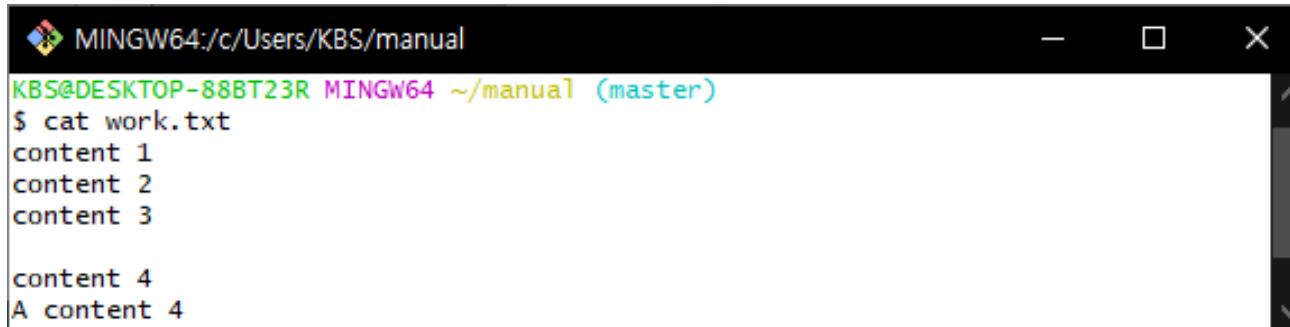
The screenshot shows a terminal window titled "MINGW64:/c/Users/KBS/manual". The window contains the following text:

```
KBS@DESKTOP-88BT23R MINGW64 ~/manual (master)
$ git log --oneline --branches
8e96fcd (HEAD -> master) merge A branch
1cd5d66 (A) A content 4
ec650fc master commit content 4
635c094 (B) work3
f84522c work2
8afd903 work1
```

브랜치 병합하기

❖ work.txt 파일 확인하기

```
$ cat work.txt
```



The screenshot shows a terminal window with a black background and white text. The title bar reads "MINGW64:/c/Users/KBS/manual". The command prompt is "KBS@DESKTOP-88BT23R MINGW64 ~/manual (master)". The user has run the command "\$ cat work.txt". The output of the command is displayed in red text:
content 1
content 2
content 3

content 4
A content 4

06

정리하기

정리하기

git branch newBranch	새로운 브랜치 newBranch 생성
git switch newBranch	기존 브랜치에서 newBranch로 변경
git log --oneline	각 커밋을 한 줄 요약 형식으로 출력
git log --oneline --branches	모든 브랜치의 커밋 기록 출력
git merge newBranch	newBranch를 master 브랜치에 병합

Git & GitHub

- ◆ GitHub로 백업하기

정수아

Contents

01 원격 저장소

02 GitHub란?

03 지역 저장소를 원격 저장소에 연결하기

01

원격 저장소

원격 저장소

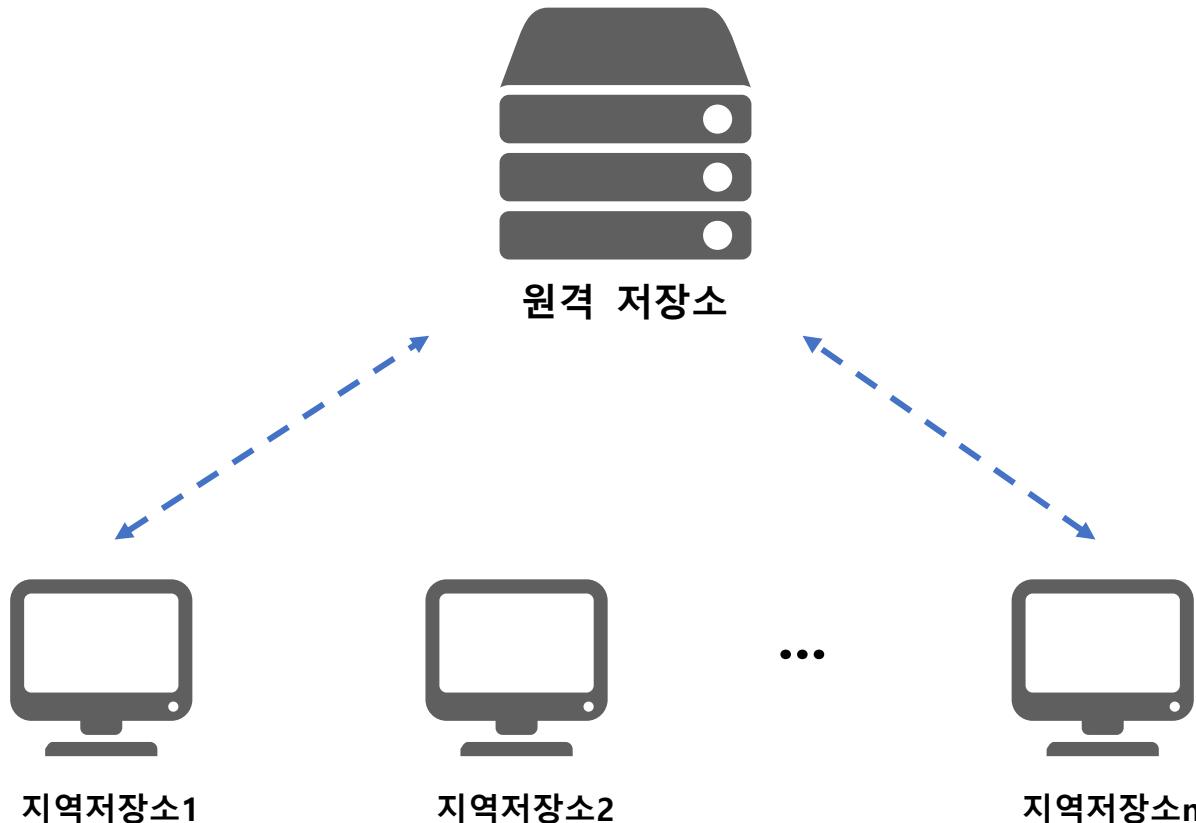
❖ 지역 저장소(local repository)

- 작업을 수행한 후 커밋을 저장한 컴퓨터

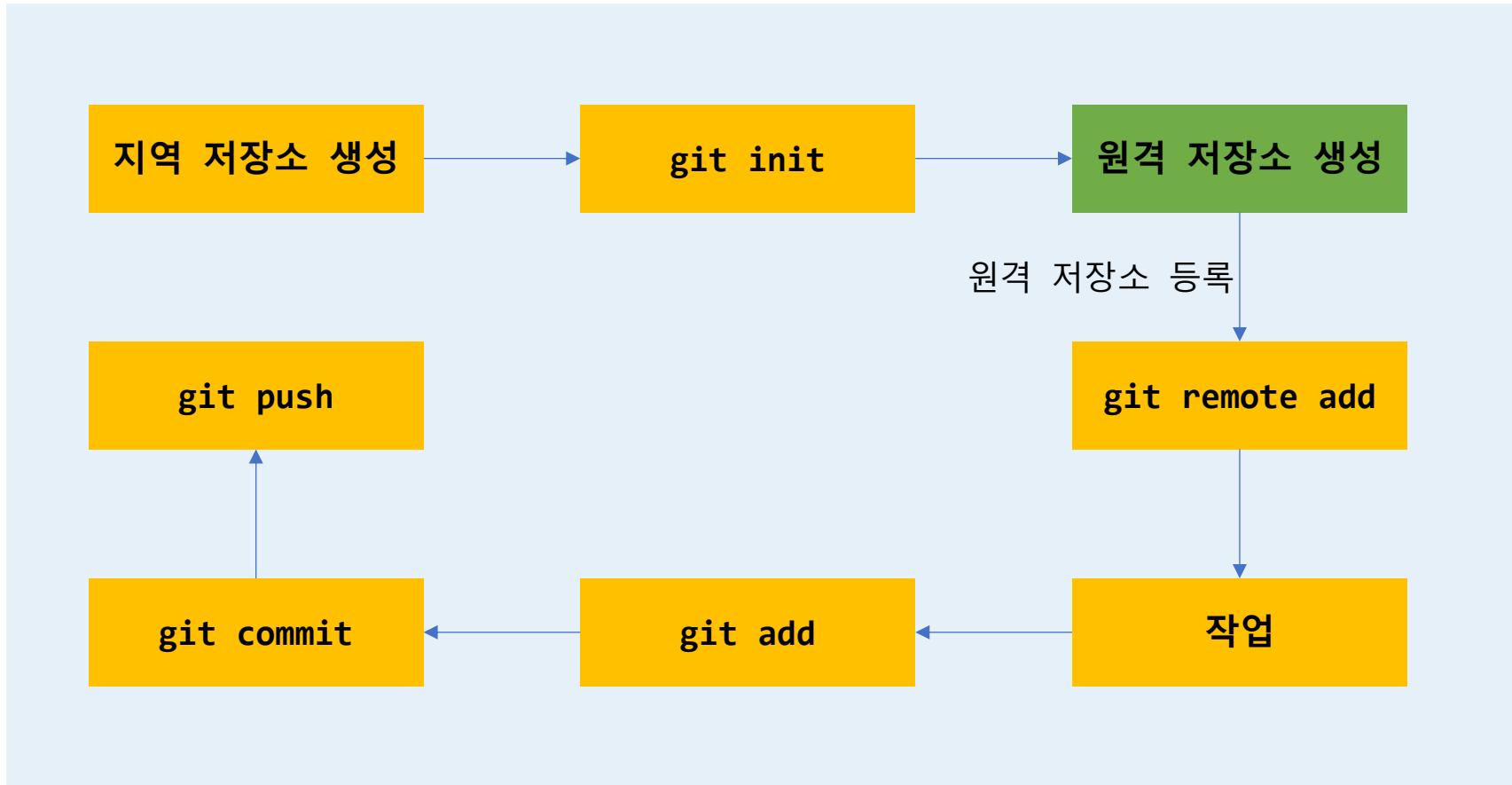
❖ 원격 저장소(remote repository)

- 지역 저장소가 아닌 컴퓨터나 서버에 만든 저장소
- 지역 저장소와 연결되어 있으면서 백업 및 협업을 가능하게 함
- 인터넷에서 원격 저장소를 제공하는 서비스를 주로 사용
 - GitHub

원격 저장소



원격 저장소



02

GitHub란?

GitHub란?

❖ GitHub

- Git과 관련해서 가장 많이 사용되는 원격 저장소 제공 서비스

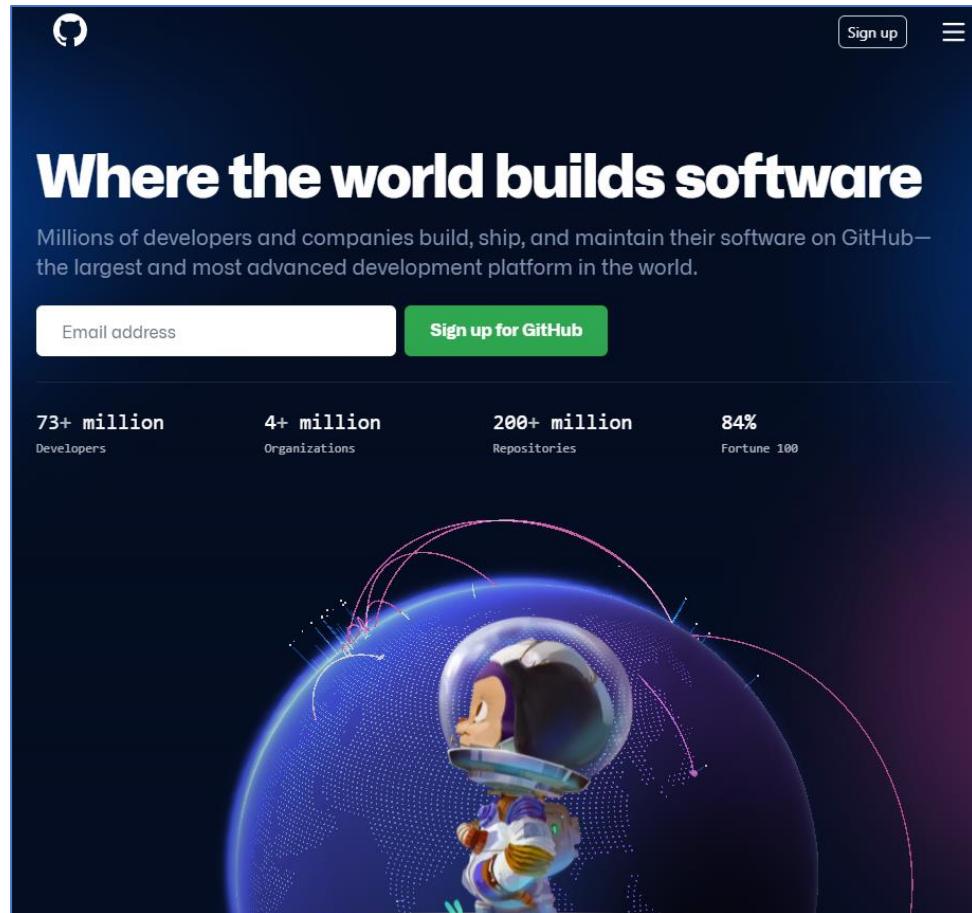
❖ 특징

- 무료 및 유료 서비스
- 다양한 오픈 소스 제공
- Git을 설치하지 않고도 온라인상에서 버전 관리 기능 사용 가능
- 지역 저장소를 온라인상에 백업 가능
- 협업 프로젝트에 사용 가능
- 개발 이력 기록 가능

GitHub 시작하기

❖ GitHub 가입

- www.github.com



GitHub 시작하기

❖ 정보 입력

Sign up to GitHub

Email*

Password*

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username*

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Your Country/Region*

▼

Korea, South

For compliance reasons, we're required to collect country information to send you occasional updates and announcements.

Continue >

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.



GitHub 시작하기

❖ 이메일 인증

Confirm your email address

We have sent a code to `asdjwdqka@gmail.com`

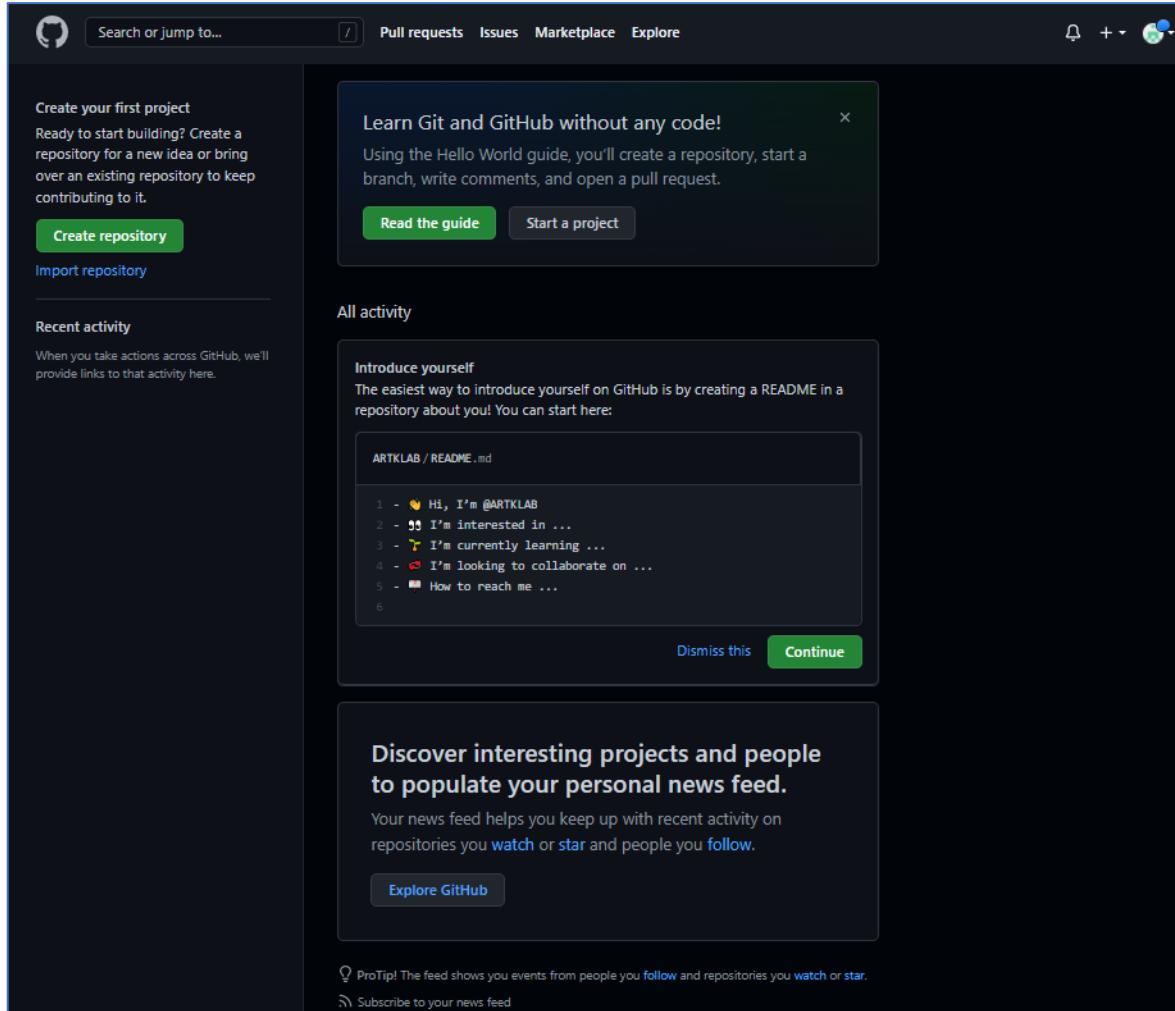
Enter code

Continue

Didn't get your email? [Resend the code](#) or [update your email address](#).

GitHub 시작하기

❖ 초기 화면



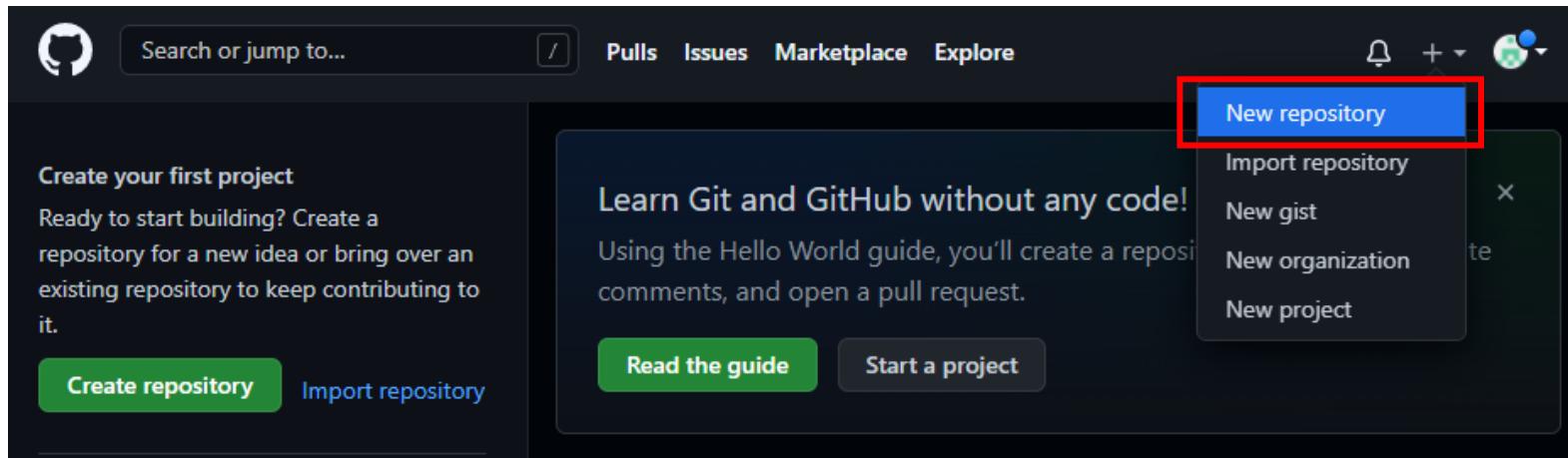
03

지역저장소를
원격저장소에
연결하기

원격저장소 생성

❖ 저장소(repository) 생성

- 오른쪽 상단 + 버튼 클릭 → New repository 선택



원격저장소 생성

Create a new repository

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

Owner *  HelloCoding22 *

Repository name * 저장소 이름

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

Description (optional) 저장소 설명(옵션)

 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](#).
.gitignore template: None ▾

Choose a license
A license tells others what they can and can't do with your code. [Learn more](#).
License: None ▾

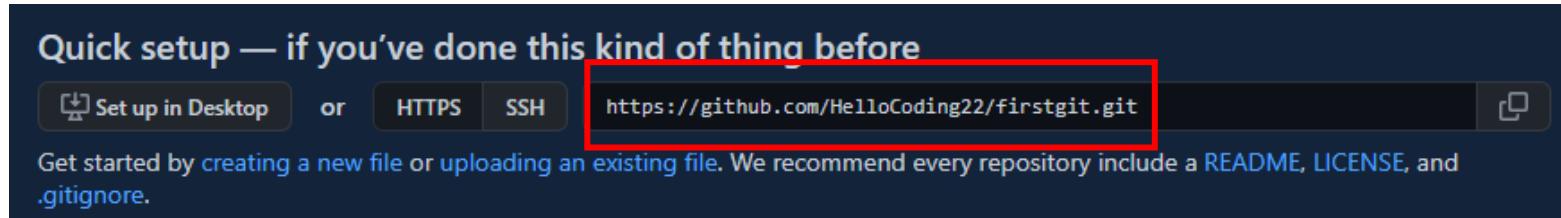
ⓘ You are creating a public repository in your personal account.

Create repository

원격저장소 생성

❖ 원격저장소 주소

`https://github.com/유저이름/저장소이름.git`



- 원격저장소 주소만 있으면 어디서든 지역 저장소를 백업하거나 다른 사람과 협업이 가능함

지역저장소와 원격저장소 연결

- ❖ 지역저장소(hello-git)를 원격저장소에 연결
 - 새로운 파일(conn.txt) 생성 후, 'A' 입력

```
$ vim conn.txt
```



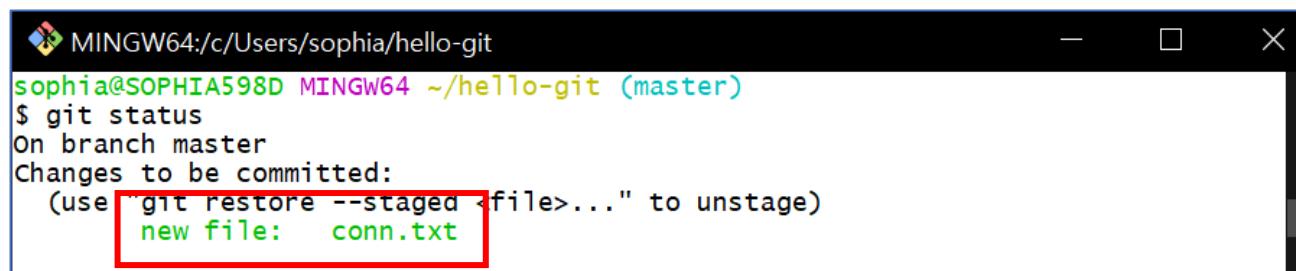
지역저장소와 원격저장소 연결

❖ 스테이지에 파일 추가

```
$ git add conn.txt
```

❖ Git 상태 확인

```
$ git status
```

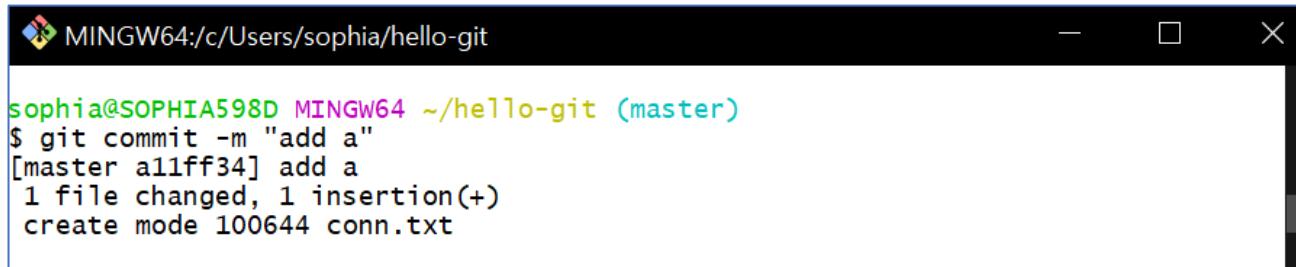


```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   conn.txt
```

지역저장소와 원격저장소 연결

❖ 커밋 실행

```
$ git commit -m "add a"
```



A screenshot of a terminal window titled 'MINGW64:/c/Users/sophia/hello-git'. The window shows the command \$ git commit -m "add a" being run, followed by the output: [master a11ff34] add a 1 file changed, 1 insertion(+) create mode 100644 conn.txt.

```
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git commit -m "add a"
[master a11ff34] add a
 1 file changed, 1 insertion(+)
 create mode 100644 conn.txt
```

지역저장소와 원격저장소 연결

❖ 브랜치 변경

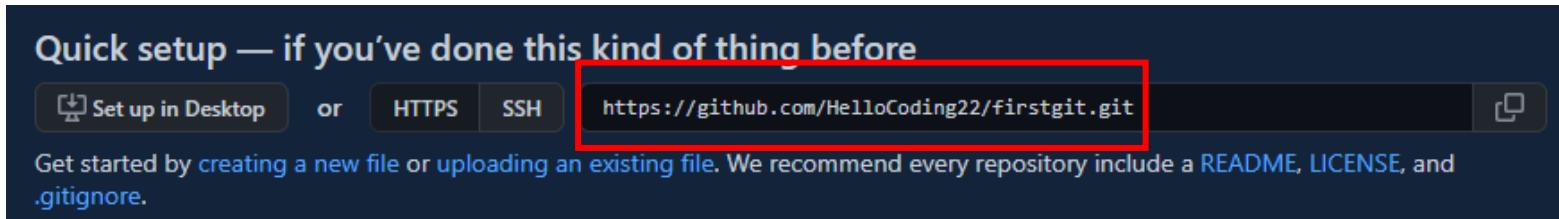
- 기존 master 브랜치에서 main 브랜치로 변경

```
$ git branch -M main
```

지역저장소와 원격저장소 연결

❖ 원격저장소에 연결

- GitHub 저장소 주소 복사



- 터미널에 명령어 입력

```
$ git remote add origin 복사한_주소
```

- remote : 원격저장소
- origin : GitHub 저장소 주소를 등록할 식별자

지역저장소와 원격저장소 연결

- ❖ 원격저장소에 제대로 연결되었는지 확인

```
$ git remote -v
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git remote -v
origin  https://github.com/HelloCoding22/firstgit.git (fetch)
origin  https://github.com/HelloCoding22/firstgit.git (push)
```

원격저장소에 파일 올리기

❖ 원격저장소에 파일 올리기(push)

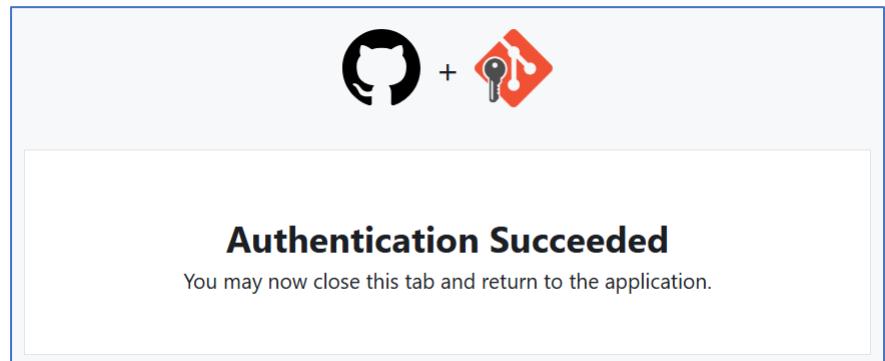
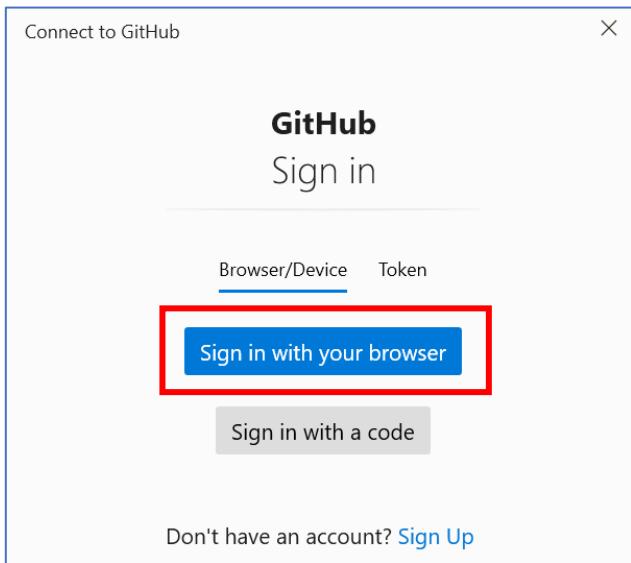
```
$ git push -u origin main
```

- -u
 - 지역저장소의 브랜치를 원격저장소에 연결
 - 최초 한 번만 실행하면 됨
- origin
 - 원격저장소 주소
- main
 - 원격저장소의 브랜치 이름

원격저장소에 파일 올리기

❖ 사용자 인증 과정

- windows



원격저장소에 파일 올리기

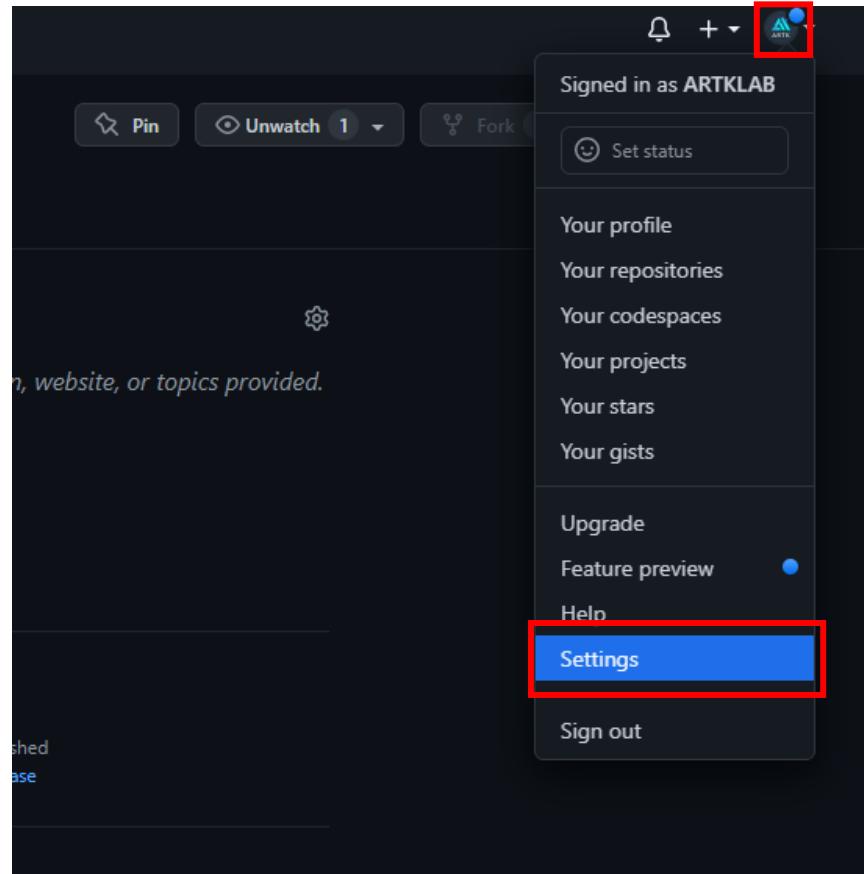
❖ 사용자 인증 과정

- mac
 - 토큰 방식의 인증을 사용
 - Github 사이트에서 토큰 발급 과정 필요

[macOS] 토큰 인증 로그인

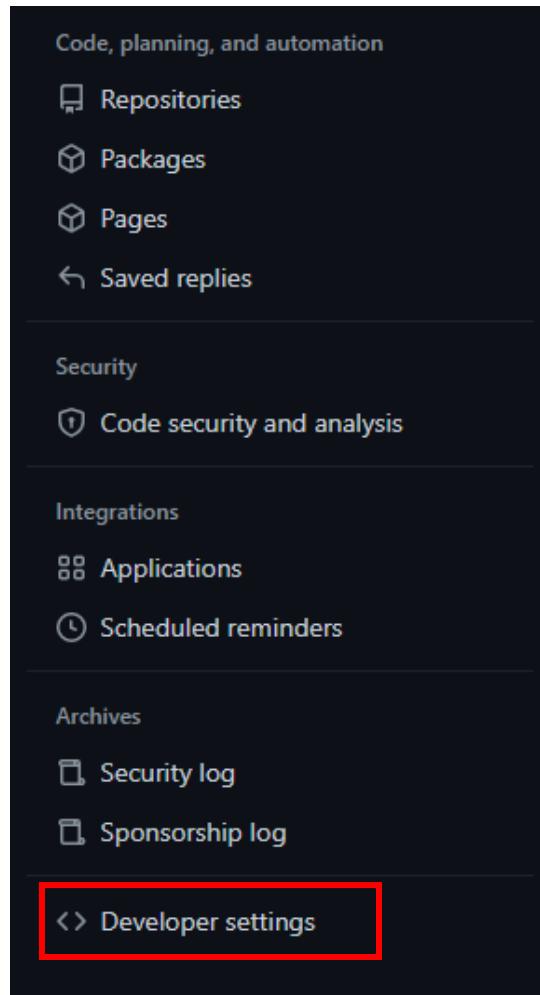
❖ Personal Access Token 생성

- GitHub 로그인 → [Settings] 선택



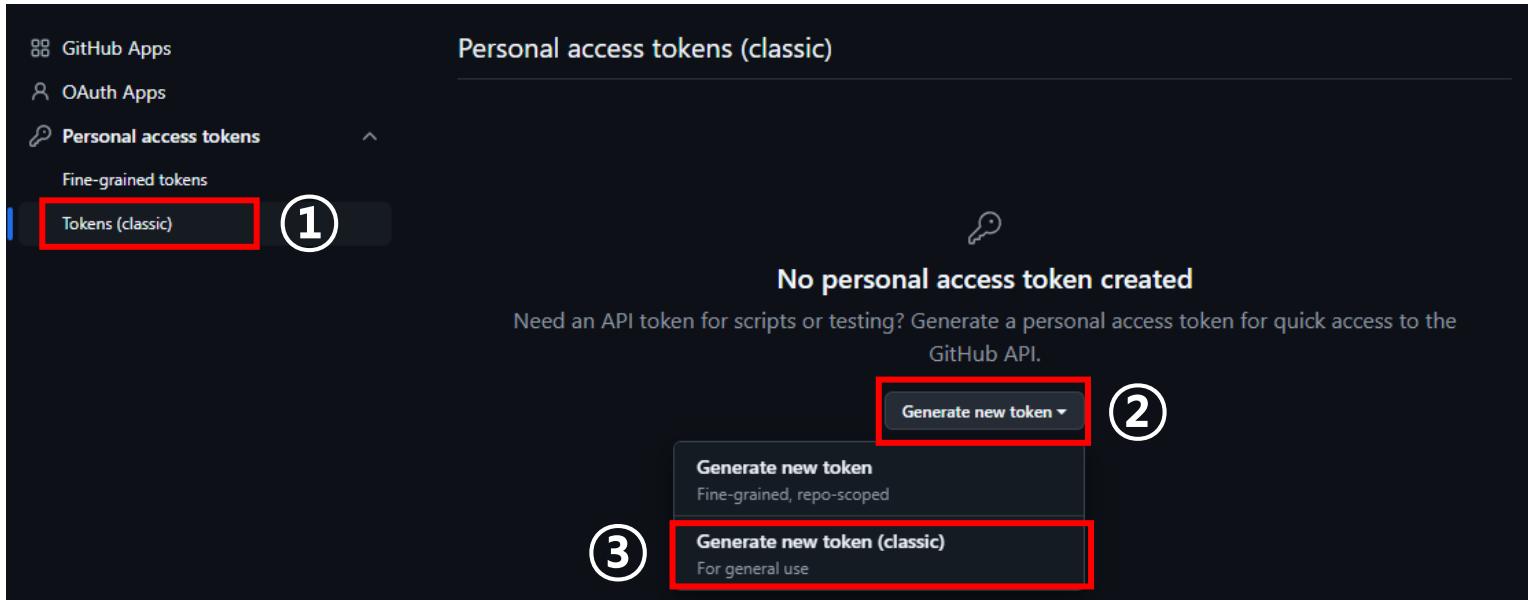
[macOS] 토큰 인증 로그인

❖ 왼쪽 메뉴 하단 → [Developer settings] 선택



[macOS] 토큰 인증 로그인

❖ [Fine-grained Token] 선택



[macOS] 토큰 인증 로그인

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note
|
what's this token for?

Expiration
30 days (Jun 14, 2025)
The token will expire on the selected date

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
 <input type="checkbox"/> workflow	Update GitHub Action workflows

Generate token **Cancel**

토큰 이름 입력
토큰 유효기간
토큰 권한 선택

[macOS] 토큰 인증 로그인

❖ 토큰 복사

The screenshot shows the GitHub 'Personal access tokens' page. At the top, there are buttons for 'Generate new token' and 'Revoke all'. Below this, a message says 'Tokens you have generated that can be used to access the GitHub API.' A prominent callout box contains the instruction 'Make sure to copy your personal access token now. You won't be able to see it again!'. Below the message, a list of tokens is shown. One token, starting with 'ghp...' and ending with 'obf', is highlighted with a red box and has a yellow background. To its right is a 'Delete' button.

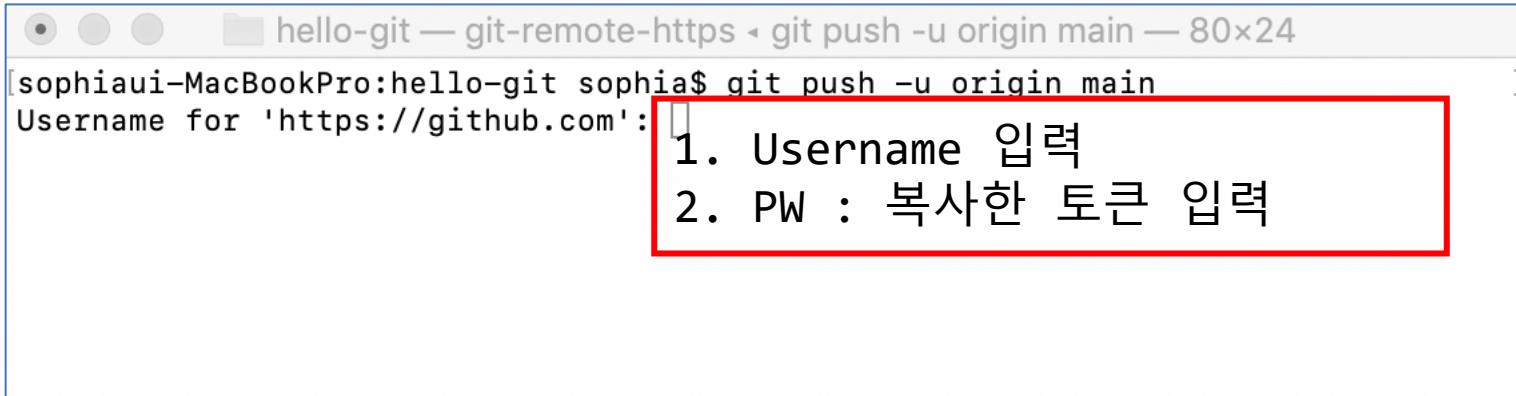
Token	Action
ghp...obf	Delete

[macOS] 토큰 인증 로그인

❖ 원격저장소에 파일 올리기(push)

```
$ git push -u origin main
```

❖ 토큰 인증



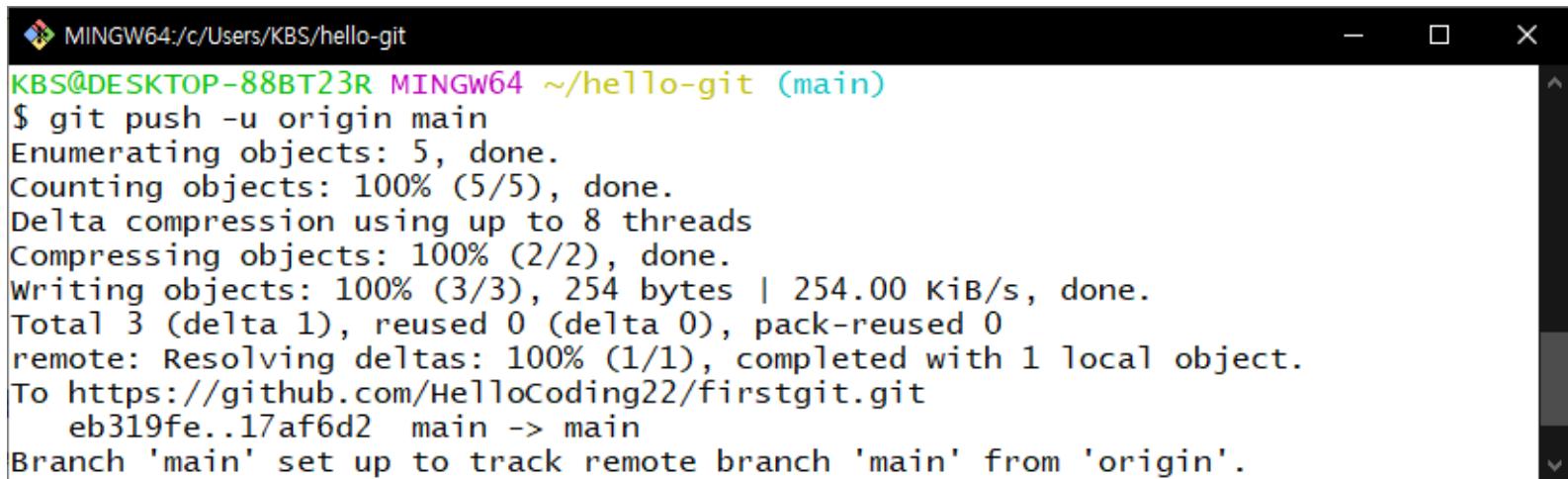
A screenshot of a terminal window titled "hello-git — git-remote-https < git push -u origin main — 80x24". The window shows the command "git push -u origin main" being run. A password prompt follows: "Username for 'https://github.com':". A red rectangular box highlights this prompt and the subsequent text "1. Username 입력" and "2. PW : 복사한 토큰 입력", which are instructions for entering the GitHub username and token respectively.

```
sophiaui-MacBookPro:hello-git sophia$ git push -u origin main
Username for 'https://github.com':
```

1. Username 입력
2. PW : 복사한 토큰 입력

원격저장소에 파일 올리기

❖ 원격저장소에 파일 올리기 성공



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 254 bytes | 254.00 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.
To https://github.com>HelloCoding22/firstgit.git
 eb319fe..17af6d2 main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

원격저장소에 파일 올리기

❖ GitHub에서 확인하기

The image shows two screenshots of a GitHub repository named 'HelloCoding22'.

Screenshot 1: Commit History

This screenshot displays a list of commits:

File	Message	Time
.gitignore	message3	2 days ago
bye.txt	Revert "test commit 2"	yesterday
coding.txt	message3	2 days ago
conn.txt	add a	25 minutes ago
hello.txt	message3	2 days ago

A red box highlights the commit for 'conn.txt'. A red arrow from the bottom screenshot points to this highlighted commit.

Screenshot 2: File Preview

This screenshot shows the details for the 'conn.txt' file in a commit:

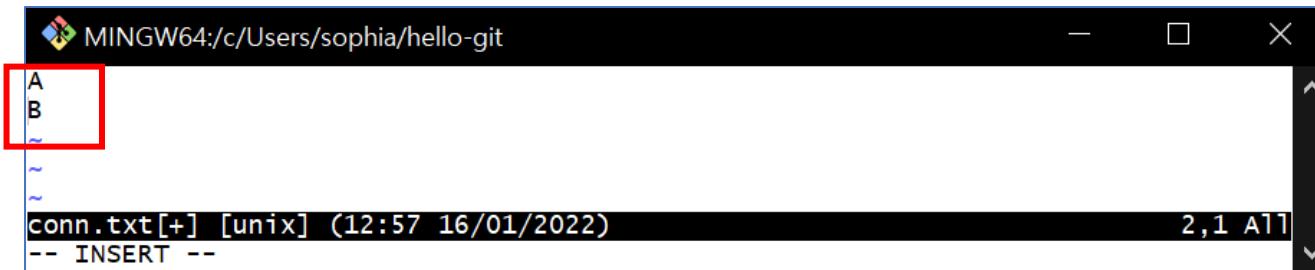
- Branch: main
- Commit: firstgit
- File Path: conn.txt
- Contributor: HelloCoding22
- Contributors: 1 contributor
- Lines: 1 lines (1 sloc)
- Size: 2 Bytes
- Content: 1 A

A red box highlights the file path 'conn.txt' in the URL bar. Another red box highlights the content '1 A' in the preview area.

원격저장소에 파일 올리기

❖ 파일 수정 후 원격저장소에 올리기

```
$ vim conn.txt
```



A screenshot of a terminal window titled "MINGW64:/c/Users/sophia/hello-git". The window displays the file "conn.txt" with the following content:

```
A
B
~
~
conn.txt[+] [unix] (12:57 16/01/2022) 2,1 All
-- INSERT --
```

The first two lines, "A" and "B", are highlighted with a red rectangular box.

원격저장소에 파일 올리기

❖ 스테이지에 파일 추가

```
$ git add conn.txt
```

❖ 커밋 실행

```
$ git commit -m "add b"
```



The screenshot shows a terminal window with the following content:

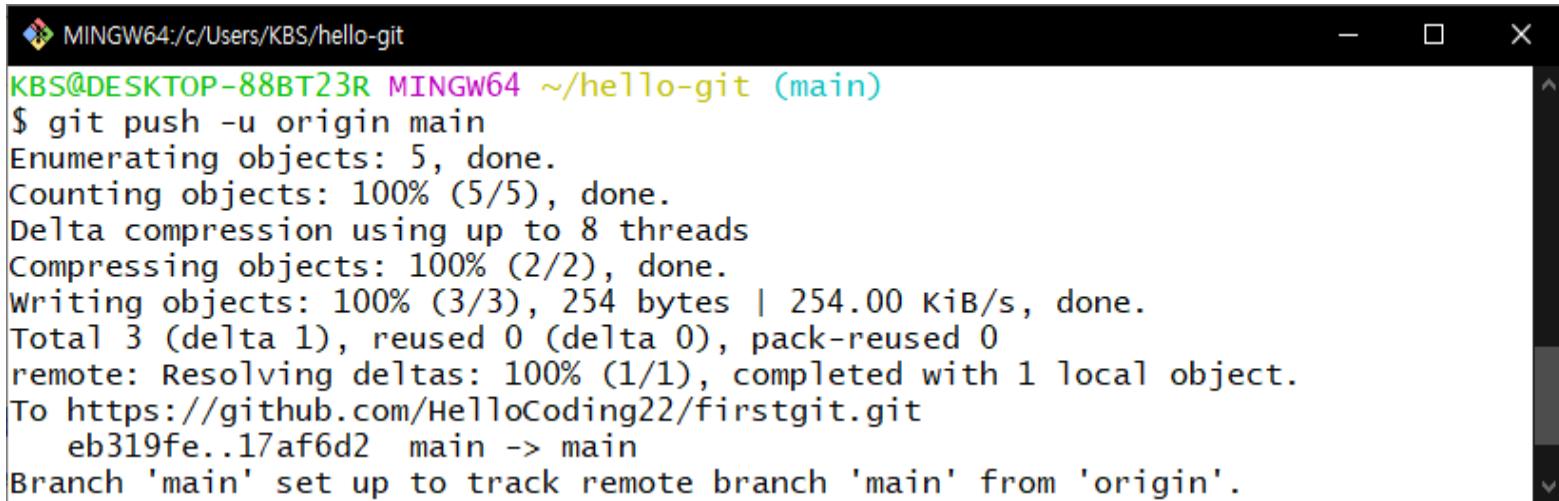
```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git commit -m "add b"
[main 17af6d2] add b
 1 file changed, 1 insertion(+)
```

원격저장소에 파일 올리기

❖ 원격저장소에 파일 올리기(push)

- 한 번 이상 원격저장소에 업로드한 경우, 다음 명령어로 push 가능

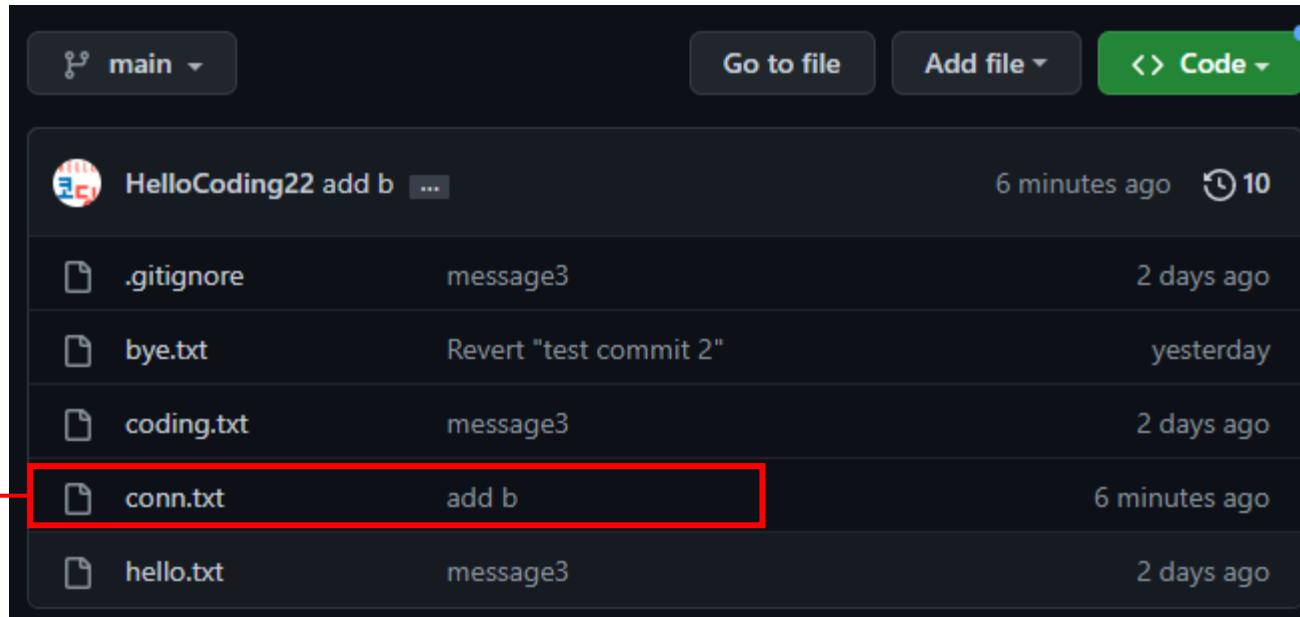
```
$ git push
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 254 bytes | 254.00 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.
To https://github.com/HelloCoding22/firstgit.git
  eb319fe..17af6d2  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

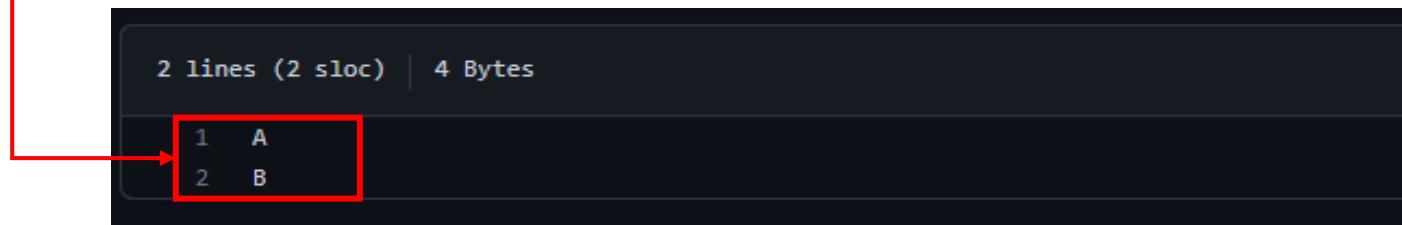
원격저장소에 파일 올리기

❖ GitHub에서 확인하기



The screenshot shows a GitHub commit history for a repository named 'HelloCoding22'. The commits are listed in reverse chronological order:

- A commit by 'HelloCoding22' titled 'add b' was made 6 minutes ago. It includes a link to the file 'conn.txt'.
- A commit titled 'Revert "test commit 2"' was made yesterday.
- A commit by 'HelloCoding22' titled 'message3' was made 2 days ago.
- A commit by 'HelloCoding22' titled 'add b' was made 6 minutes ago. This commit is highlighted with a red rectangle.
- A commit by 'HelloCoding22' titled 'message3' was made 2 days ago.

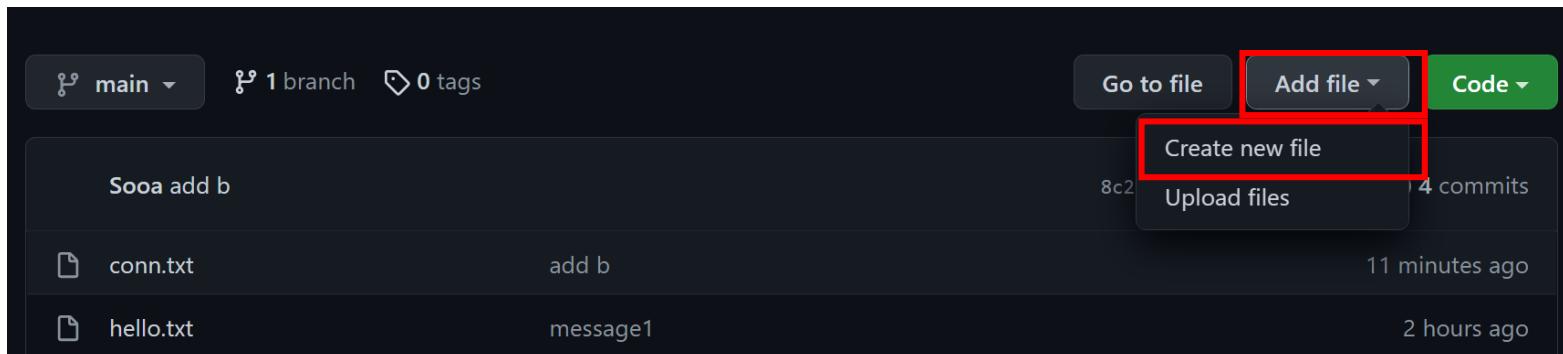


The screenshot shows the GitHub code editor displaying the contents of the 'conn.txt' file. The file contains two lines of text: 'A' and 'B'. The entire code block is highlighted with a red rectangle, and a red arrow points from the bottom-left corner of this rectangle to the first line of the file's content.

```
2 lines (2 sloc) | 4 Bytes
1 A
2 B
```

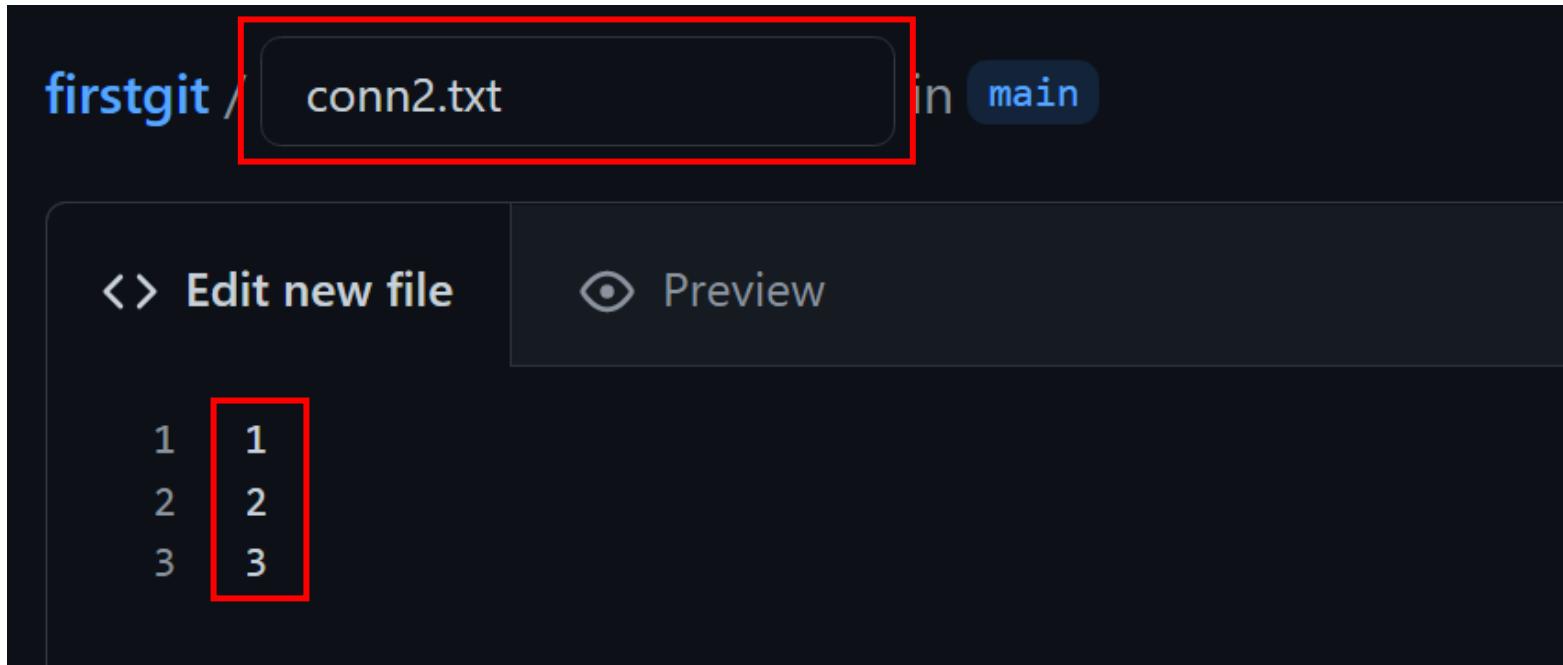
GitHub 사이트에서 직접 커밋하기

- ❖ 지역저장소가 있는 컴퓨터를 사용하지 못하는 경우
 - GitHub 사이트에서 직접 커밋이 가능



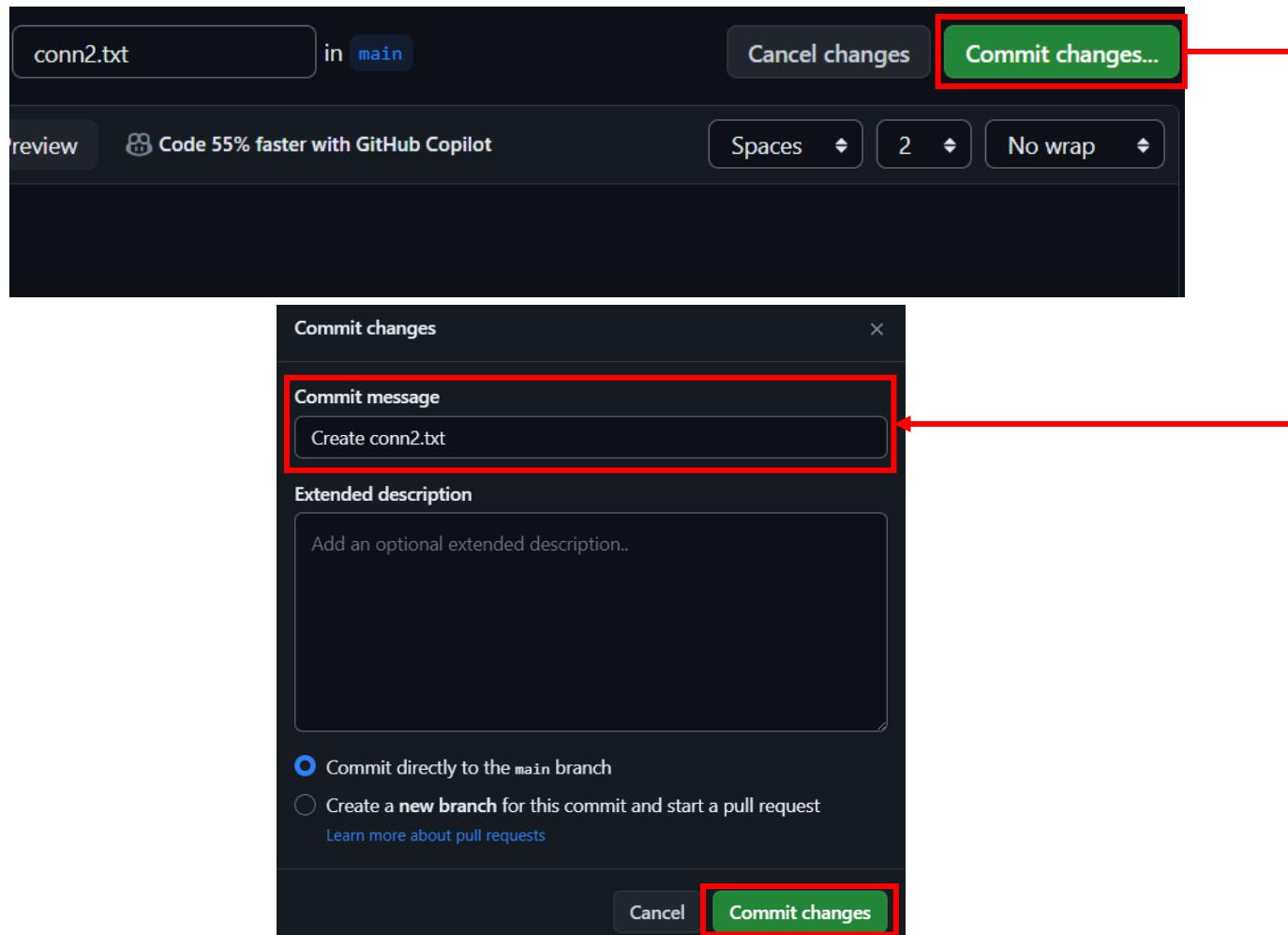
GitHub 사이트에서 직접 커밋하기

❖ 파일명과 내용 입력



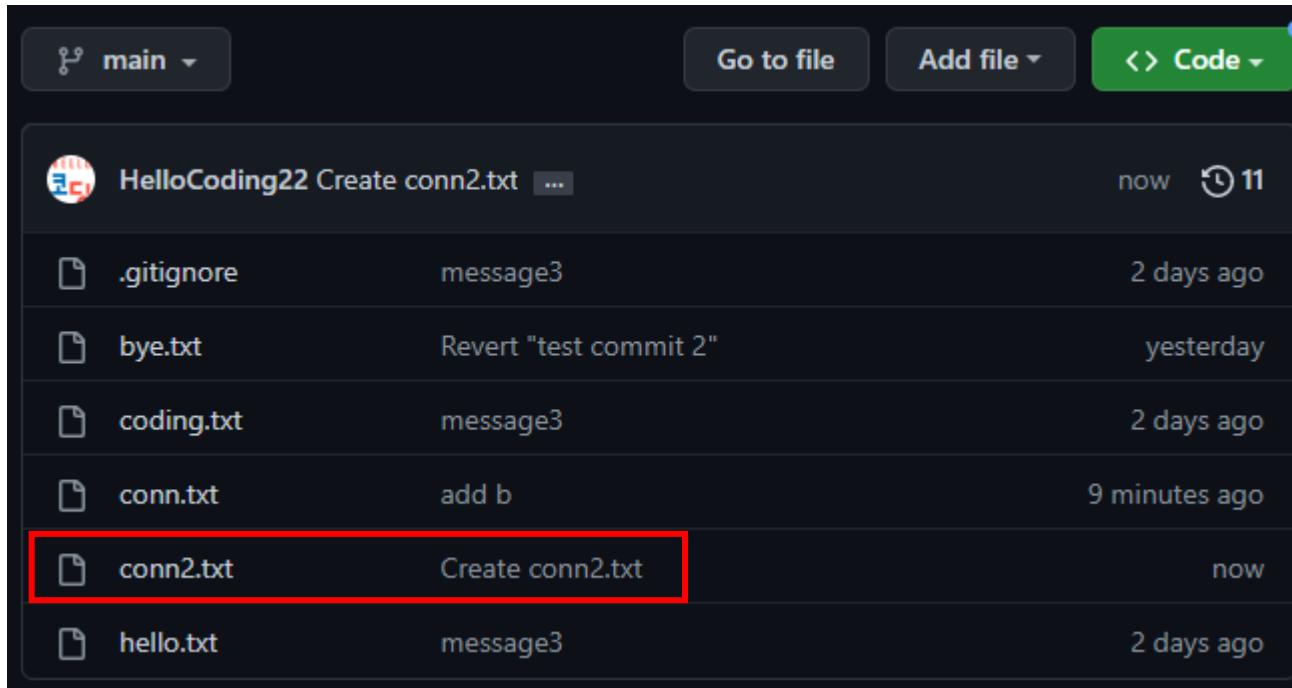
GitHub 사이트에서 직접 커밋하기

❖ 커밋 메시지 입력 후 커밋



GitHub 사이트에서 직접 커밋하기

❖ 새 파일(conn2.txt) 추가



The screenshot shows a GitHub repository's commit history. At the top, there are buttons for 'main' (with a dropdown arrow), 'Go to file', 'Add file', and 'Code'. Below the commits, there is a search bar and a filter icon.

Commit	Message	Time
HelloCoding22 Create conn2.txt	...	now 11
.gitignore	message3	2 days ago
bye.txt	Revert "test commit 2"	yesterday
coding.txt	message3	2 days ago
conn.txt	add b	9 minutes ago
conn2.txt	Create conn2.txt	now
hello.txt	message3	2 days ago

The commit for 'conn2.txt' is highlighted with a red rectangle.

원격저장소에서 파일 다운로드

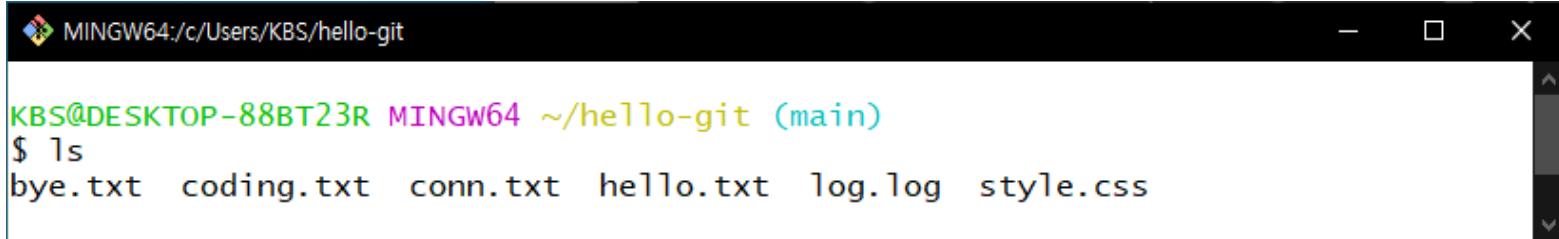
❖ 원격저장소에서 파일 다운로드(pull)

- 원격저장소와 지역저장소의 상태가 다른 경우
 - 원격저장소에 있는 파일을 다른 사용자가 수정
 - GitHub 사이트에서 직접 커밋
- 원격저장소와 지역저장소의 상태를 같게 만들어야 함
 - 원격저장소의 파일을 지역저장소로 가져옴
 - 풀(pull)

원격저장소에서 파일 내려받기

❖ 지역저장소 파일 확인

```
$ ls
```



A screenshot of a terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The window shows the command 'ls' being run and its output, which includes files named 'bye.txt', 'coding.txt', 'conn.txt', 'hello.txt', 'log.log', and 'style.css'. The terminal has a dark theme with light-colored text.

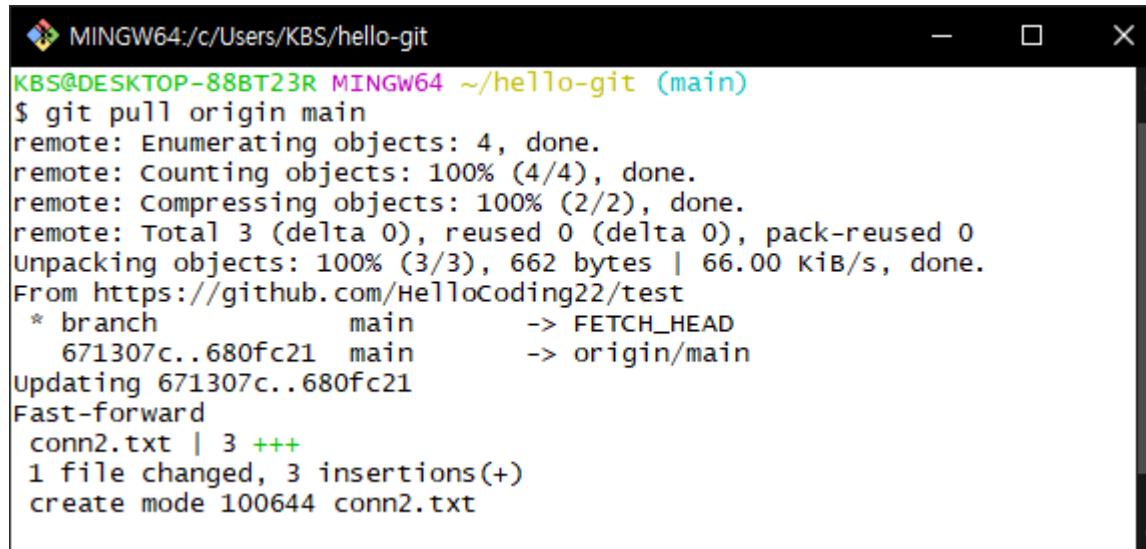
```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ ls
bye.txt  coding.txt  conn.txt  hello.txt  log.log  style.css
```

- conn2.txt 파일이 존재하지 않음

원격저장소에서 파일 내려받기

❖ 원격저장소의 내용을 가져오기

```
$ git pull origin main
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The command \$ git pull origin main is entered, followed by its execution output:

```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 662 bytes | 66.00 KiB/s, done.
From https://github.com/HelloCoding22/test
 * branch            main      -> FETCH_HEAD
   671307c..680fc21  main      -> origin/main
Updating 671307c..680fc21
Fast-forward
 conn2.txt | 3 ++
 1 file changed, 3 insertions(+)
 create mode 100644 conn2.txt
```

원격저장소에서 파일 내려받기

❖ 지역저장소 파일 확인

```
$ ls
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (main)
$ ls
bye.txt  coding.txt  conn.txt  conn2.txt  hello.txt  log.log  style.css
```

- conn2.txt 파일 생성됨

(에러 대응)지역저장소와 원격저장소 연결

❖ 지역저장소와 원격저장소의 브랜치가 다른 경우

- 파일 업로드(push) 실행 시, 에러 발생
- 지역저장소 브랜치 : master
- 원격저장소 브랜치 : main

지역저장소 브랜치



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git push -u origin main
To https://github.com>HelloCoding22/firstgit.git
! [rejected]          main -> main (fetch first)
error: failed to push some refs to 'https://github.com>HelloCoding22/firstgit.git'
hint: Updates were rejected because the remote contains work t
hat you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote change
s, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help'
```

정리하기

git remote add origin 원격저장소_주소	원격저장소에 연결하기
git remote -v	원격저장소에 연결되었는지 확인하기
git push -u origin main	지역저장소의 커밋을 원격저장소에 업로드하기
git push	두번째 커밋을 원격 저장소에 올리기
git pull	원격저장소의 커밋을 지역저장소로 다운로드하기

Git & GitHub

- ◆ GitHub로 협업하기

정수아

Contents

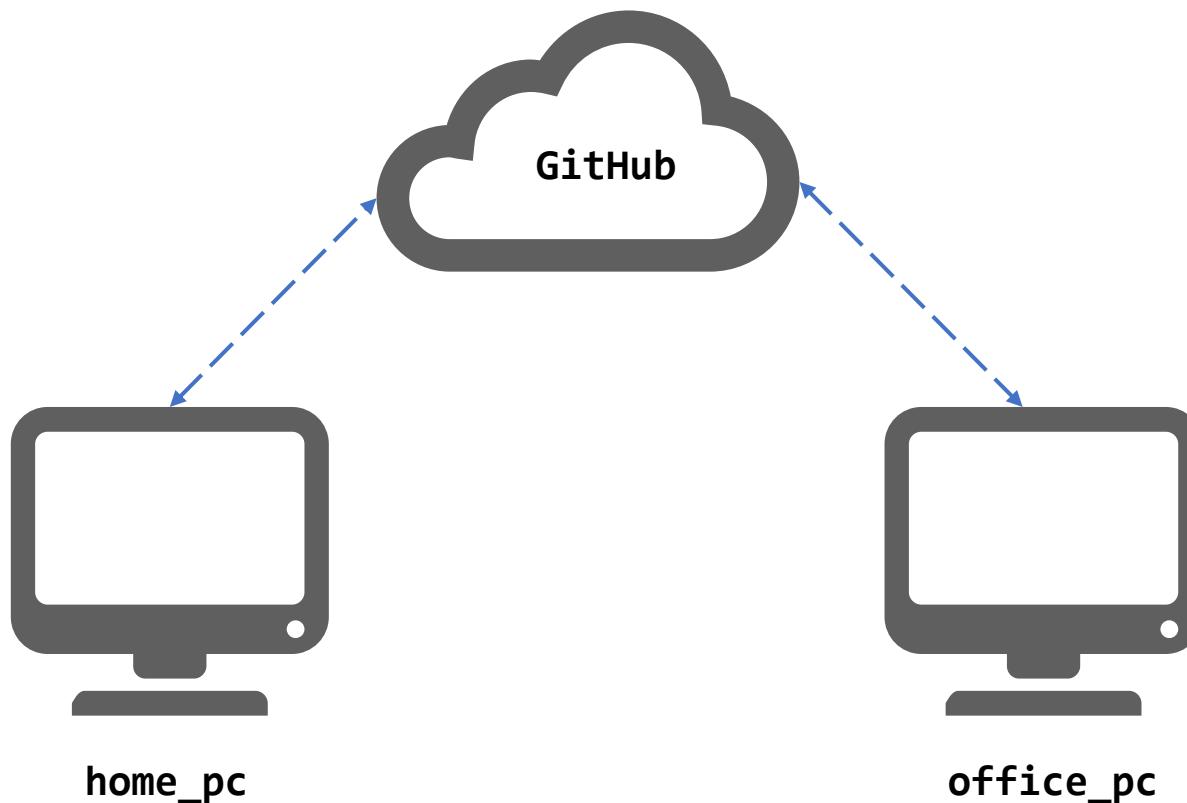
01 원격저장소 공유하기

01

원격저장소 공유하기

원격저장소 공유하기

- ❖ 여러 대의 컴퓨터에서 하나의 GitHub 계정을 공유



원격저장소 공유하기

❖ 원격저장소 복제하기

```
$ git clone 원격_저장소_주소 복제할_경로
```

원격저장소 공유하기

- ❖ home_pc 디렉터리에 원격저장소 복제하기

```
$ git clone 원격_저장소_주소 home_pc
```

- ❖ office_pc 디렉터리에 원격저장소 복제하기

```
$ git clone 원격_저장소_주소 office_pc
```

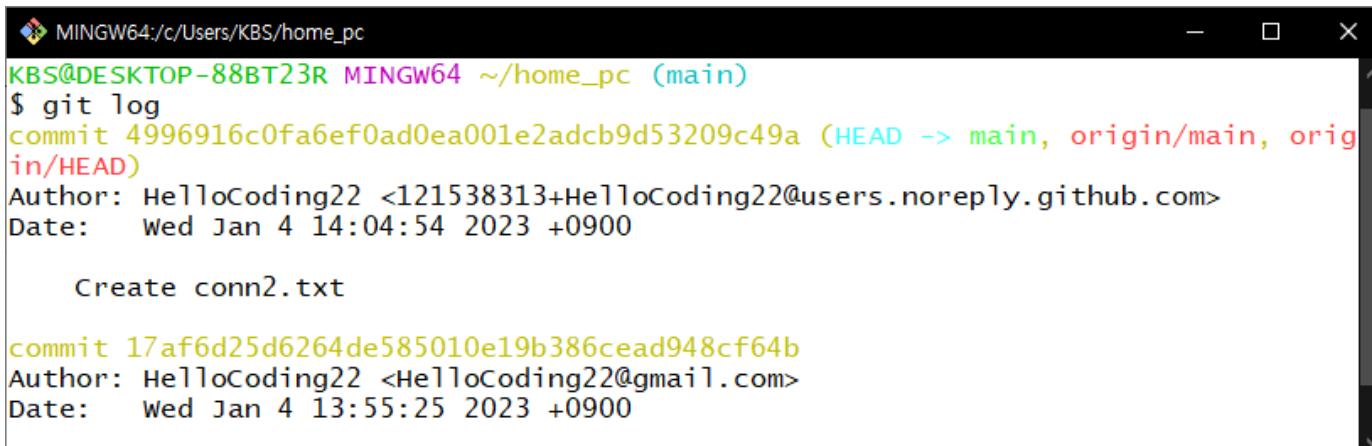
원격저장소 공유하기

- ❖ 원격저장소가 복제 되었는지 확인
 - home_pc 디렉터리로 이동

```
$ cd home_pc
```

- ❖ 로그 확인하기

```
$ git log
```



The screenshot shows a terminal window with the following text output:

```
MINGW64:/c/Users/KBS/home_pc
KBS@DESKTOP-88BT23R MINGW64 ~/home_pc (main)
$ git log
commit 4996916c0fa6ef0ad0ea001e2adcb9d53209c49a (HEAD -> main, origin/main, origin/HEAD)
Author: HelloCoding22 <121538313+HelloCoding22@users.noreply.github.com>
Date:   Wed Jan 4 14:04:54 2023 +0900

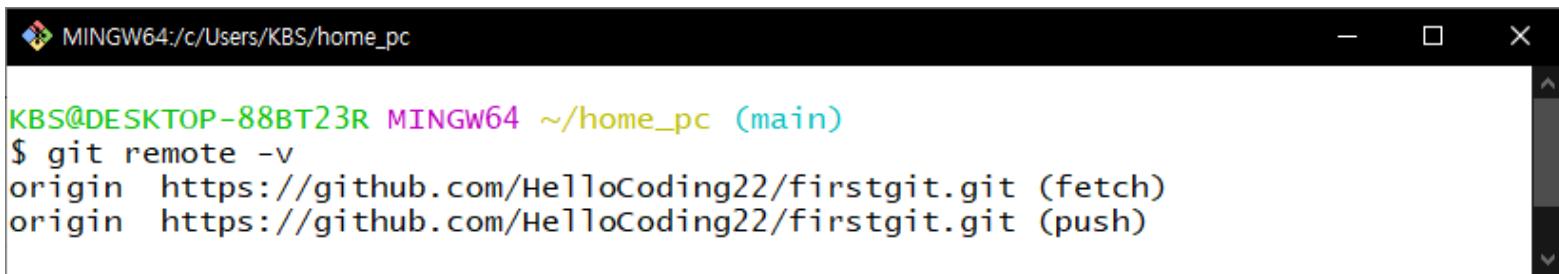
    Create conn2.txt

commit 17af6d25d6264de585010e19b386cead948cf64b
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Wed Jan 4 13:55:25 2023 +0900
```

원격저장소 공유하기

- ❖ 지역저장소가 원격저장소와 연결되었는지 확인

```
$ git remote -v
```

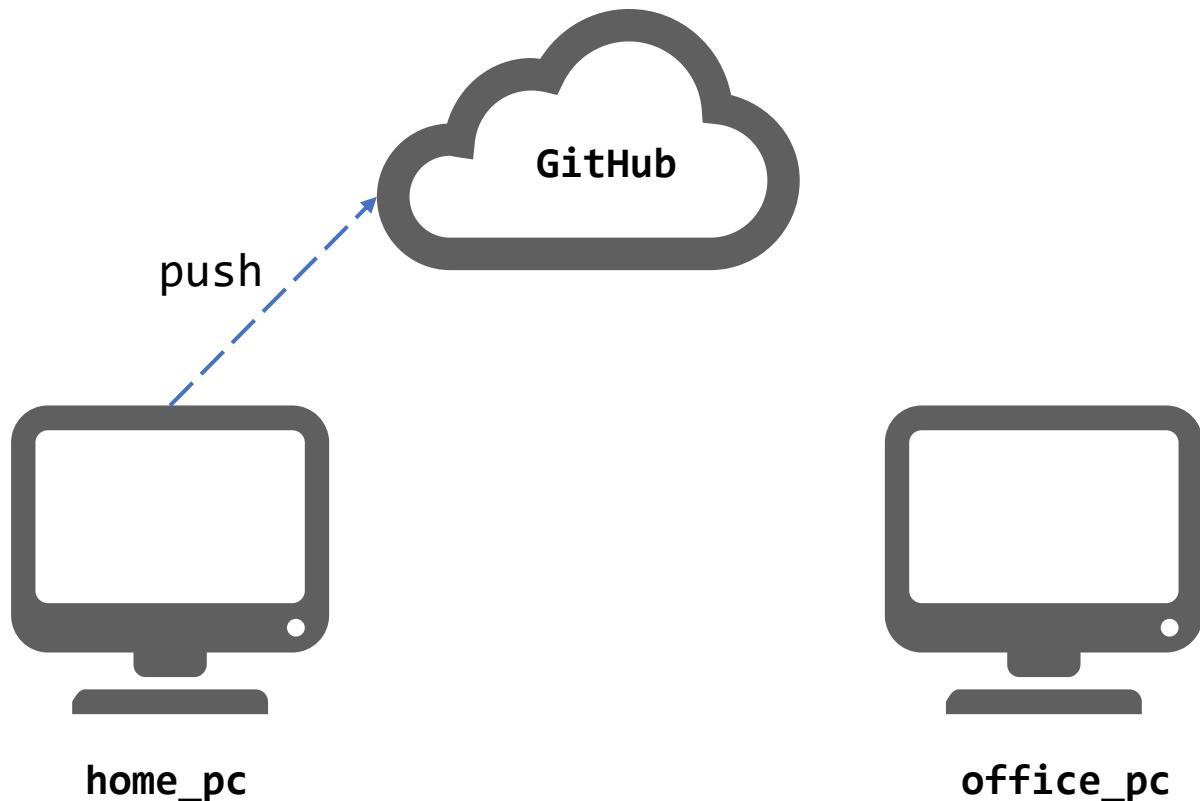


The screenshot shows a terminal window with a black background and white text. The title bar reads "MINGW64:/c/Users/KBS/home_pc". The command \$ git remote -v is entered, and the output shows two remote origins: "origin https://github.com/HelloCoding22/firstgit.git (fetch)" and "origin https://github.com/HelloCoding22/firstgit.git (push)".

```
KBS@DESKTOP-88BT23R MINGW64 ~/home_pc (main)
$ git remote -v
origin https://github.com/HelloCoding22/firstgit.git (fetch)
origin https://github.com/HelloCoding22/firstgit.git (push)
```

원격 저장소 공유하기

- ❖ home_pc에서 작업하기



원격 저장소 공유하기

- ❖ home_pc에서 작업하기
 - 새 파일 만들고, 내용 입력

```
$ vim firstHome.txt
```

```
1
2
3
~
~
```

firstHome.txt[+] [unix] (09:33 09/01/2023) 3 ,1 All
-- INSERT --

원격 저장소 공유하기

❖ home_pc에서 작업하기

- 스테이지에 올리기

```
$ git add firstHome.txt
```

- 커밋하기

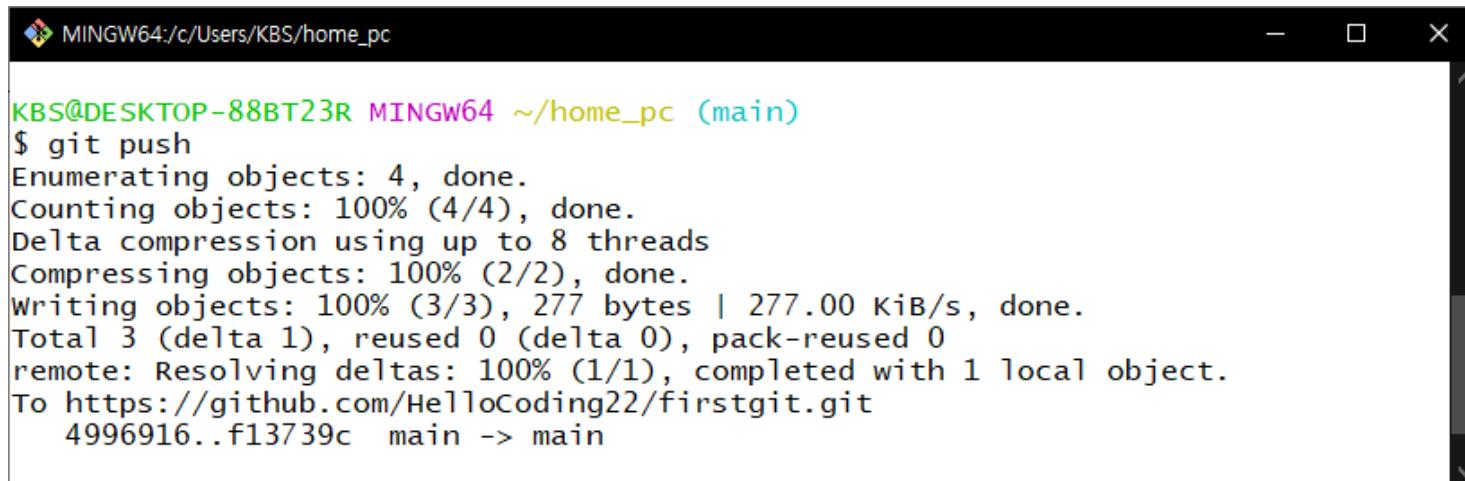
```
$ git commit -m "add firstHome"
```

원격 저장소 공유하기

❖ home_pc에서 작업하기

- 지역 저장소의 커밋을 원격 저장소에 올리기

```
$ git push
```

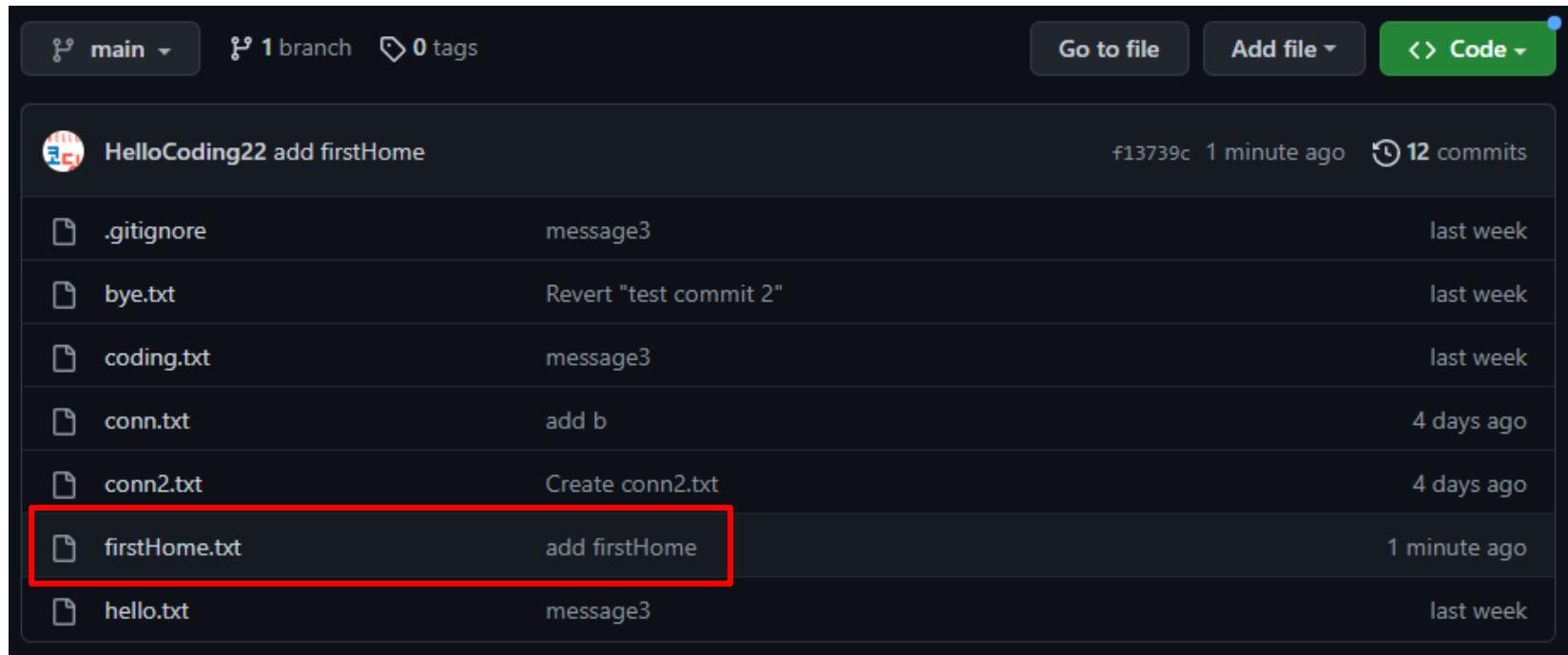


The screenshot shows a terminal window with a black background and white text. The window title is 'MINGW64:/c/Users/KBS/home_pc'. The command \$ git push is entered, followed by its execution output:

```
KBS@DESKTOP-88BT23R MINGW64 ~/home_pc (main)
$ git push
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), 277 bytes | 277.00 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.
To https://github.com/HelloCoding22/firstgit.git
 4996916..f13739c main -> main
```

원격 저장소 공유하기

❖ GitHub에서 확인하기

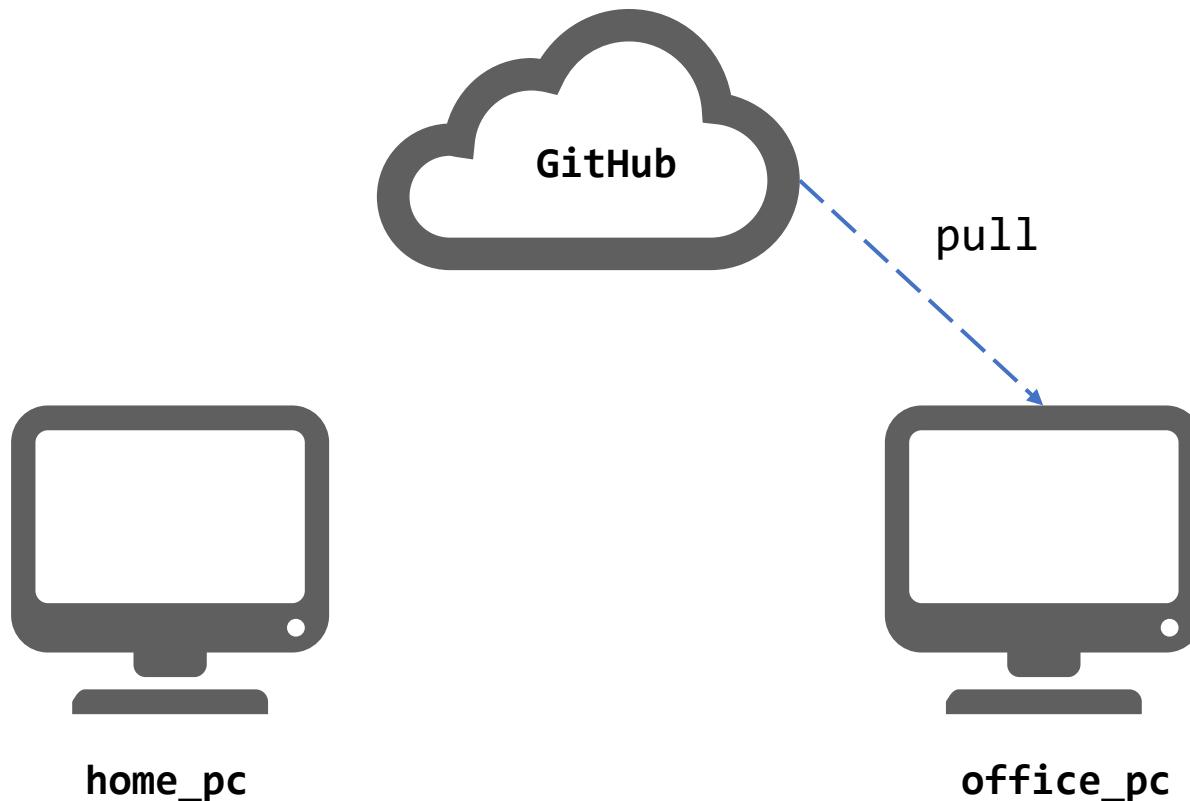


The screenshot shows a GitHub repository interface. At the top, it displays 'main' branch, 1 branch, 0 tags, and various navigation buttons like 'Go to file', 'Add file', and 'Code'. Below this, a list of commits is shown:

Commit	Message	Date
.gitignore	message3	last week
bye.txt	Revert "test commit 2"	last week
coding.txt	message3	last week
conn.txt	add b	4 days ago
conn2.txt	Create conn2.txt	4 days ago
firstHome.txt	add firstHome	1 minute ago
hello.txt	message3	last week

원격 저장소 공유하기

- ❖ office_pc에서 작업하기



원격 저장소 공유하기

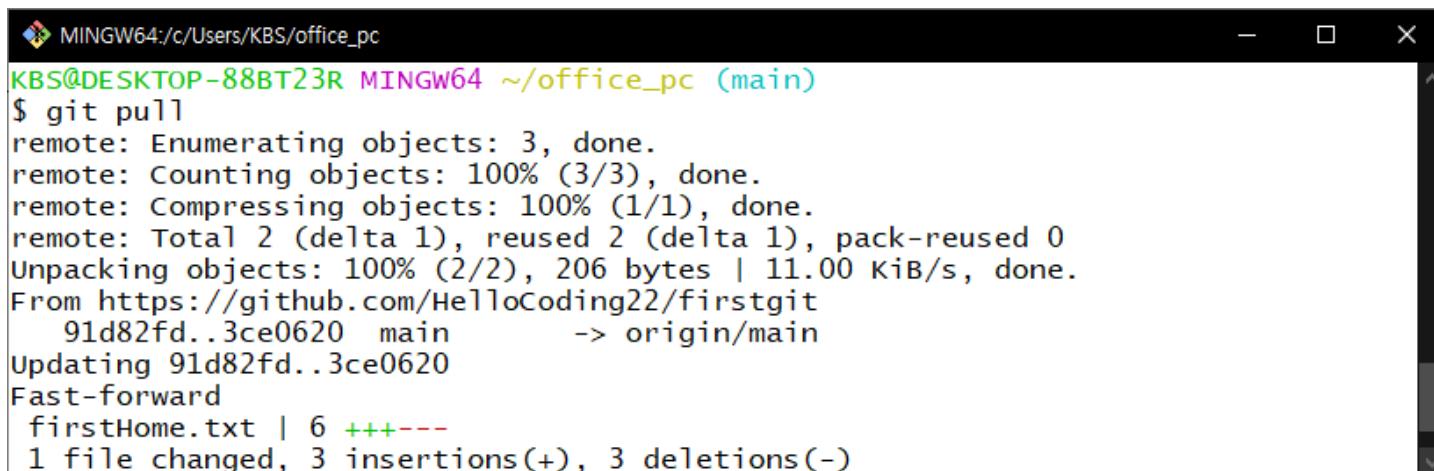
❖ office_pc에서 작업하기

- office_pc 디렉터리로 이동

```
$ cd office_pc
```

❖ 원격 저장소에서 파일 내려받기

```
$ git pull
```

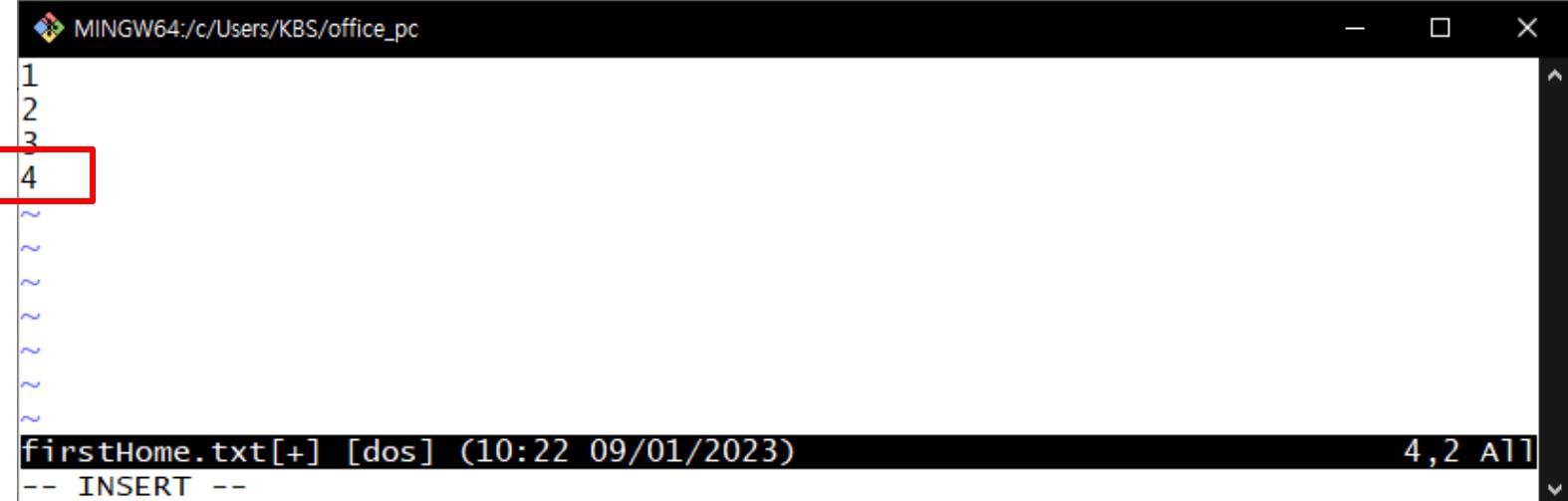


```
MINGW64:/c/Users/KBS/office_pc
KBS@DESKTOP-88BT23R MINGW64 ~/office_pc (main)
$ git pull
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 2 (delta 1), reused 2 (delta 1), pack-reused 0
Unpacking objects: 100% (2/2), 206 bytes | 11.00 KiB/s, done.
From https://github.com>HelloCoding22/firstgit
 91d82fd..3ce0620 main      -> origin/main
Updating 91d82fd..3ce0620
Fast-forward
  firstHome.txt | 6 +++---
  1 file changed, 3 insertions(+), 3 deletions(-)
```

원격 저장소 공유하기

- ❖ office_pc에서 작업하기
 - firstHome.txt 파일 수정

```
$ vim firstHome.txt
```



```
MINGW64:/c/Users/KBS/office_pc
1
2
3
4
~
~  
~  
~  
~  
~  
~  
~  
firstHome.txt[+] [dos] (10:22 09/01/2023) 4,2 All
-- INSERT --
```

원격 저장소 공유하기

❖ office_pc에서 작업하기

- 스테이지에 올리기

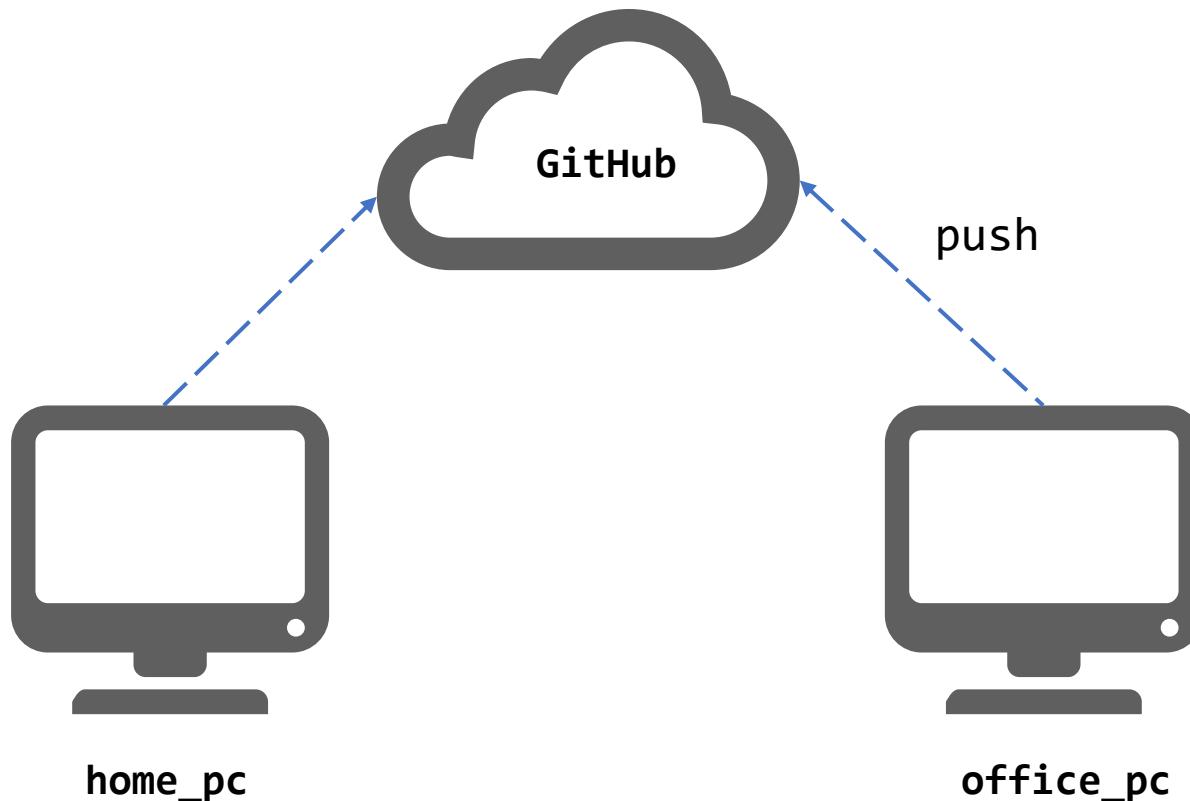
```
$ git add firstHome.txt
```

- 커밋하기

```
$ git commit -m "add 4"
```

원격 저장소 공유하기

- ❖ office_pc에서 작업하기

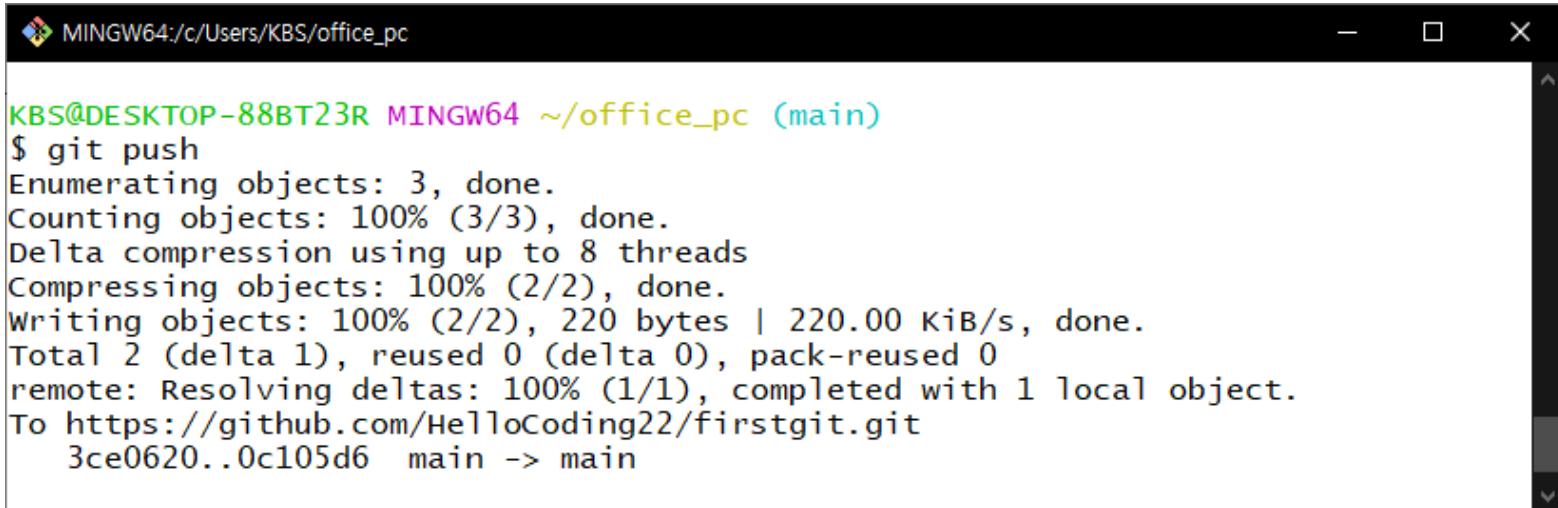


원격 저장소 공유하기

❖ office_pc에서 작업하기

- 지역 저장소의 커밋을 원격 저장소에 올리기

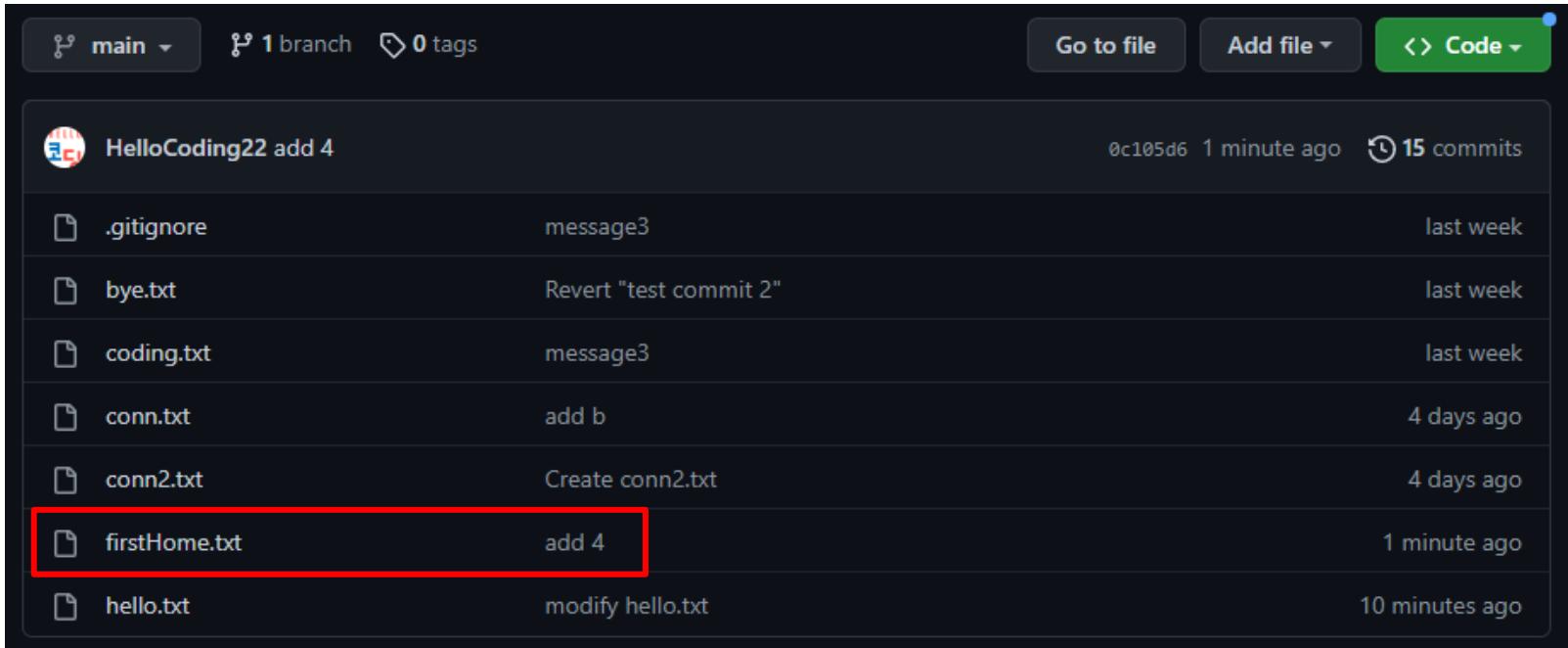
```
$ git push
```



A screenshot of a terminal window titled 'MINGW64:c/Users/KBS/office_pc'. The window shows the command \$ git push followed by its execution output. The output includes: Enumerating objects: 3, done. Counting objects: 100% (3/3), done. Delta compression using up to 8 threads. Compressing objects: 100% (2/2), done. Writing objects: 100% (2/2), 220 bytes | 220.00 KiB/s, done. Total 2 (delta 1), reused 0 (delta 0), pack-reused 0. remote: Resolving deltas: 100% (1/1), completed with 1 local object. To https://github.com>HelloCoding22/firstgit.git 3ce0620..0c105d6 main -> main.

원격 저장소 공유하기

❖ GitHub에서 확인하기



The screenshot shows a GitHub repository interface with a commit history. The top navigation bar includes 'main' (selected), '1 branch', '0 tags', 'Go to file', 'Add file', and a green 'Code' button. The commit list is titled 'HelloCoding22 add 4' and shows the following entries:

File	Message	Time
.gitignore	message3	last week
bye.txt	Revert "test commit 2"	last week
coding.txt	message3	last week
conn.txt	add b	4 days ago
conn2.txt	Create conn2.txt	4 days ago
firstHome.txt	add 4	1 minute ago
hello.txt	modify hello.txt	10 minutes ago

Git & GitHub

- ◆ GitHub에서 소통하기

정수아

Contents

01 GitHub 프로필 작성하기

02 README 파일 작성하기

03 마크다운 문법



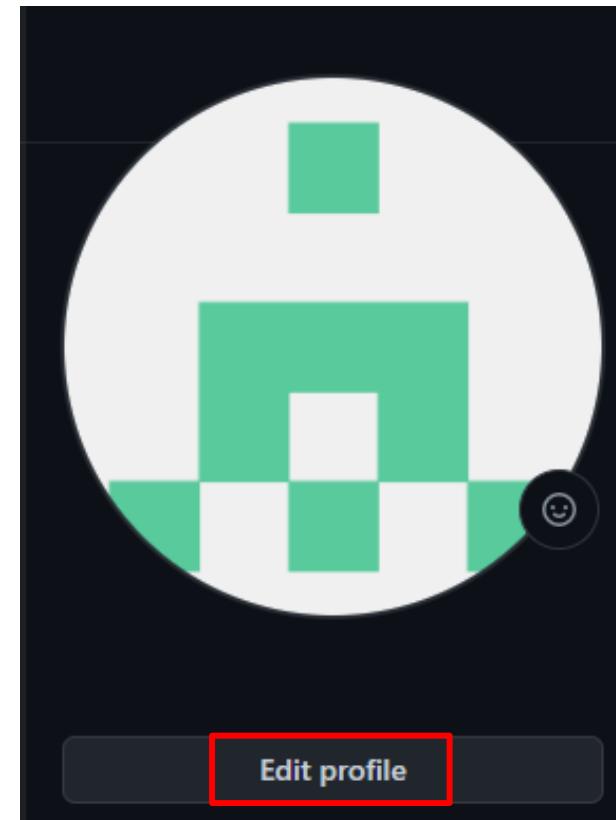
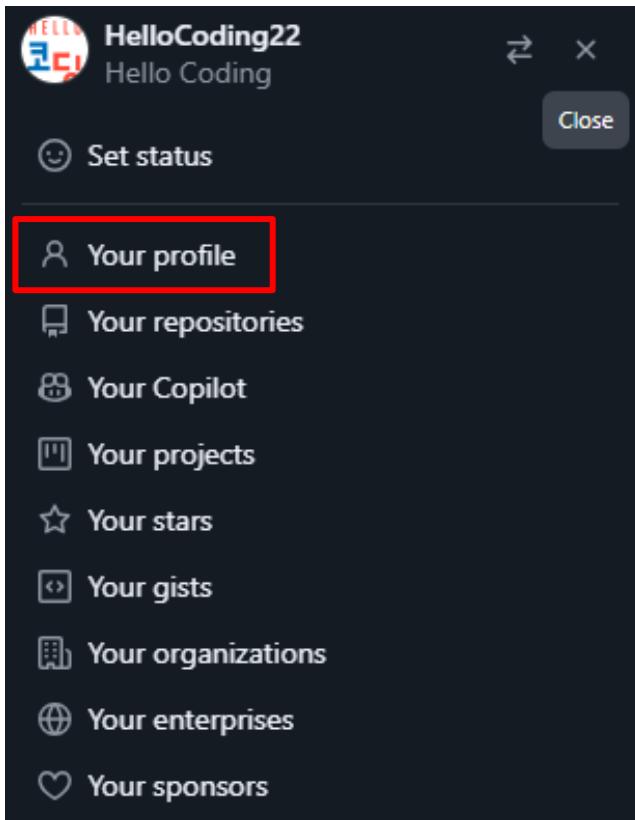
01

GitHub 프로필 작성하기

GitHub 프로필 작성하기

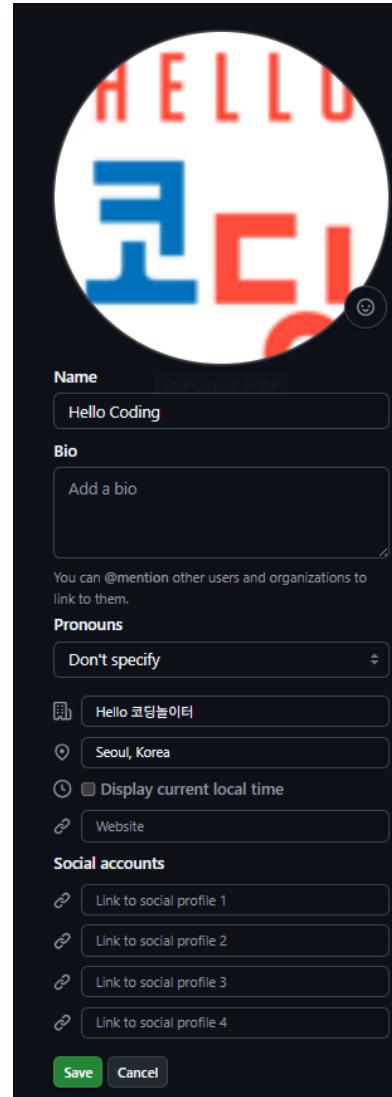
❖ 프로필 관리하기

- GitHub 홈페이지 오른쪽 상단 이미지 클릭 → [Your profile]



GitHub 프로필 작성하기

❖ 프로필 작성 화면



GitHub 프로필 작성하기

❖ 컨트리뷰션(contributions) 그래프

- 사용자가 최근 1년 동안 GitHub에서 얼마나 활발하게 활동했는지를 시각적으로 보여줌
- 사각형 하나 : 하루를 의미
- 초록색 사각형 : 컨트리뷰션이 있는 날
- 초록색이 진할수록 그날 컨트리뷰션이 많았다는 뜻



02

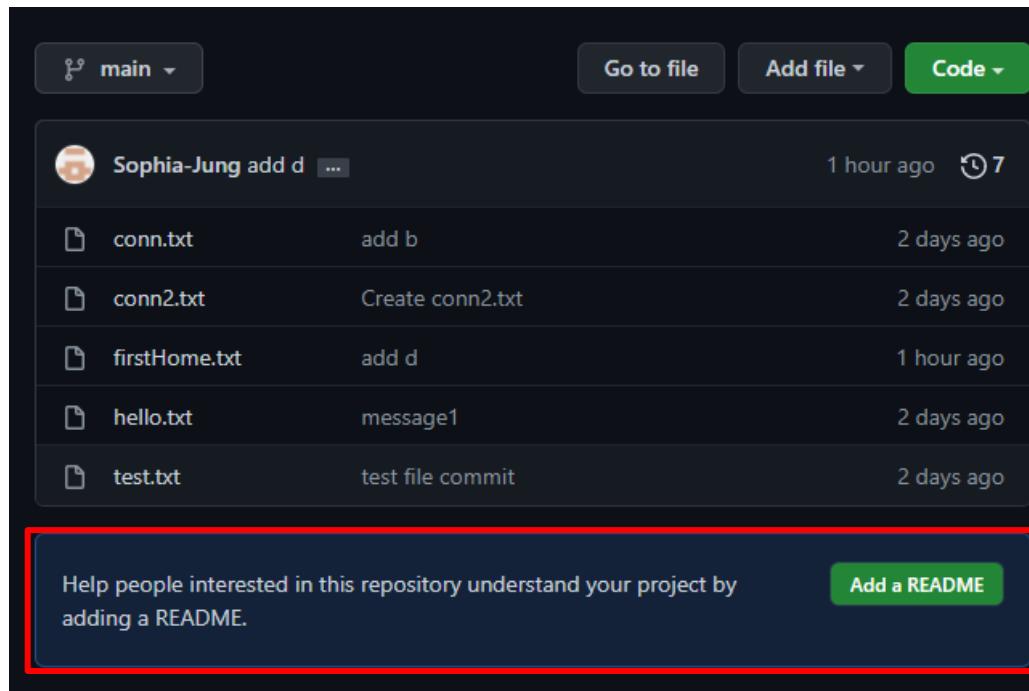
README 파일 작성하기

README 파일 작성하기

❖ README 파일

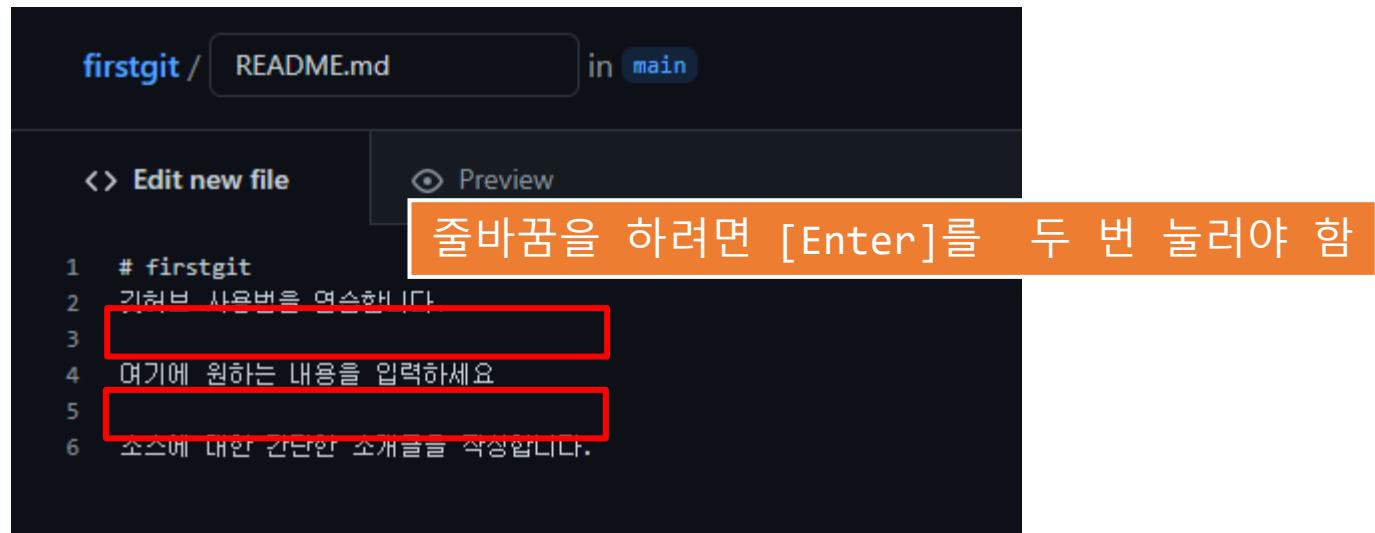
- 프로젝트나 저장소의 소개서 같은 역할을 하는 파일

❖ README 파일 작성하기



README 파일 작성하기

❖ README 파일에 내용 입력하기



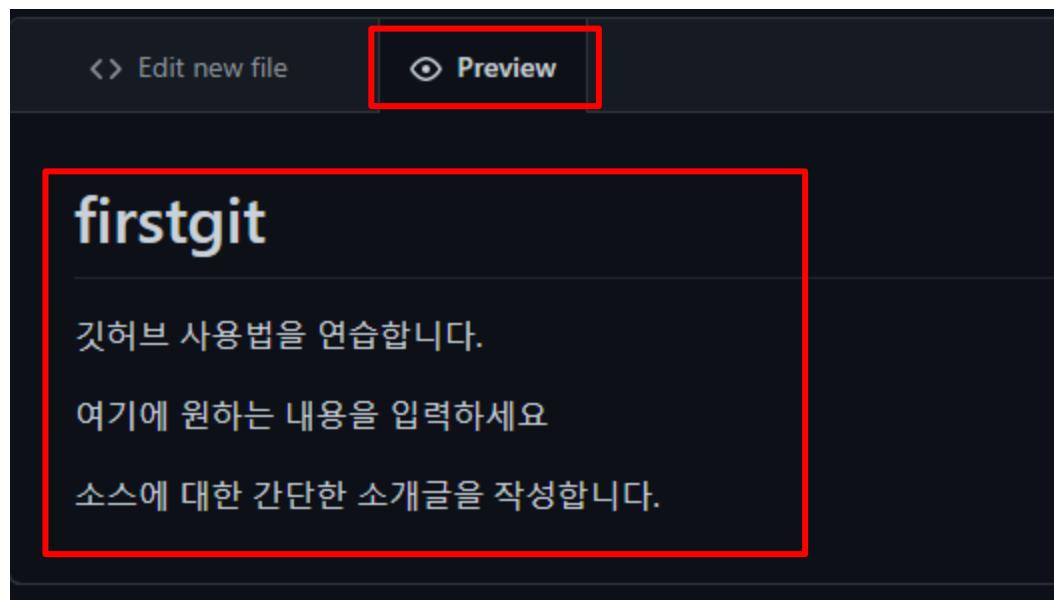
The screenshot shows a GitHub code editor interface. At the top, it displays the repository name "firstgit" and the file path "README.md". Below this, there are two tabs: "Edit new file" and "Preview". The "Preview" tab is selected, showing the following content:

```
1 # firstgit
2 기획부 사용법을 연습합니다.
3
4 여기에 원하는 내용을 입력하세요
5
6 소스에 대한 간단한 소개글을 작성합니다.
```

A large orange callout box highlights the text "줄바꿈을 하려면 [Enter]를 두 번 눌러야 함" (To create a new line, you need to press Enter twice) in the preview area. Two red boxes highlight specific lines of code: line 3 and line 6.

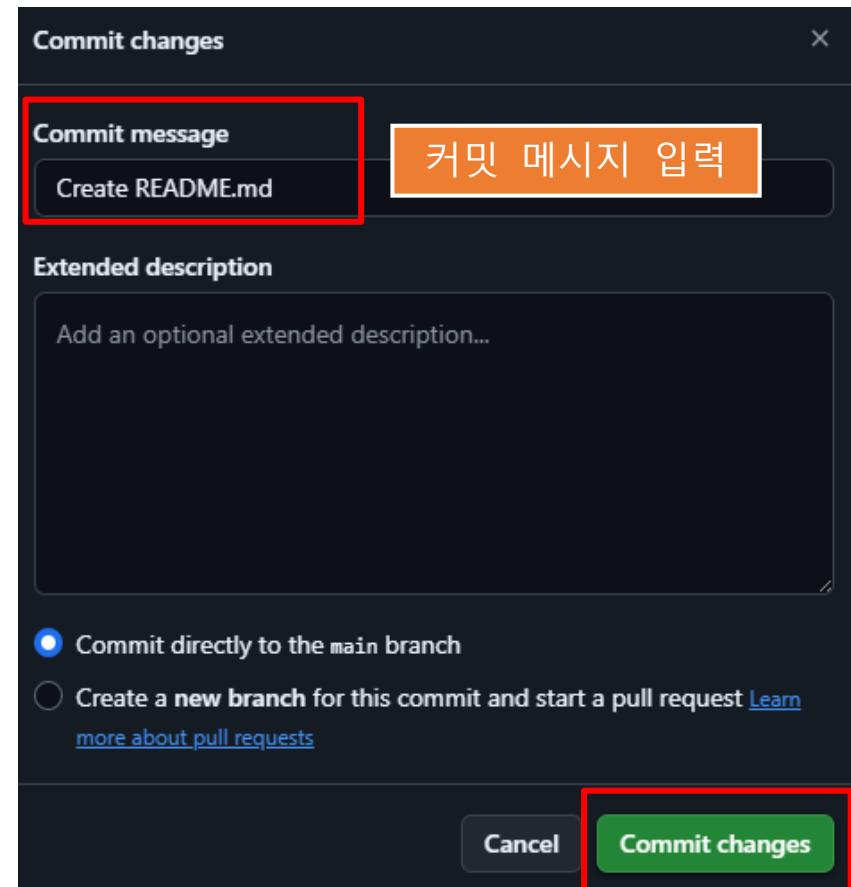
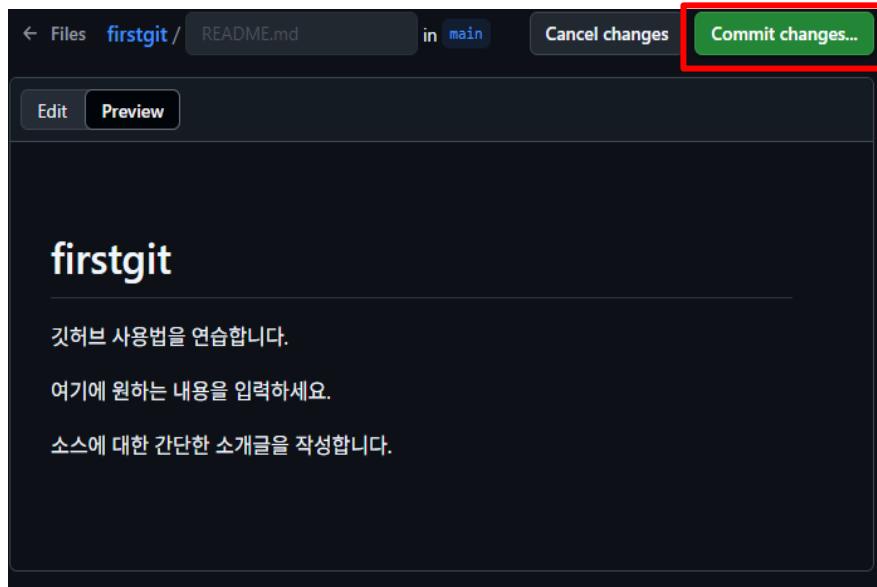
README 파일 작성하기

❖ README 파일 미리보기



README 파일 작성하기

❖ README 파일 커밋하기



README 파일 작성하기

❖ 저장소의 첫 화면

The screenshot shows a GitHub repository interface. At the top, it displays 'main' branch, '1 branch', '0 tags', 'Go to file', 'Add file', and 'Code' buttons. Below this is a list of commits:

Commit	Message	Time
.gitignore	message3	last week
README.md	Create README.md	now
bye.txt	Revert "test commit 2"	last week
coding.txt	message3	last week
conn.txt	add b	5 days ago
conn2.txt	Create conn2.txt	5 days ago
firstHome.txt	add 4	3 hours ago
hello.txt	modify hello.txt	3 hours ago

Below the commits, the 'README.md' file is shown with its content:

```
firstgit

깃허브 사용법을 연습합니다.

여기에 원하는 내용을 입력하세요.

소스에 대한 간단한 소개들을 작성합니다.
```

README 파일 작성하기

❖ README 파일 수정하기

The screenshot shows a GitHub repository interface. At the top, it displays 'main' branch, 1 branch, 0 tags, and 16 commits. The commit history is listed below:

File	Message	Time
.gitignore	message3	last week
README.md	Create README.md	now
bye.txt	Revert "test commit 2"	last week
coding.txt	message3	last week
conn.txt	add b	5 days ago
conn2.txt	Create conn2.txt	5 days ago
firstHome.txt	add 4	3 hours ago
hello.txt	modify hello.txt	3 hours ago

Below the commit history is the 'README.md' file content:

```
firstgit
```

깃허브 사용법을 연습합니다.

여기에 원하는 내용을 입력하세요.

소스에 대한 간단한 소개들을 작성합니다.

A red box highlights the edit icon (pencil) next to the file name in the README.md section.

03

마크다운 문법

마크다운 문법

❖ 마크다운 문법

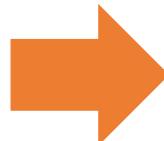
- 간단한 문법만으로도 문서의 구조(제목, 목록 등)를 표현할 수 있게 해주는 문서 형식
- 파일의 확장자는 .md 형식
 - 예) README.md
- 장점
 - 사용이 쉽고, HTML 태그에 비해 간단
 - 문서 작성이 편리함
- 단점
 - 마크다운을 지원하는 프로그램이나 사이트에서만 사용 가능

마크다운 문법

❖ 제목 만들기

- 텍스트 앞에 #을 붙임
- #을 1~6개까지 붙여서 글자 크기 조정

```
# 첫번째 제목  
## 두번째 제목  
### 세번째 제목  
#### 네번째 제목  
##### 다섯번째 제목  
##### 여섯번째 제목
```



첫번째 제목

두번째 제목

세번째 제목

네번째 제목

다섯번째 제목

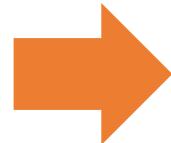
여섯번째 제목

마크다운 문법

❖ 텍스트 단락 바꾸기

- 텍스트 단락을 바꾸려면 [Enter]키를 두 번 눌러야 함
- [Enter]키를 한 번 누른 경우

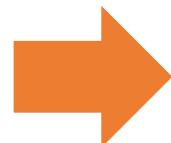
```
# 첫번째 제목  
안녕하세요.  
정수아입니다.
```



```
첫번째 제목  
_____  
안녕하세요. 정수아입니다.
```

- [Enter]키를 두 번 누른 경우

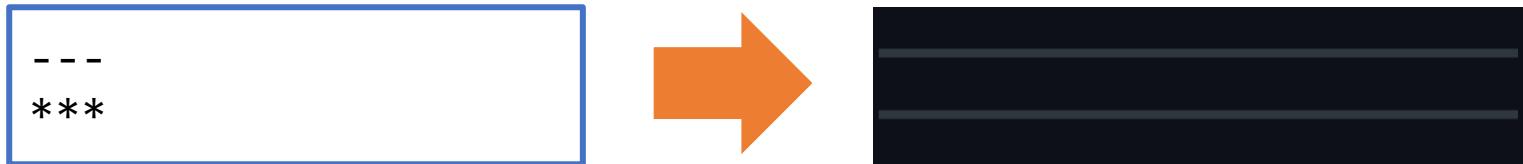
```
# 첫번째 제목  
안녕하세요.  
  
정수아입니다.
```



```
첫번째 제목  
_____  
안녕하세요.  
정수아입니다.
```

마크다운 문법

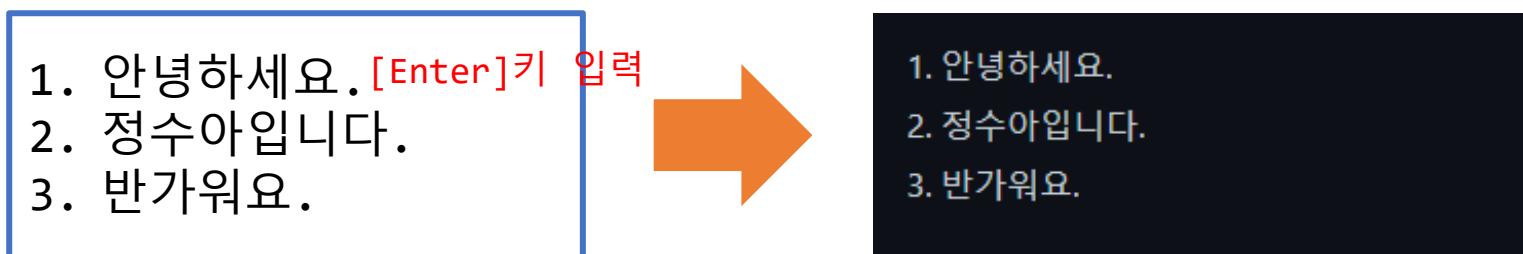
❖ 내용에 가로줄 추가하기



❖ 순서 있는 목록 만들기(1, 2, 3, ...)

- 목록은 항목 전체가 하나의 단락이기 때문에 [Enter]키를 한 번만 눌러서 줄을 바꿈

"2."
자동 생성



마크다운 문법

❖ 순서 없는 목록 만들기

- +, -, * 기호를 붙여서 작성하면 자동으로 글머리 기호가 생성

"_"
자동 생성

- 안녕하세요. [Enter] 키 입력
- 정수아입니다.
- 반가워요.



- 안녕하세요
- 정수아입니다.
- 반가워요.

- 템(Tab) 키를 눌러 항목을 들여 쓰면 여러 단계를 가진 목록 생성 가능

- + 안녕하세요
- + 정수아입니다.
- + 반가워요.



- 안녕하세요
- 정수아입니다.
- 반가워요.

마크다운 문법

❖ 텍스트 강조하기

- 텍스트의 일부분을 굵게 또는 기울임체로 강조

굵게	텍스트의 앞뒤를 ** 또는 __로 감싸기
기울임체	텍스트의 앞뒤를 * 또는 _로 감싸기
굵은 기울임체	텍스트의 앞뒤를 *** 또는 ___로 감싸기
취소선	텍스트의 앞뒤를 ~~로 감싸기

안녕하세요

정수아 입니다.

~~반가워요.~~



안녕하세요

정수아입니다.

반가워요.

마크다운 문법

❖ 소스코드 삽입하기

- 소스코드를 삽입하려면 backquote(`) 키를 사용

```
# 소스코드 삽입하기
```

1. 한 줄 소스 코드

```
`function add(x, y){ return x + y }`
```

2. 여러 줄 소스코드

```
```python
```

```
number = input("정수입력>")
number = int(number)
```

```
if number>0:
```

```
 print("양수입니다")
```

```
if number<0:
```

```
 print("음수입니다")
```

```
if number==0:
```

```
 print("0입니다")
```

```
```
```

소스코드 삽입하기

1. 한 줄 소스 코드 `function add(x, y) { return x + y }`

2. 여러 줄 소스코드

```
number = input("정수입력>")  
number = int(number)
```

```
if number>0:  
    print("양수입니다")
```

```
if number<0:  
    print("음수입니다")
```

```
if number==0:  
    print("0입니다")
```

마크다운 문법

❖ 링크 삽입하기

- <링크 주소>
- [링크 텍스트](링크 주소)

```
<https://cafe.naver.com/hicode>
```

```
[Hello코딩놀이터](https://cafe.naver.com/hicode)
```

<https://cafe.naver.com/hicode>

[Hello코딩놀이터](#)

Git & GitHub

- ◆ 오픈소스 저장소 사용하기 - Python

정수아

Contents

- 01** 오픈소스 저장소 복제하기
- 02** 오픈소스 저장소에 합치기 요청
- 03** 오픈소스 저장소에 합치기 수락

01

오픈소스 저장소 복제

오픈소스 저장소 복제

❖ 오픈소스 저장소로 이동

<https://github.com/HelloCoding22/Python-Game.git>

클릭

The screenshot shows a GitHub repository page for 'HelloCoding22 / Python-Game'. The 'Code' tab is selected. At the top right, there are three buttons: 'Watch 1', 'Fork 0' (which is highlighted with a red box), and 'Star 0'. Below the tabs, it shows 'main' branch, '1 branch', '0 tags', 'Go to file', 'Add file', and a green 'Code' dropdown. The main area lists a commit by 'HelloCoding22' creating 'game.py' 3 days ago, and a file named 'README.md'. On the right, there's an 'About' section with the following details:
오픈소스 저장소 클론 수업-Python 버전
Readme
0 stars
1 watching
0 forks

오늘 오픈소스 저장소 클론 수업-Python 버전

오픈소스 저장소 복제

❖ 본인 계정에 Python-Game 저장소 복제

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner * Repository name *

Sophia-Jung / Python-Game ✓

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

오픈소스 저장소 클론 수업-Python 버전

Copy the main branch only
Contribute back to HelloCoding22/Python-Game by adding your own branch. [Learn more.](#)

ⓘ You are creating a fork in your personal account.

Create fork

오픈소스 저장소 복제

❖ 본인 계정에 Python-Game 저장소 복제

Sophia-Jung / Python-Game Public

forked from HelloCoding22/Python-Game

Pin Watch 0 Fork 1 Star 0

Code Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

This branch is up to date with HelloCoding22/Python-Game:main. Contribute Sync fork

HelloCoding22 Create game.py 4722890 3 days ago 3 commits

README.md Update README.md 3 days ago

game.py Create game.py 3 days ago

About

오픈소스 저장소 클론 수업-Python 버전

Readme 0 stars 0 watching 1 fork

Releases

No releases published Create a new release

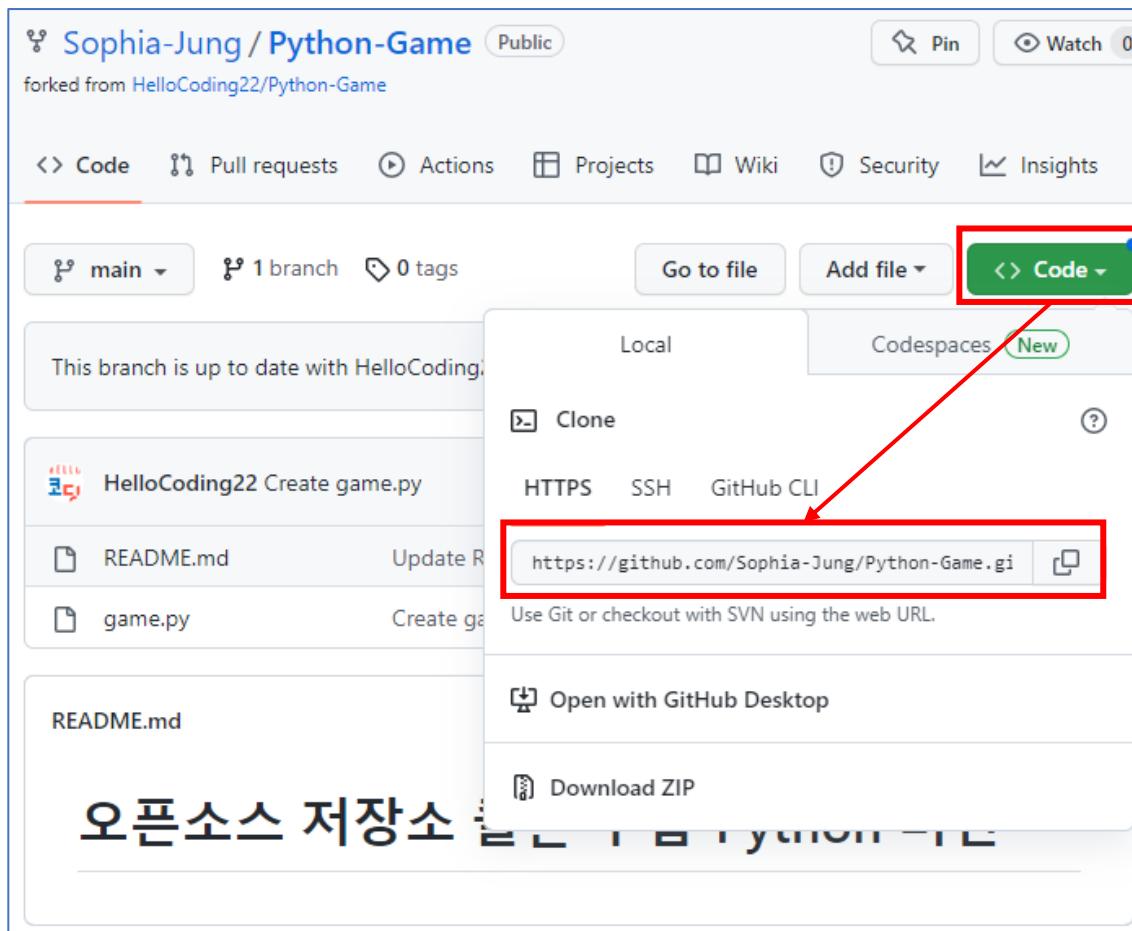
Packages

No packages published

오픈소스 저장소 클론 수업-Python 버전

오픈소스 저장소 복제

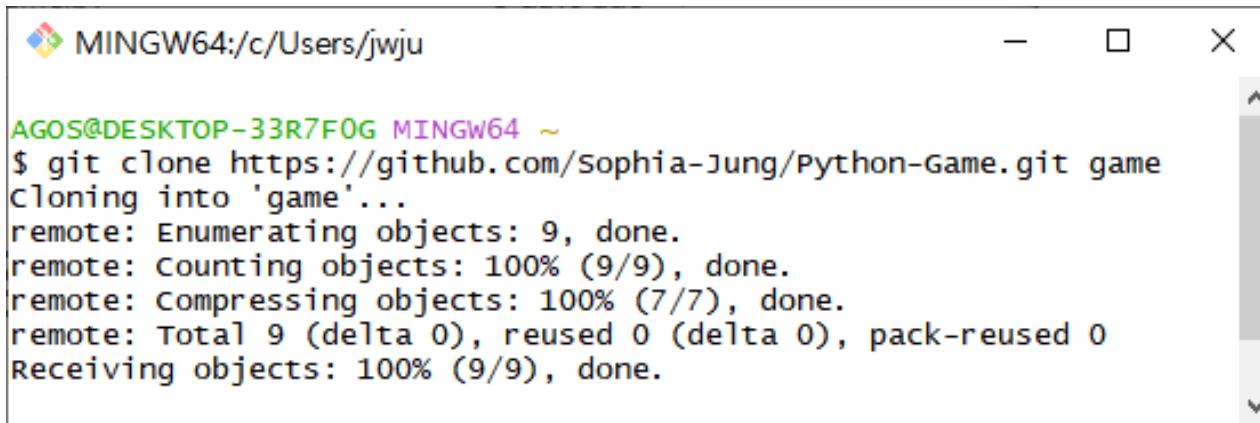
❖ 복제한 저장소 주소 복사



오픈소스 저장소 복제

❖ 복제된 저장소를 지역저장소로 클론하기

```
$ git clone 복사한_저장소_주소 game
```



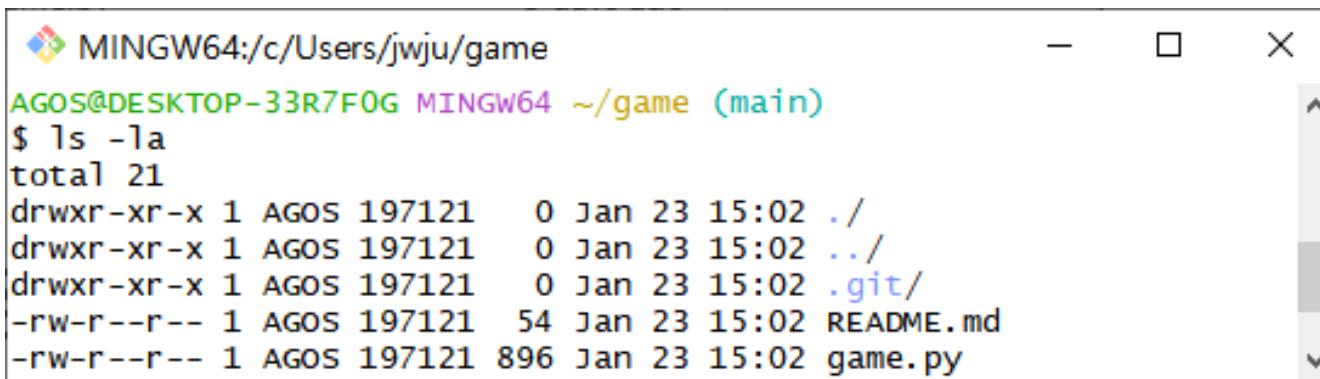
The screenshot shows a terminal window titled 'MINGW64:/c/Users/jwju'. The command \$ git clone https://github.com/Sophia-Jung/Python-Game.git game was run, and the output shows the progress of cloning a repository from GitHub. The output includes:

```
AGOS@DESKTOP-33R7F0G MINGW64 ~
$ git clone https://github.com/Sophia-Jung/Python-Game.git game
Cloning into 'game'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), done.
```

오픈소스 저장소 복제

❖ 지역저장소로 복사되었는지 확인

```
$ cd game  
$ ls -la
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/jwju/game'. The window title bar includes icons for minimize, maximize, and close. The terminal prompt is 'AGOS@DESKTOP-33R7F0G MINGW64 ~/game (main)'. The user has run the command '\$ ls -la' which lists the contents of the directory:

```
total 21  
drwxr-xr-x 1 AGOS 197121 0 Jan 23 15:02 ./  
drwxr-xr-x 1 AGOS 197121 0 Jan 23 15:02 ../  
drwxr-xr-x 1 AGOS 197121 0 Jan 23 15:02 .git/  
-rw-r--r-- 1 AGOS 197121 54 Jan 23 15:02 README.md  
-rw-r--r-- 1 AGOS 197121 896 Jan 23 15:02 game.py
```

오픈소스 저장소 복제

❖ game.py 소스 코드 수정하기



The screenshot shows a terminal window titled "MINGW64:/c/Users/KBS/game". The window contains a Python script for a game. The title bar has a red box around the text "# 소스 코드 수정 중". The terminal output shows the script's logic for determining the computer's move and comparing it with the user's input. The bottom status bar indicates the file is "+", the date is "10/01/2023", and the cursor is at "5,1 Top".

```
import random
num1 = int(input("하나를 선택하세요 : 가위(0), 바위(1), 보(2) :"))
num2 = random.randrange(0, 3)

if num2 == 0:
    print("컴퓨터는 가위를 냈습니다.")
elif num2 == 1:
    print("컴퓨터는 바위를 냈습니다.")
else:
    print("컴퓨터는 보를 냈습니다.")

if num1 == 0:
    if num2 == 1:
        print("컴퓨터가 이겼습니다")
    elif num2 == 2:
        print("당신이 이겼습니다.")
    else:
        print("비겼습니다.")
elif num1 == 1:
    if num2 == 0:
        print("당신이 이겼습니다.")
    elif num2 == 2:
        print("컴퓨터가 이겼습니다.")
    else:
        print("비겼습니다.")
else:
    print("비겼습니다.")
```

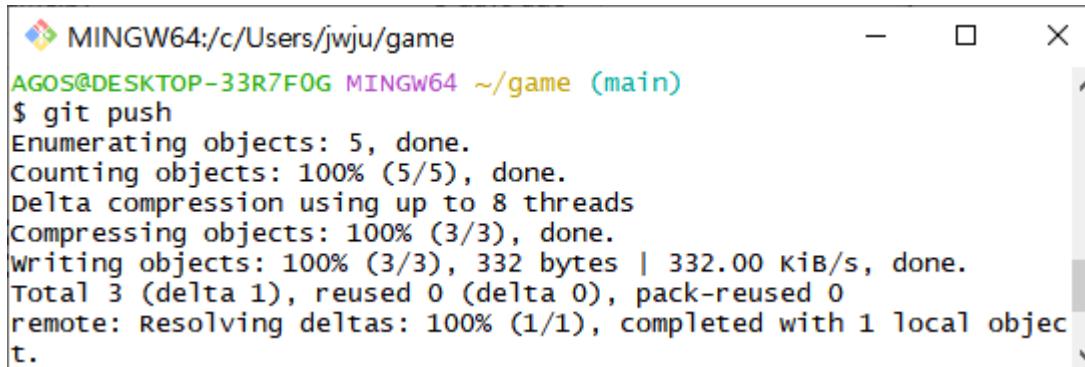
오픈소스 저장소 복제

❖ 커밋하기

```
$ git commit -am "add my text"
```

❖ 원격 저장소에 올리기

```
$ git push
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/jwju/game'. The command \$ git push is run, followed by a series of status messages indicating the progress of the push operation:

```
AGOS@DESKTOP-33R7F0G MINGW64 ~/game (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 332 bytes | 332.00 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.
```

오픈소스 저장소 복제

❖ game.py 커밋 완료

The screenshot shows two GitHub pages. The top page is a pull request from a branch named 'main' to 'HelloCoding22/main'. It has 1 branch, 0 tags, and 4 commits. One commit by Sophia-Jung adds text to 'game.py'. The bottom page is the 'game.py' file in the 'Python-Game' repository. It shows the same commit history, including the addition of Korean text '# 소스 코드 수정 중' to line 1. A red box highlights the commit message in the pull request and the Korean text in the file content.

This branch is 1 commit ahead of HelloCoding22:main.

Sophia-Jung add my text

README.md Update README.md 3 days ago

game.py add my text

Sophia-Jung add my text

2 contributors

35 lines (31 sloc) | 891 Bytes

1 # 소스 코드 수정 중

복제한 저장소의 파일은 수정되었지만,
원본 소스가 있던 저장소에는 반영되지 않았음

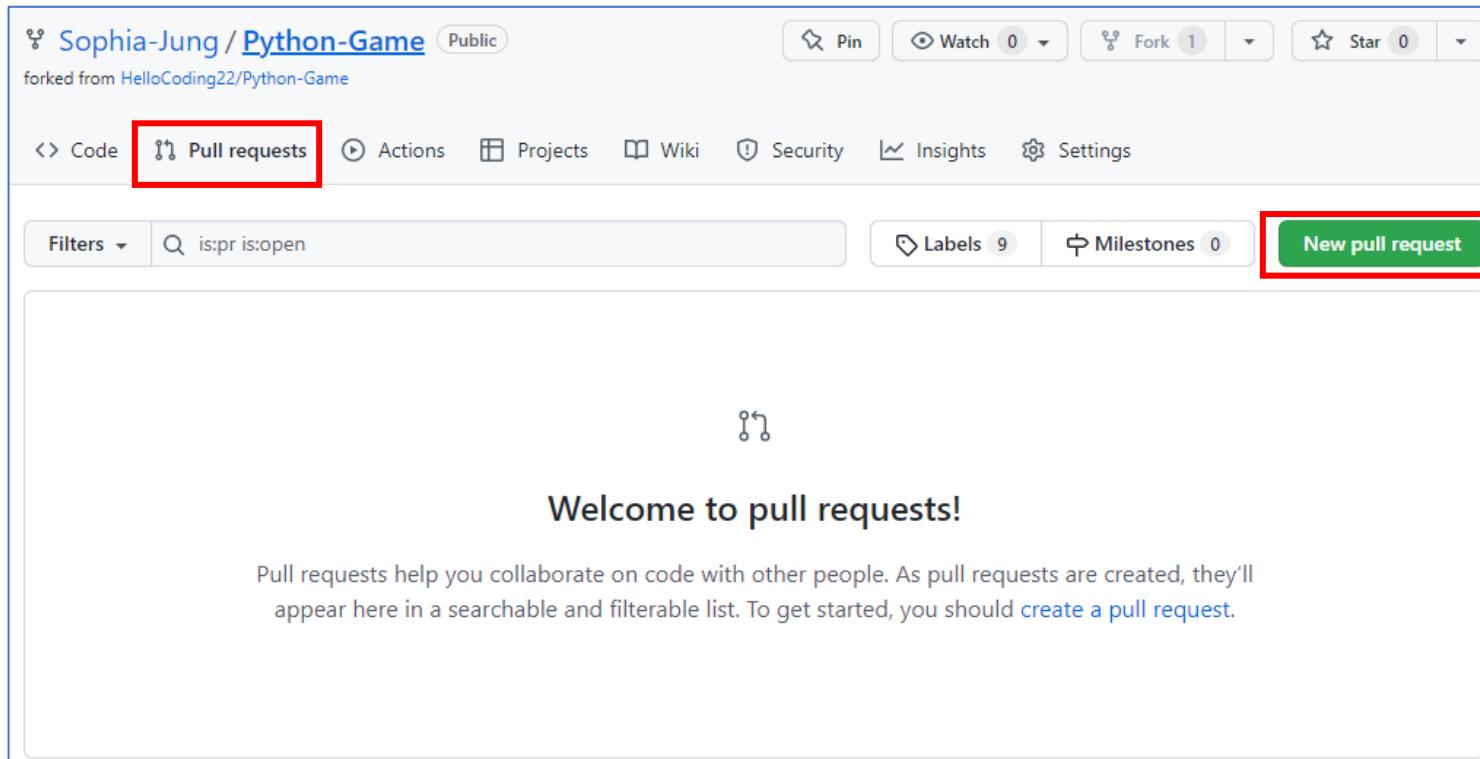
02

오픈소스 저장소에
합치기 요청

오픈소스 저장소에 합치기 - 요청

❖ 풀 리퀘스트(pull request)

- 원본 저장소의 개발자에게 수정한 내용을 반영해 달라는 요청



오픈소스 저장소에 합치기 - 요청

- ❖ 원본 저장소의 파일 vs 복제된 저장소의 파일
 - 변경 사항을 보여줌

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base repository: HelloCoding22/Python-Game base: main head repository: Sophia-Jung/Python-Game compare: main

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#) [Create pull request](#)

-o 1 commit 1 file changed 1 contributor

Commits on Jan 23, 2023

add my text
Sophia-Jung committed 2 minutes ago 889cb17

Showing 1 changed file with 2 additions and 0 deletions. [Split](#) [Unified](#)

game.py

```
@@ -1,3 +1,5 @@
 1 + # 소스 코드 수정 중
 2 +
 3 import random
 4 num1 = int(input("하나를 선택하세요 : 가위(0), 바위(1), 보(2) :"))
 5 num2 = random.randrange(0, 3)
```

오픈소스 저장소에 합치기 - 요청

❖ 상단 “Create pull request” 버튼 클릭

The screenshot shows a GitHub interface for comparing changes between two repositories. At the top, it says "Comparing changes" and "Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks." Below this, there are dropdown menus for "base repository" (HelloCoding22/Python-Game), "base" (main), "head repository" (Sophia-Jung/Python-Game), and "compare" (main). A green checkmark indicates "Able to merge. These branches can be automatically merged." In the center, there's a text box for discussing changes with others, followed by a large green "Create pull request" button, which is outlined in red. Below the discussion box, it says "Discuss and review the changes in this comparison with others. Learn about pull requests". Underneath, it shows "1 commit", "1 file changed", and "1 contributor". The commit details show a single commit from "Sophia-Jung" on Jan 23, 2023, titled "add my text". The file changed is "game.py" with 2 additions and 0 deletions. The code diff shows the addition of Korean comments and imports.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: HelloCoding22/Python-Game base: main head repository: Sophia-Jung/Python-Game compare: main

Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

-o 1 commit 1 file changed 1 contributor

Commits on Jan 23, 2023

add my text
Sophia-Jung committed 2 minutes ago

Showing 1 changed file with 2 additions and 0 deletions.

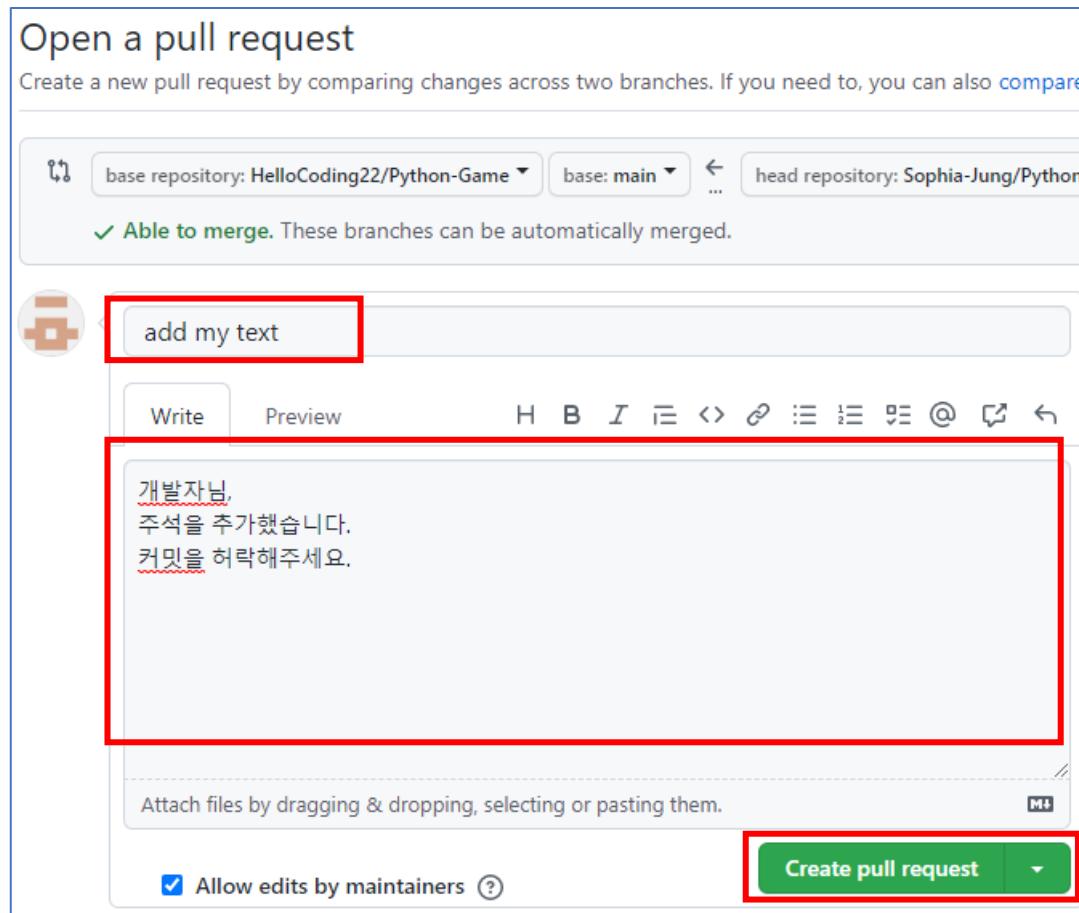
Split Unified

v 2 game.py

| | | |
|-----|-----|---|
| ... | ... | @@ -1,3 +1,5 @@ |
| 1 | 2 | + # 소스 코드 수정 중 |
| 3 | 4 | import random |
| 4 | 5 | num1 = int(input("하나를 선택하세요 : 가위(0), 바위(1), 보(2) :")) |
| 5 | | num2 = random.randrange(0, 3) |

오픈소스 저장소에 합치기 - 요청

❖ 커밋에 대한 설명 추가



오픈소스 저장소에 합치기 - 요청

❖ 개발자와 소통하는 공간

- 개발자와 질문과 답변을 주고 받으면서 수정한 내용을 원본 소스에 반영할지 여부를 결정

The screenshot shows a GitHub pull request titled "add my text #1". The pull request is from the user "Sophia-Jung" into the repository "HelloCoding22:main". The commit message is "add my text". A comment from "Sophia-Jung" is visible, reading: "개발자님, 주석을 추가했습니다. 커밋을 허락해주세요." (Developer, I have added a comment. Please approve the commit.). The pull request has 1 commit and 1 file changed.

03

오픈소스 저장소에
합치기 수락

오픈소스 저장소에 합치기 - 수락

❖ 원본 저장소

The screenshot shows a GitHub repository page for 'HelloCoding22 / Python-Game'. The 'Pull requests' tab is selected, highlighted with a red box. A modal window titled 'Label issues and pull requests for new contributors' is open, providing instructions for GitHub's first-time contributor experience. Below the modal, there are filters for 'is:pr is:open', a 'Labels' section (9), a 'Milestones' section (0), and a green 'New pull request' button. At the bottom, there are sorting options like 'Author', 'Label', 'Projects', 'Milestones', 'Reviews', 'Assignee', and 'Sort'. A specific pull request is highlighted with a red box, labeled '#1 opened now by Sophia-Jung'.

HelloCoding22 / Python-Game Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests 1 Actions Projects Wiki Security ...

Label issues and pull requests for new contributors Dismiss

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

Filters is:pr is:open Labels 9 Milestones 0 New pull request

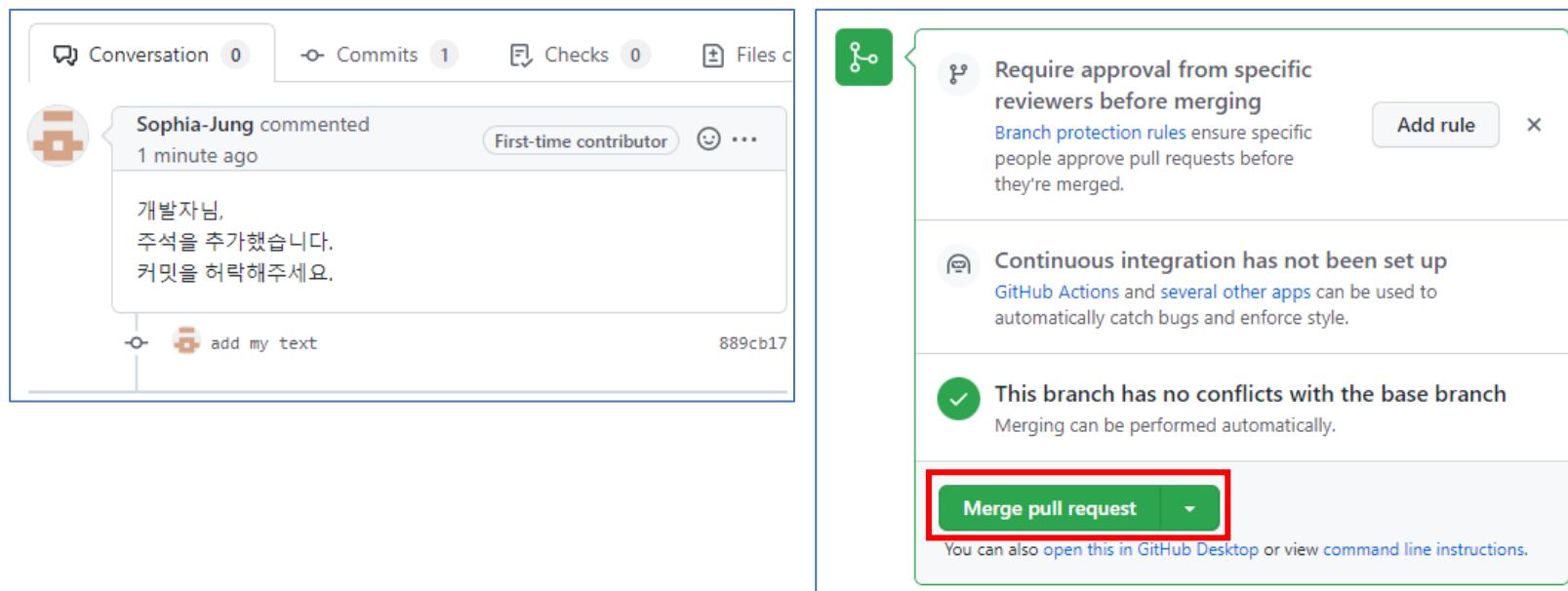
1 Open 0 Closed

Author Label Projects Milestones Reviews Assignee Sort

add my text #1 opened now by Sophia-Jung

오픈소스 저장소에 합치기 - 수락

- ❖ 다른 사용자가 보낸 수정 내용 검토 후, 수락 여부 결정
 - 수정을 허락하려면 Merge pull request 버튼 클릭



오픈소스 저장소에 합치기 - 수락

❖ 원본 저장소의 소스 코드에 수정사항 반영

The screenshot shows a GitHub repository page for 'HelloCoding22 / Python-Game'. A red box highlights the repository name 'HelloCoding22 / Python-Game'. Below the repository name, there are buttons for Pin, Unwatch (1), Fork (1), and Star (0). The 'Code' tab is selected. In the top navigation bar, there are links for Issues, Pull requests, Actions, Projects, Wiki, Security, and three dots. Below the navigation bar, there is a dropdown for the branch ('main') and a breadcrumb trail ('Python-Game / game.py'). To the right of the breadcrumb trail are 'Go to file' and three dots buttons. The main content area shows a commit from 'Sophia-Jung' with the message 'add my text'. It includes a timestamp 'Latest commit 889cb17 6 minutes ago' and a 'History' link. Below the commit, it says '2 contributors' with icons for Sophia-Jung and another user. At the bottom, it shows '35 lines (31 sloc) | 891 Bytes'. A red box highlights the first line of code: '# 소스 코드 수정 중'. The full code listing is:

```
1 # 소스 코드 수정 중
2
3 import random
4 num1 = int(input("하나를 선택하세요 : 가위(0), 바위(1), 보(2) :"))
5 num2 = random.randrange(0, 3)
```

Git & GitHub

- ◆ VS CODE에서 깃 사용하기

정수아

Contents

01 환경 설정

02 VS CODE에서 파일 커밋

03 원격 저장소에 push

04 README 파일 작성

05 원격 저장소에서 pull

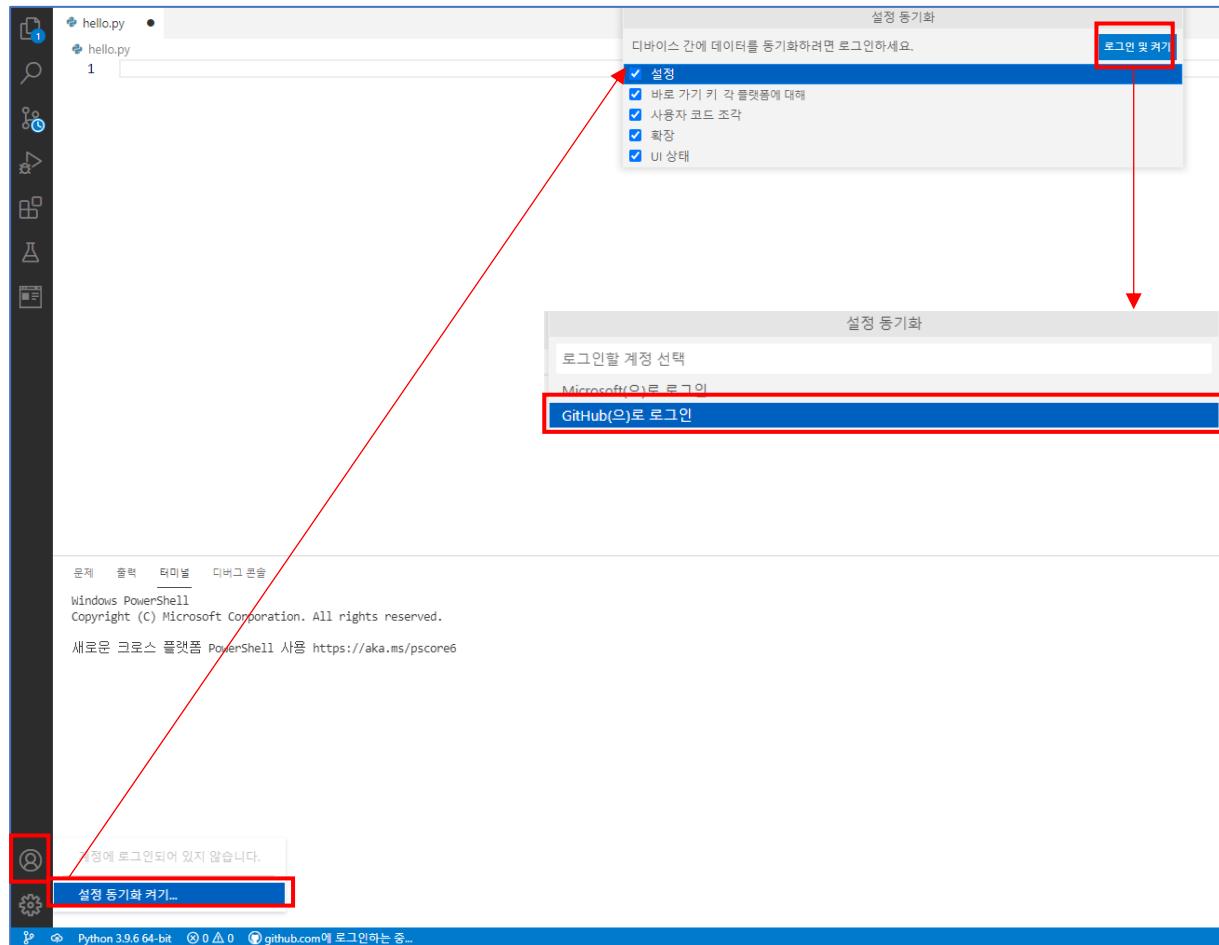
06 원격 저장소 복제

01

환경 설정

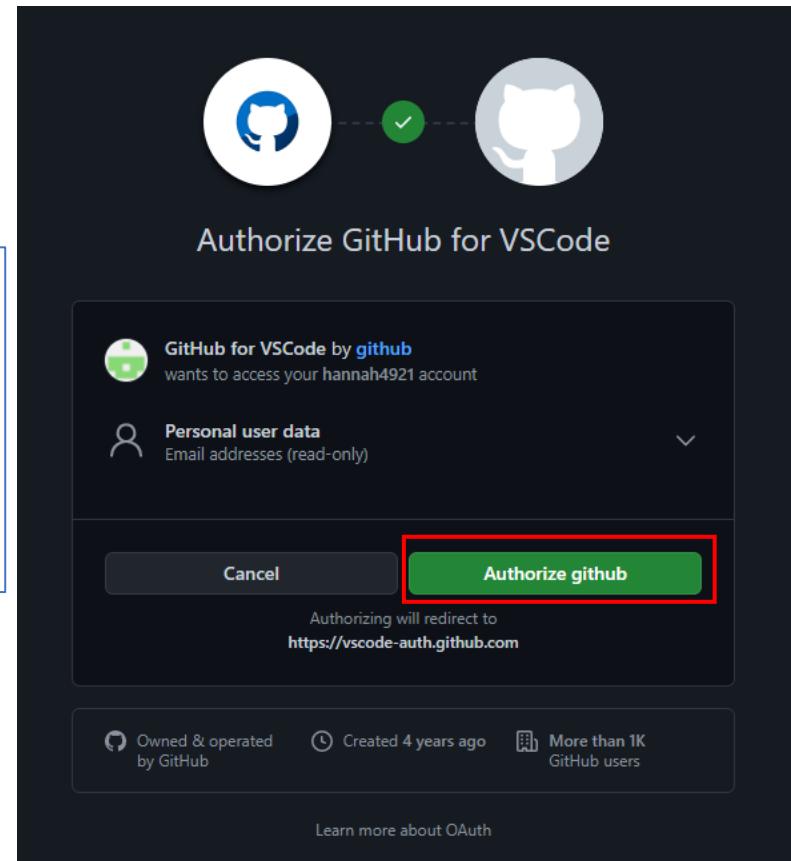
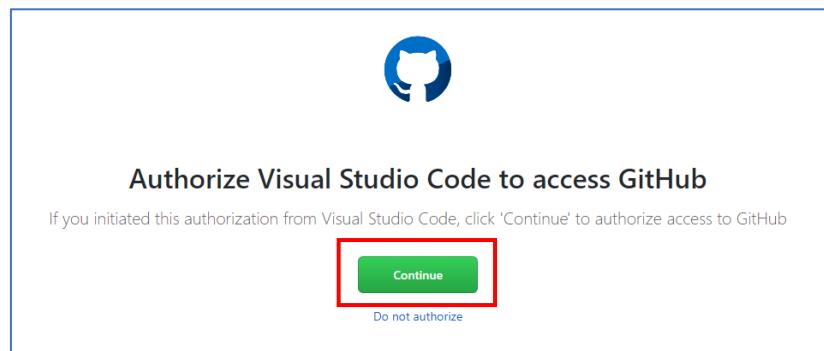
환경 설정

❖ GitHub 계정 로그인하기



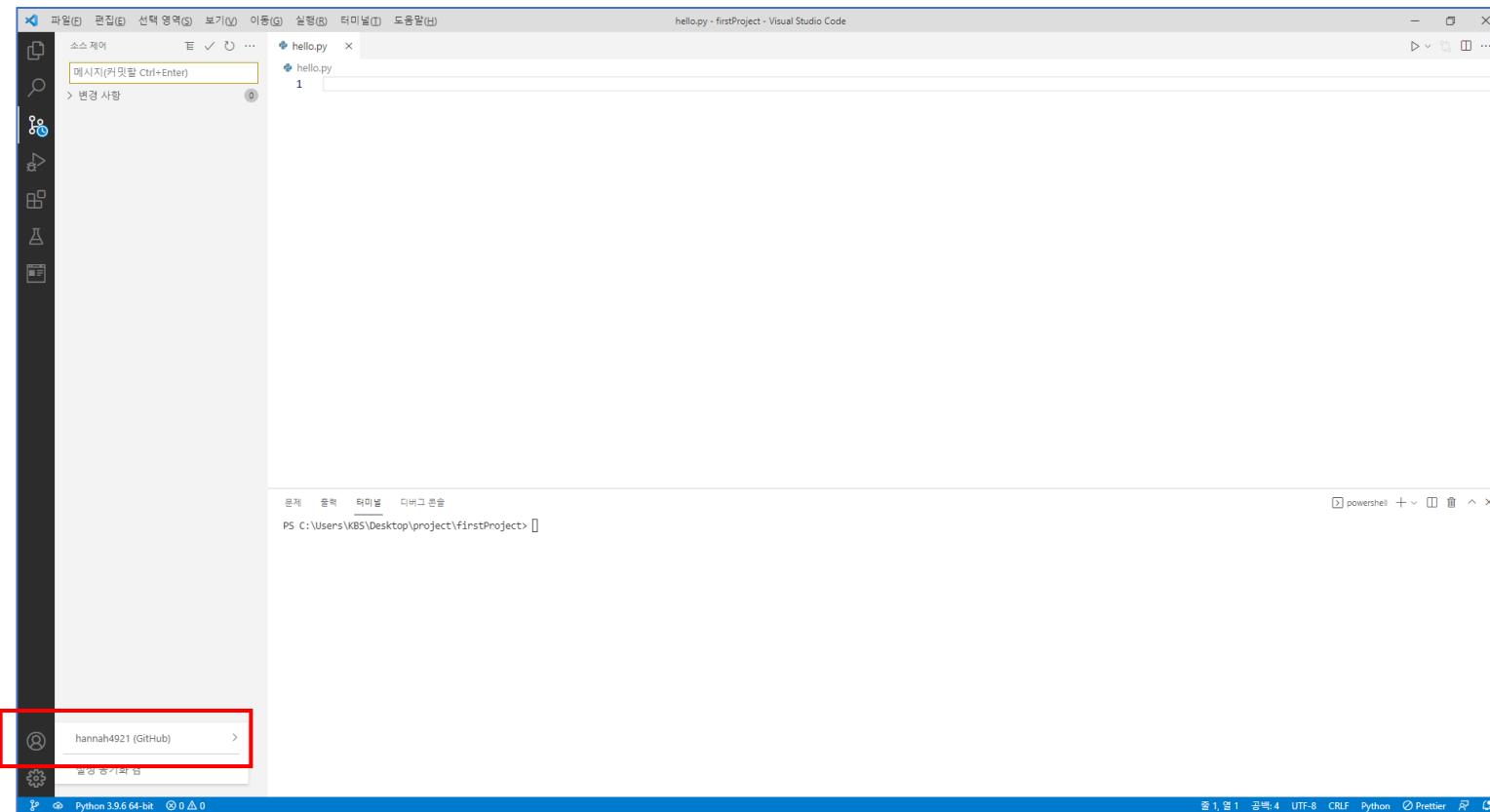
환경 설정

❖ 권한 수락



환경 설정

❖ 로그인 확인



02

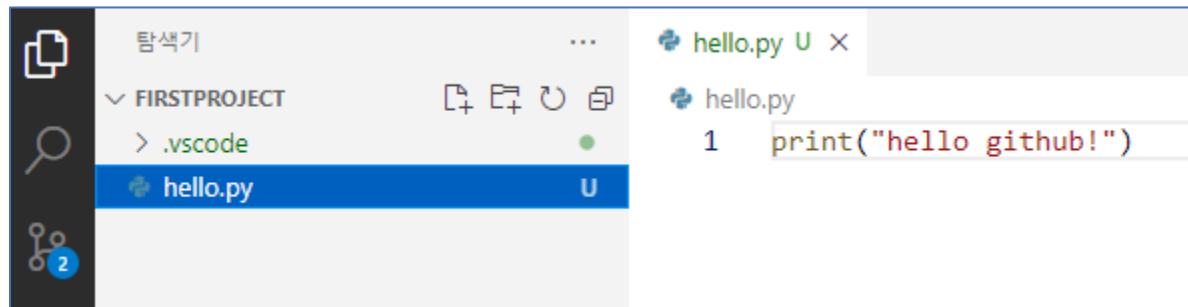
VSCODE에서 파일 커밋

VSCODE에서 파일 커밋

❖ 새 프로젝트 생성

- firstProject 폴더
- hello.py 파일

```
print("hello github!")
```



VSCODE에서 파일 커밋

❖ 깃 저장소 초기화

```
$ git init
```

❖ 터미널에 사용자 정보 입력

- 사용자 이름

```
$ git config --global user.name "Sooa"
```

- 사용자 이메일

```
$ git config --global user.email "Sooa@gmail.com"
```

VSCODE에서 파일 커밋

❖ 스테이지에 파일 올리기

```
$ git add hello.py
```

❖ 커밋하기

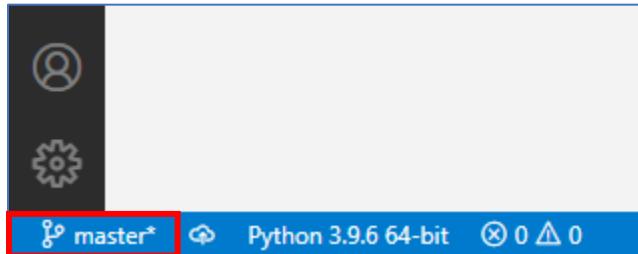
```
$ git commit -m "first commit"
```

03

원격저장소에 push

원격저장소에 push하기

- ❖ main 브랜치 생성 및 변경(master → main)



```
$ git branch -m main
```



원격저장소에 push하기

❖ 원격저장소 생성하기 - firstProject

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 *  HelloCoding22

Repository name * firstProject is available.

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

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:

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

Choose a license

License:

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

원격저장소에 push하기

❖ 지역저장소와 원격저장소 연결하기

```
$ git remote add origin 원격저장소_주소
```

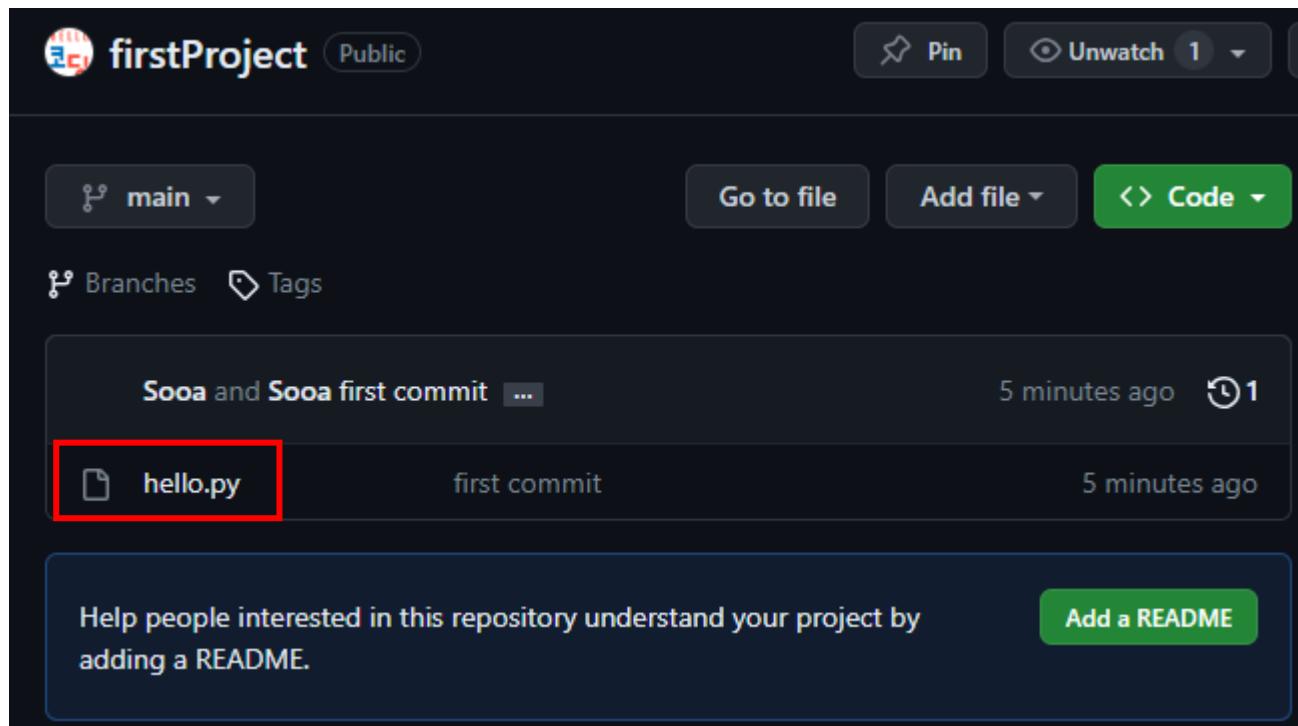
❖ 연결 확인하기

```
$ git remote -v
```

원격저장소에 push하기

❖ 원격저장소에 push하기

```
$ git push -u origin main
```

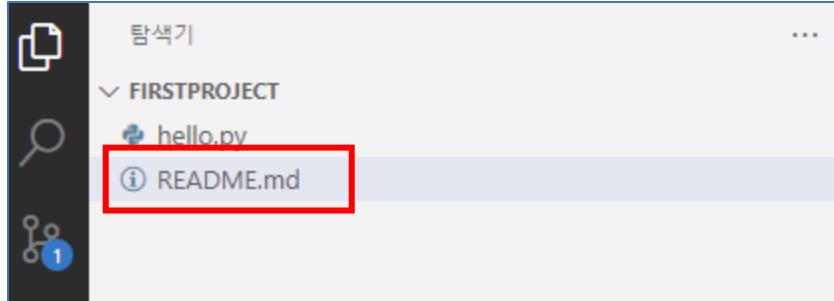


04

README 파일 작성

README 파일 작성

❖ README.md 파일 생성



README 파일 작성

❖ README 파일 작성

GitHub 연습

비주얼 스튜디오 코드를 사용해서 GitHub를 연습합니다.

README.md 파일을 작성 중입니다.

README 파일 작성

❖ 스테이지에 올리기

```
$ git add README.md
```

❖ 커밋하기

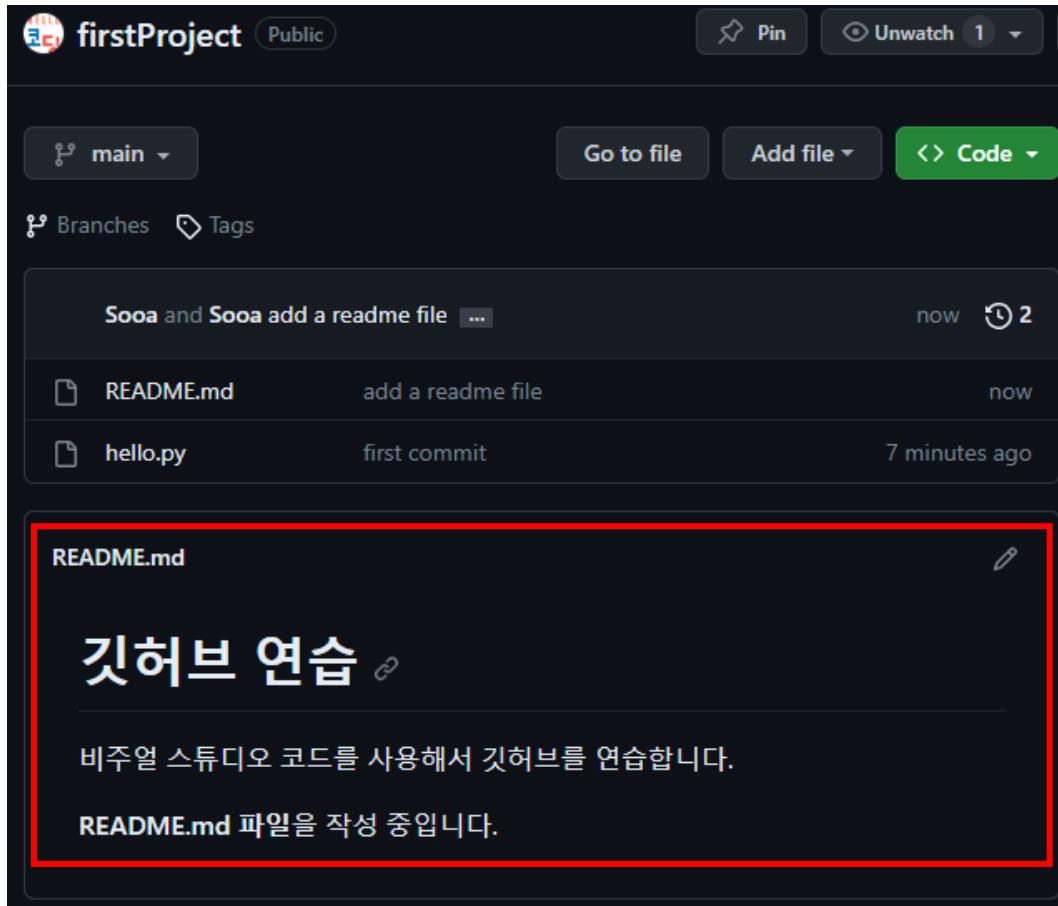
```
$ git commit -m "add a readme file"
```

❖ 원격 저장소에 push하기

```
$ git push
```

README 파일 작성

❖ README 파일 push 완료



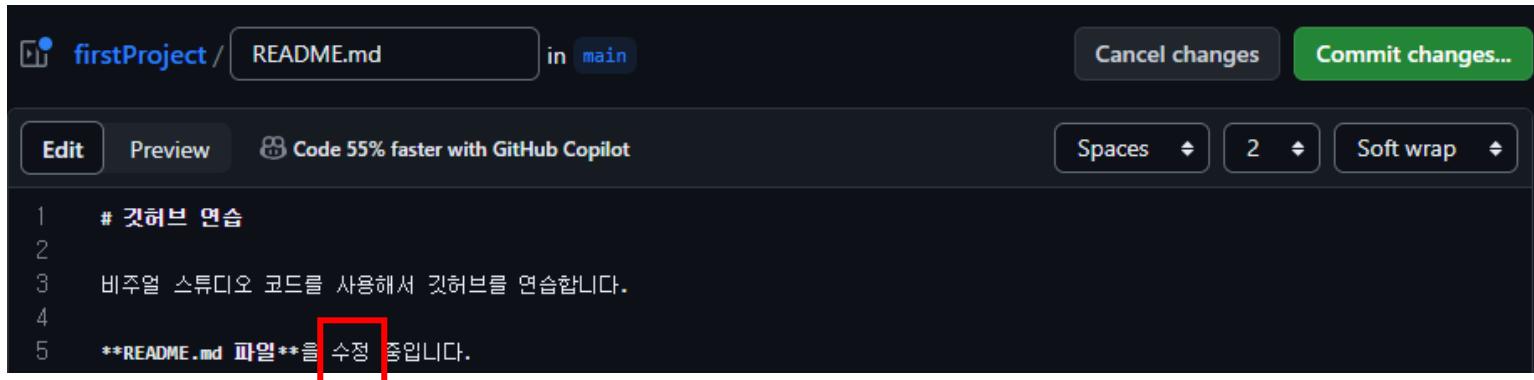
05

원격저장소에서 pull

원격저장소에서 pull하기

❖ 원격저장소에서 pull하기

- 원격저장소를 pull하기 위해서는 원격저장소에 변경 사항이 있어야 함
 - 연결된 원격저장소에서 직접 파일 수정 후 커밋하기
 - 예) README.md 파일 수정



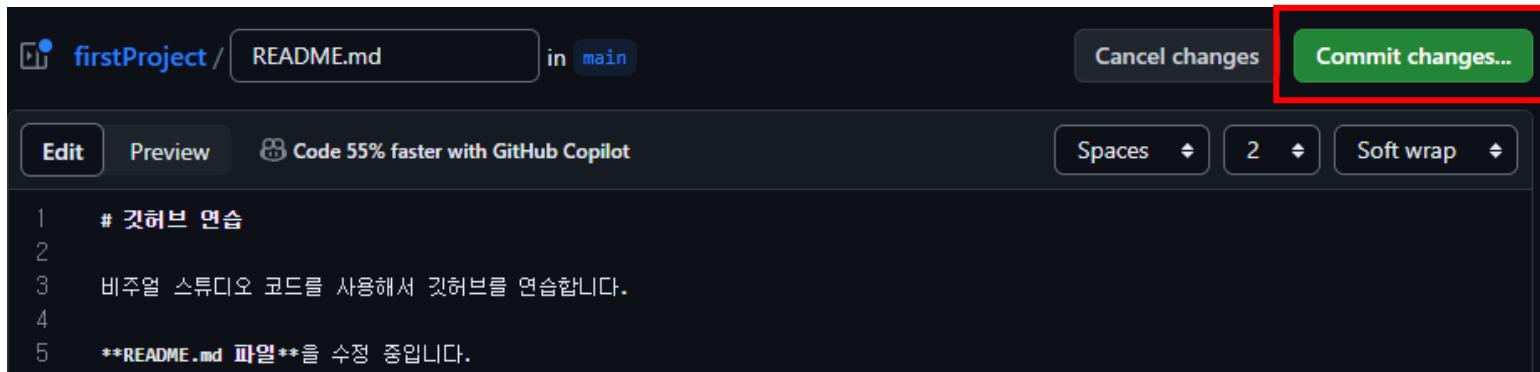
The screenshot shows the GitHub code editor interface for a project named 'firstProject'. The file being edited is 'README.md' in the 'main' branch. The editor displays the following content:

```
1 # 깃허브 연습
2
3 비주얼 스튜디오 코드를 사용해서 깃허브를 연습합니다.
4
5 **README.md 파일**을 수정 중입니다.
```

A red box highlights the text '**README.md 파일**을 수정 중입니다.' in the fifth line. At the top right of the editor, there are buttons for 'Cancel changes' and 'Commit changes...', and below them are settings for 'Spaces', '2', and 'Soft wrap'.

원격저장소에서 pull하기

❖ 커밋하기



원격저장소에서 pull하기

❖ 원격저장소 pull하기

```
$ git pull
```

❖ 원격저장소에서 수정한 내용이 즉시 반영

The screenshot shows a GitHub commit history. The first commit is from '깃허브 연습' and has the following message:

```
1 # 깃허브 연습  
2  
3 비주얼 스튜디오 코드를 사용해서 깃허브를 연습합니다.  
4  
5 **README.md 파일**을 수정 중입니다.  
6
```

06

원격저장소 복제

원격저장소 복제하기

❖ 원격저장소 복제하기

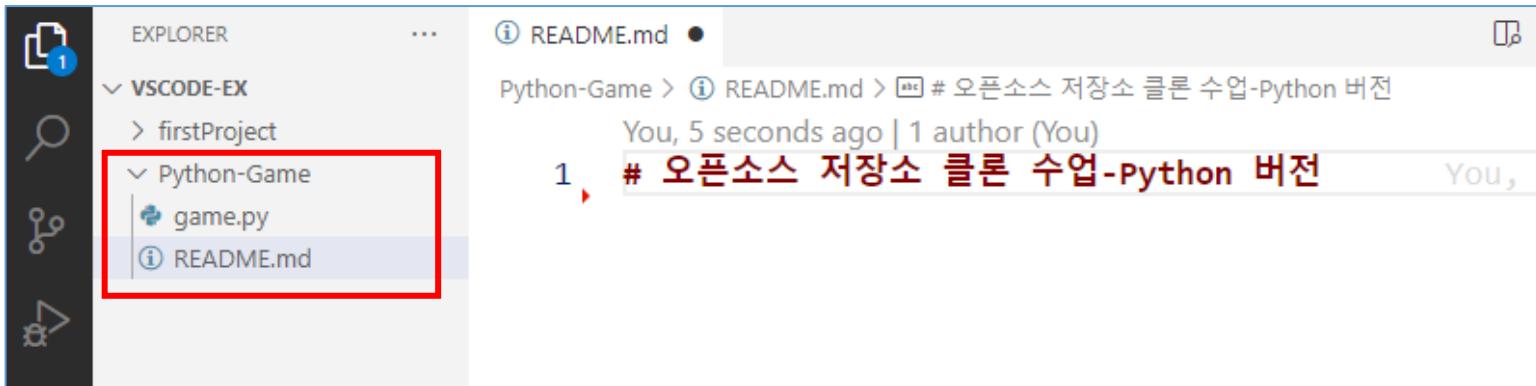
```
$ git clone 복제할_저장소_주소
```

- 예시

```
$ git clone https://github.com/HelloCoding22/Python-Game.git
```

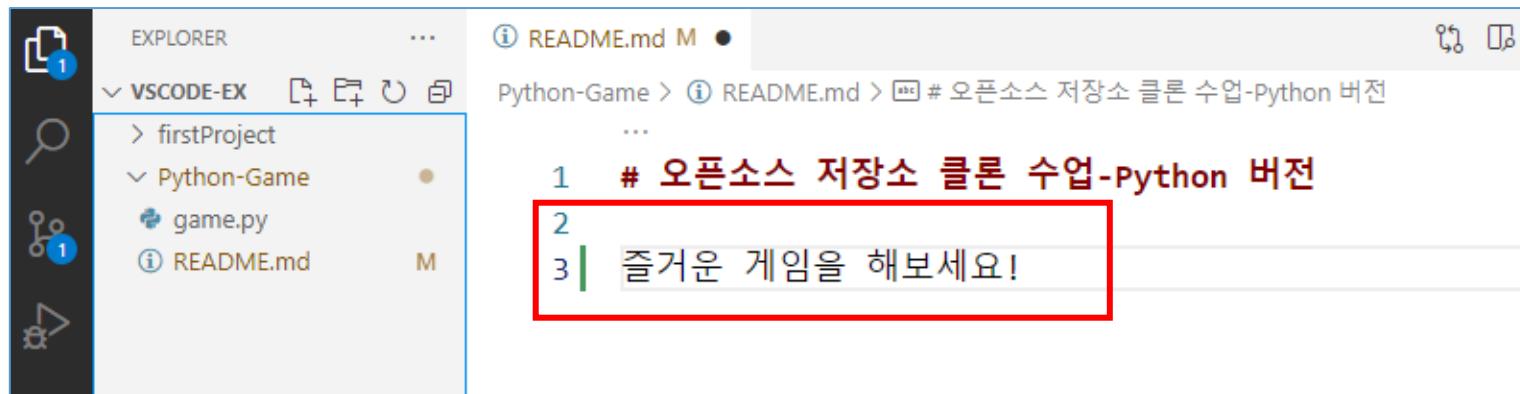
원격저장소 복제하기

❖ 원격저장소 복제 완료



지역 저장소에서 파일 수정하기

❖ 지역 저장소에서 복제한 파일 수정하기



원격 저장소에 push하기

❖ 스테이지에 올리기

```
$ git add README.md
```

❖ 커밋하기

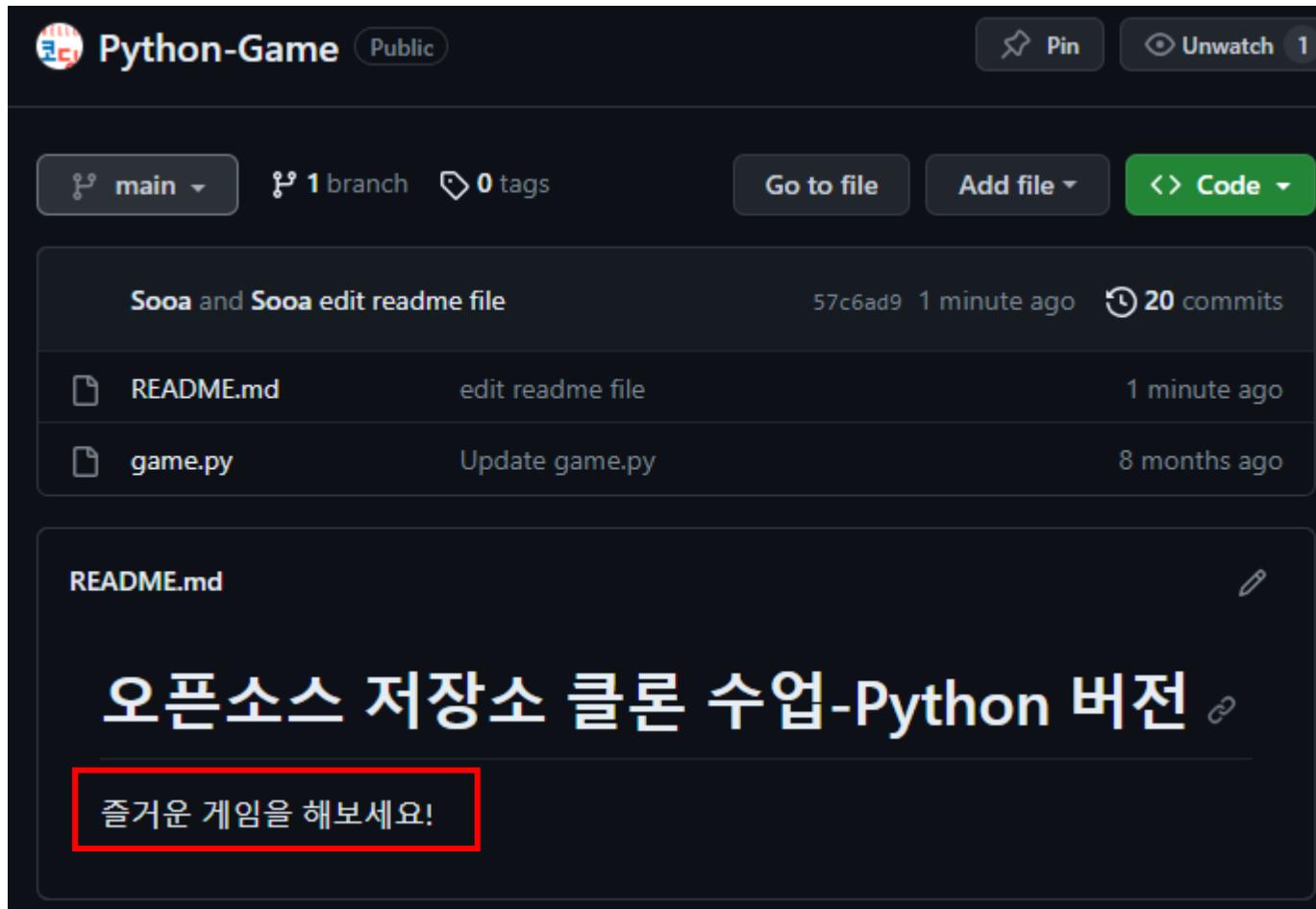
```
$ git commit -m "edit readme file"
```

❖ 원격 저장소에 push하기

```
$ git push
```

원격 저장소에 push하기

❖ 원격 저장소 확인하기



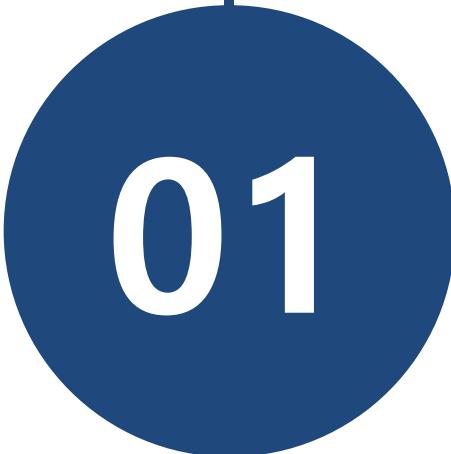
Git & GitHub

- ◆ 팀 프로젝트를 위한 GitHub 설정 - Python

정수아

Contents

01 팀 프로젝트를 위한 GitHub 설정



01

팀 프로젝트를 위한 GitHub 설정

[팀장] 저장소 이름 설정

Create a new repository

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

Owner *  HelloCoding22

Repository name * 

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

Description (optional)
팀 프로젝트를 위한 깃허브 설정 수업

 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.](#)

.gitignore template: [None ▾](#)

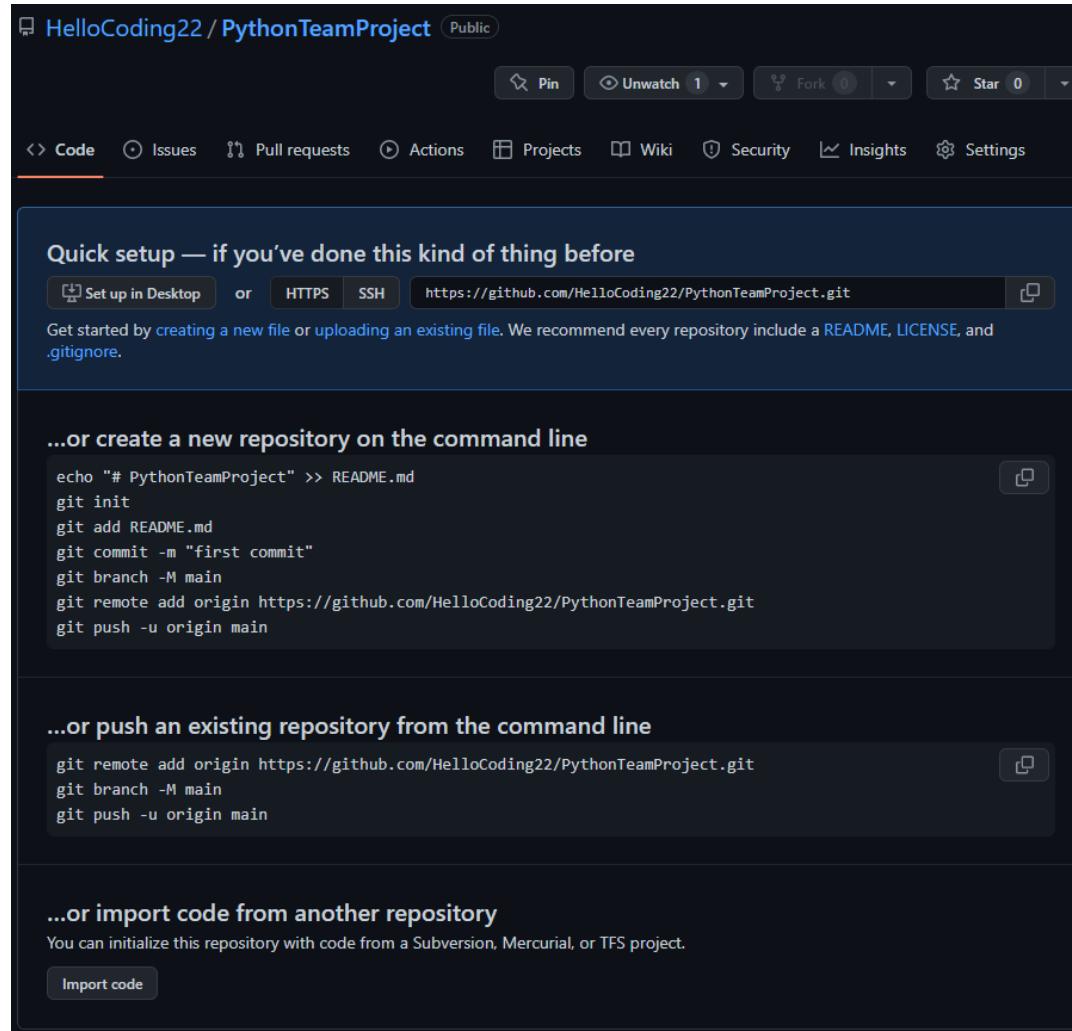
Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: [None ▾](#)

 You are creating a public repository in your personal account.

[Create repository](#)

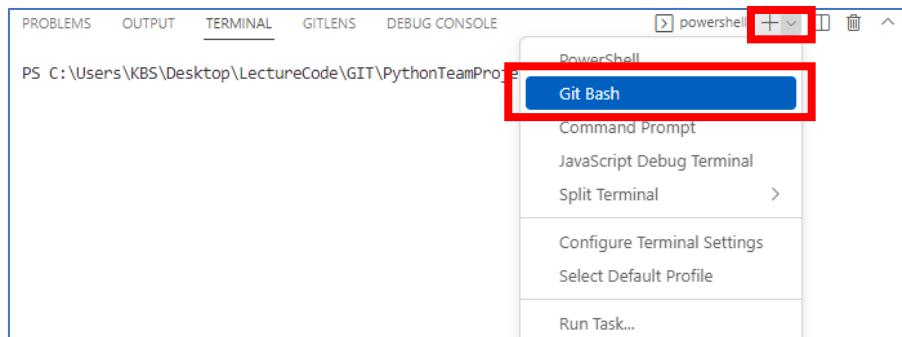
[팀장] 저장소 생성 완료



[팀장] 프로젝트 초기 설정하기

❖ 프로젝트 생성 후 작업 시작

- 터미널에서 Git Bash 실행



- 프로젝트 폴더 초기화

```
$ git init
```

- 예) hello.py

```
print("Hello!")
```

[팀장] 프로젝트 초기 설정하기

❖ 스테이지에 올리기

```
$ git add .
```

❖ 커밋하기

```
$ git commit -m "first commit"
```

❖ 브랜치 변경하기

```
$ git branch -M main
```

[팀장] 프로젝트 초기 설정하기

- ❖ 지역저장소와 원격저장소를 연결하기

```
$ git remote add origin 원격_저장소_주소
```

- ❖ 원격저장소에 push 하기

```
$ git push -u origin main
```

[팀장] GitHub 저장소에 push 완료

The screenshot shows a GitHub repository page for 'HelloCoding22 / PythonTeamProject' (Public). The main navigation bar includes 'Pin', 'Unwatch 1', 'Fork 0', and 'Star 0'. Below the bar are links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. A dropdown menu for the 'main' branch is open, showing options like 'Go to file', 'Add file', and 'Code'. A red box highlights the commit list, which contains two entries: 'HelloCoding22 first commit' (1 minute ago) and 'hello.py first commit' (1 minute ago). To the right of the commit list is an 'About' section with statistics: '팀 프로젝트를 위한 깃허브 설정 수업' (Team project setup on GitHub), '0 stars', '1 watching', and '0 forks'. At the bottom left, there's a call to action to 'Add a README', and at the bottom right, a 'Releases' section indicating 'No releases published' and a link to 'Create a new release'.

>HelloCoding22 / PythonTeamProject (Public)

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main Go to file Add file Code About

HelloCoding22 first commit 1 minute ago 1

hello.py first commit 1 minute ago

Help people interested in this repository understand your project by adding a README. Add a README

Releases

No releases published Create a new release

[팀장] Branch 보호 규칙 설정

The screenshot shows the GitHub repository settings page for 'HelloCoding22 / PythonTeamProject'. The 'Settings' tab is selected. On the left, the 'Branches' section is highlighted with a red box and an arrow pointing from the main title. The 'Add classic branch protection rule' button is also highlighted with a red box and an arrow pointing to it from below.

Classic branch protections have not been configured
Define branch rules to disable force pushing, prevent branches from being deleted, or require pull requests before merging. Learn more about [repository rules](#) and [protected branches](#).

Add branch ruleset Add classic branch protection rule

[팀장] Branch 보호 규칙 설정

Branch name pattern *

main 브랜치 이름

Protect matching branches

보호 규칙

Require a pull request before merging
When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

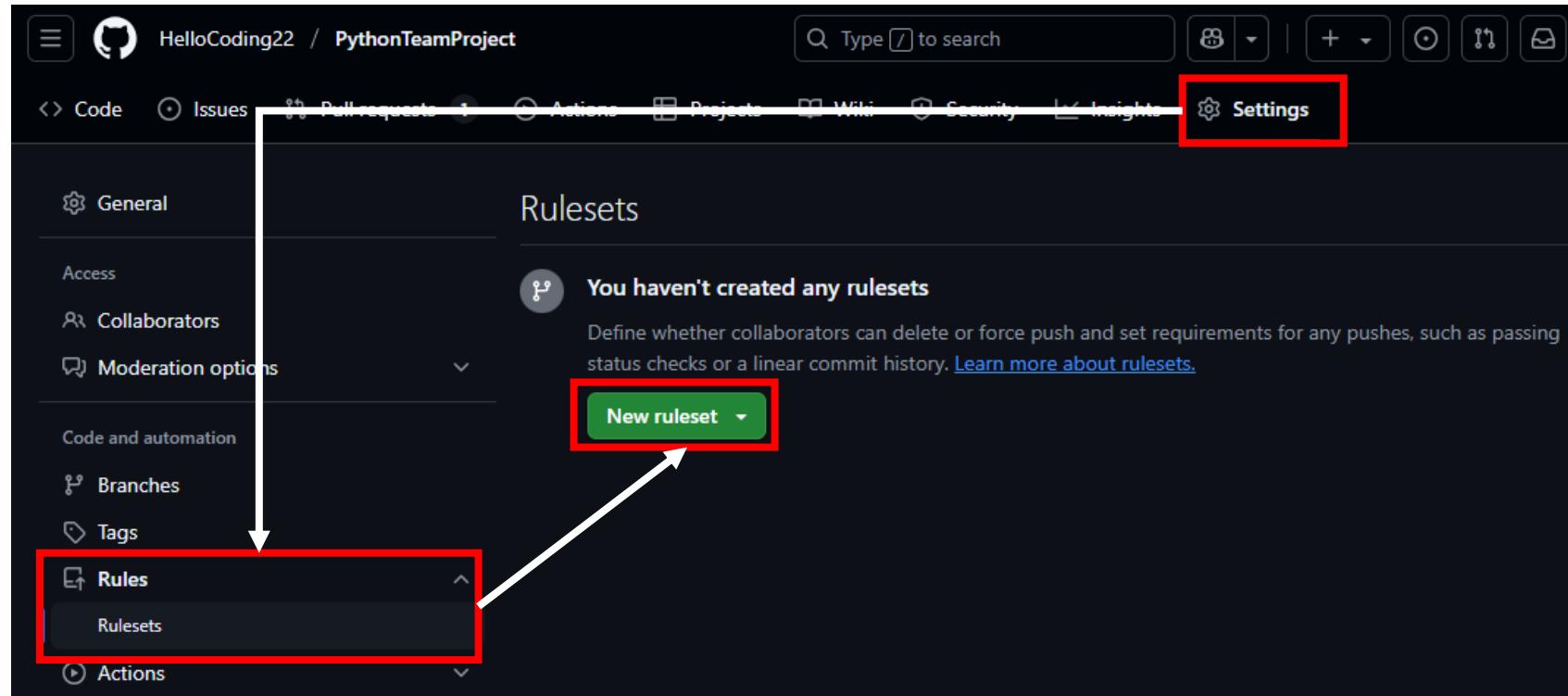
Require approvals
When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.
Required number of approvals before merging: 1 ▾

Dismiss stale pull request approvals when new commits are pushed
New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

Require review from Code Owners
Require an approved review in pull requests including files with a designated code owner.

Require approval of the most recent push
The most recent push must be approved by someone other than the last pusher.

[팀장] Ruleset 규칙 설정



[팀장] Ruleset 규칙 설정

Ruleset Name *

Enforcement status

Disabled

Bypass list

+ Add bypass

Exempt roles, teams, and apps from this ruleset by adding them to the bypass list.

Bypass list is empty

Targets

Which branches do you want to make a ruleset for?

Target branches

Branch targeting determines which branches will be protected by this ruleset. Use inclusion patterns to expand the list of branches under this ruleset. Use exclusion patterns to exclude branches.

Branch targeting criteria

Add target

Branch targeting has not been configured

[팀장] Ruleset 규칙 설정

Rules

Which rules should be applied?

Branch rules

Restrict creations
Only allow users with bypass permission to create matching refs.

Restrict updates
Only allow users with bypass permission to update matching refs.

Restrict deletions
Only allow users with bypass permissions to delete matching refs.

Require linear history
Prevent merge commits from being pushed to matching refs.

Require deployments to succeed
Choose which environments must be successfully deployed to before refs can be pushed into a ref that matches this rule.

Require signed commits
Commits pushed to matching refs must have verified signatures.

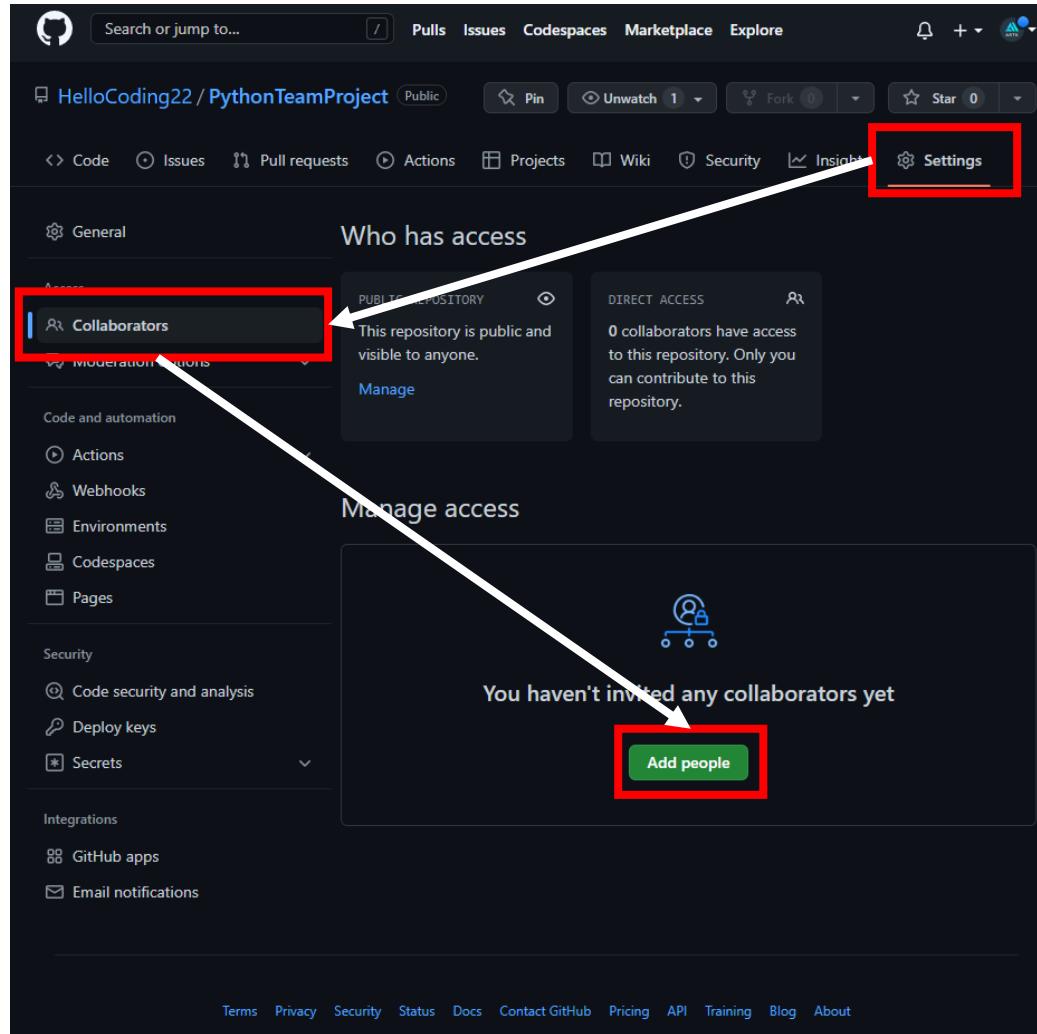
Require a pull request before merging
Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.

Require status checks to pass
Choose which status checks must pass before the ref is updated. When enabled, commits must first be pushed to another ref where the checks pass.

Block force pushes
Prevent users with push access from force pushing to refs.

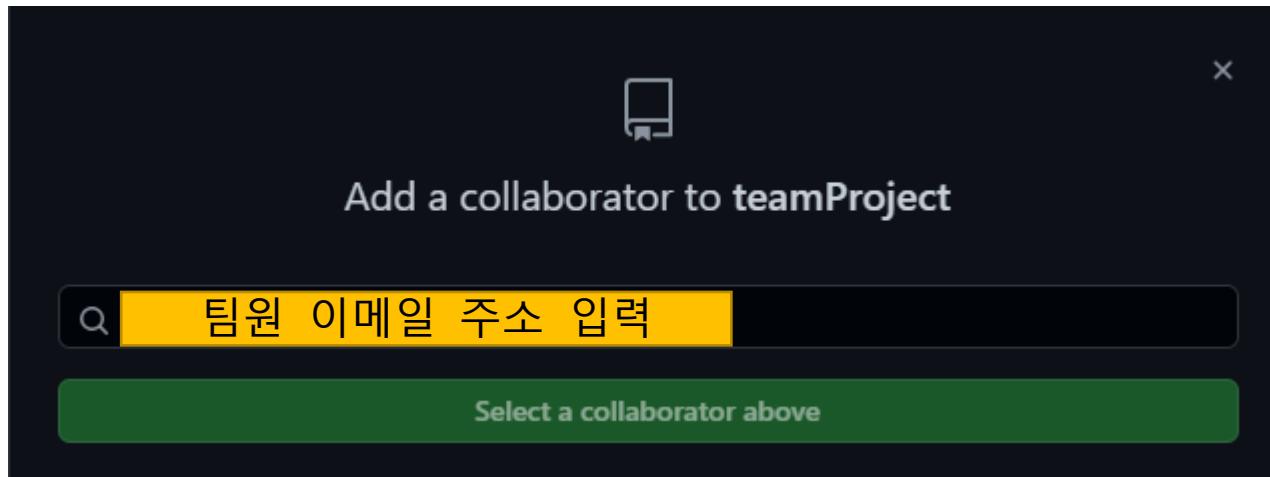
Require code scanning results
Choose which tools must provide code scanning results before the reference is updated. When configured, code scanning must be enabled and have results for both the commit and the reference being updated.

[팀장] 팀원 초대하기



[팀장] 팀원 초대하기

❖ 팀원 초대하기



[팀장] 팀원 초대하기

The screenshot shows the 'Who has access' section of a GitHub repository settings page. On the left sidebar, 'Collaborators' is selected under 'Access'. The main area displays two sections: 'PUBLIC REPOSITORY' (with 1 collaborator) and 'DIRECT ACCESS' (with 0 collaborators). Below this, the 'Manage access' section lists a single user, Sophia-Jung, with the status 'Awaiting Sophia-Jung's response'. A red box highlights this row. At the top right of the 'Manage access' section is a green 'Add people' button.

General

Who has access

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Codespaces

Pages

Security

PUBLIC REPOSITORY

This repository is public and visible to anyone.

DIRECT ACCESS

1 has access to this repository. 0 collaborators. 1 invitation.

Manage

Add people

Type ▾

Manage access

Select all

Find a collaborator...

Sophia-Jung

Awaiting Sophia-Jung's response

Pending Invite

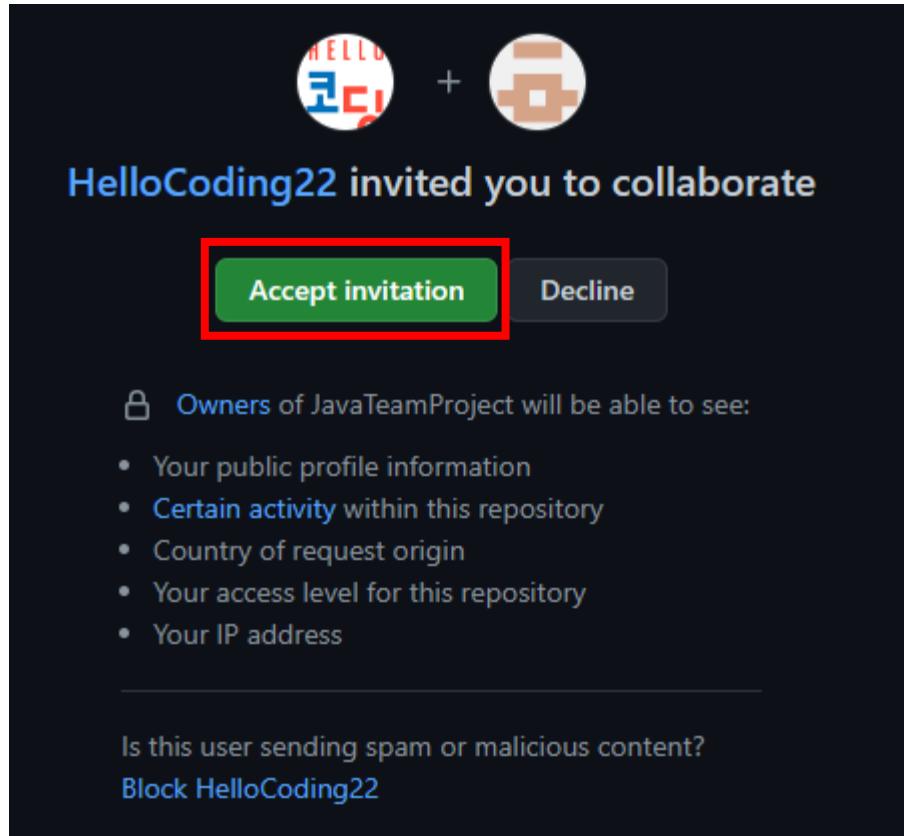
Remove

초대 수락 대기중

[팀원] 프로젝트 참여하기

The screenshot shows the GitHub user profile page for 'Sophia-Jung'. The top navigation bar includes 'Pull requests', 'Issues', 'Codespaces', 'Marketplace', and 'Explore'. The sidebar on the left lists account settings like 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', and access management for 'Billing and plans', 'Emails', 'Password and authentication', 'Sessions', 'SSH and GPG keys', and 'Organizations'. The 'Organizations' tab is currently selected. The main content area displays the 'Organizations' section, featuring a card for 'PythonTeamProject' with a warning icon and the message 'Invitation expires in 7 days.' Below this is the 'Transform account' section with a button to 'Turn Sophia-Jung into an organization'. On the right, a dropdown menu is open, showing options like 'Signed in as Sophia-Jung', 'Set status', 'Your profile', 'Your repositories', 'Your organizations' (which is highlighted with a red box and has a blue dot next to it), 'Your projects', 'Your stars', 'Your gists', 'Your sponsors', 'Upgrade', 'Try Enterprise', 'Feature preview' (with a blue dot), 'Help', 'Settings', and 'Sign out'. A red box also highlights the 'Join' button in the 'PythonTeamProject' card.

[팀원] 프로젝트 참여하기



[팀원] 프로젝트 참여 완료

The screenshot shows a GitHub repository page for 'HelloCoding22 / PythonTeamProject'. A red box highlights the message 'You now have push access to the HelloCoding22/PythonTeamProject repository.' in the top notification bar. The repository is public, with 1 watch, 0 forks, and 0 stars. The 'Code' tab is selected. The main area displays a commit from 'HelloCoding22' and a file named 'hello.py'. On the right, there's an 'About' section in Korean, a 'Releases' section, and a footer note about adding a README.

You now have push access to the HelloCoding22/PythonTeamProject repository.

HelloCoding22 / PythonTeamProject Public

Watch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights

main Go to file Add file Code About

HelloCoding22 first commit ... 7 minutes ago 1

hello.py first commit 7 minutes ago

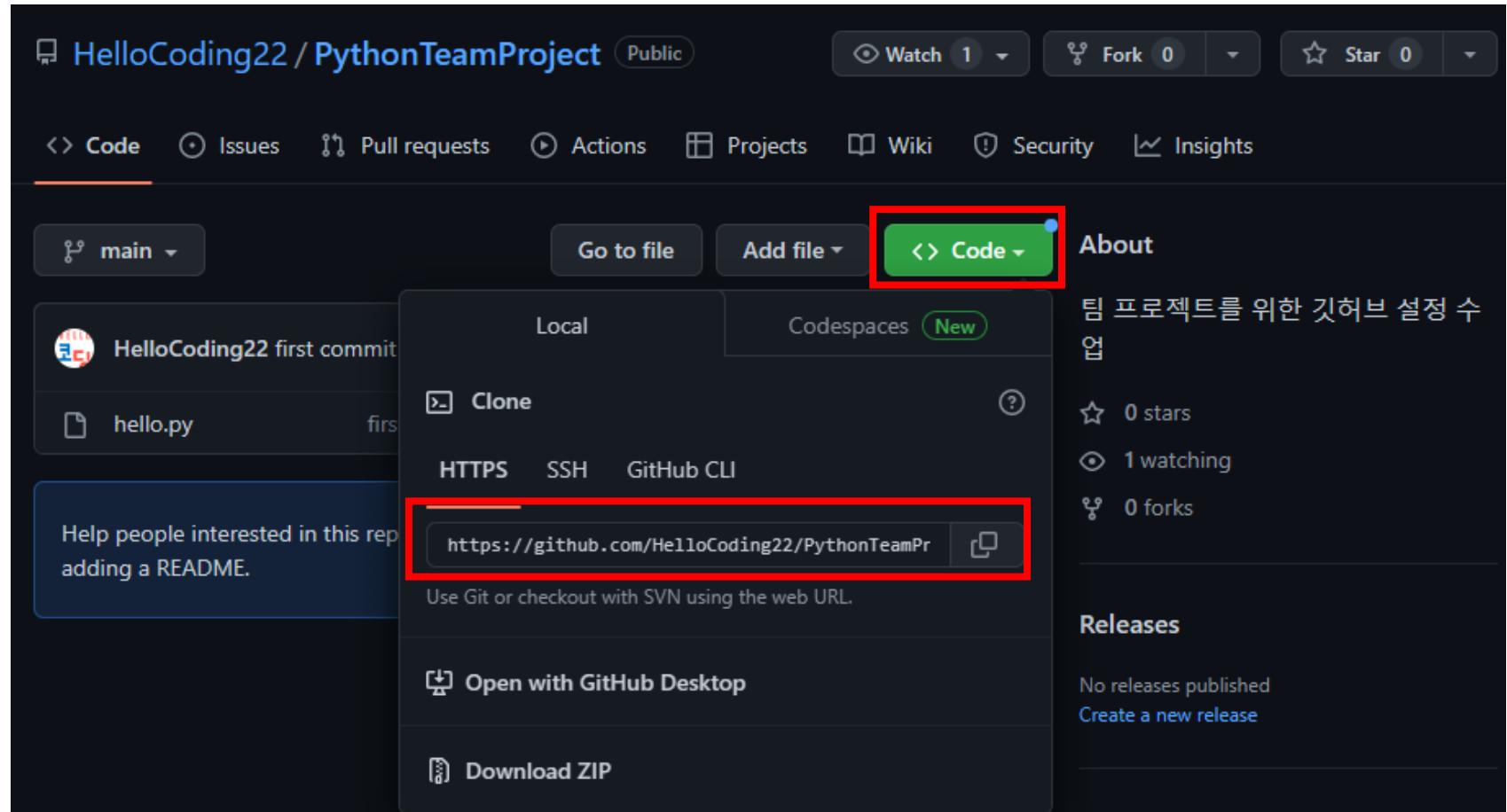
Help people interested in this repository understand your project by adding a README. Add a README

팀 프로젝트를 위한 깃허브 설정 수업

0 stars 1 watching 0 forks

No releases published Create a new release

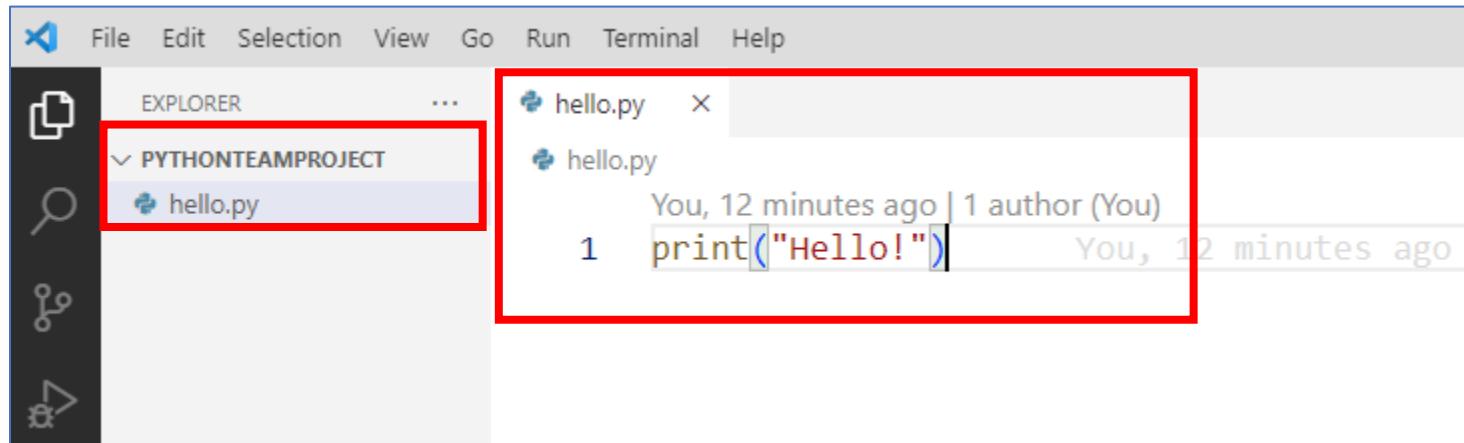
[팀원] GitHub 저장소 복제하기



[팀원] GitHub 저장소 복제하기

❖ GitHub 저장소 복제하기

```
$ git clone https://github.com/팀장계정/프로젝트이름.git
```



[팀원] 작업하기

❖ 팀원 본인 브랜치 생성

- 예) sooa 브랜치 생성

```
$ git branch sooa
```

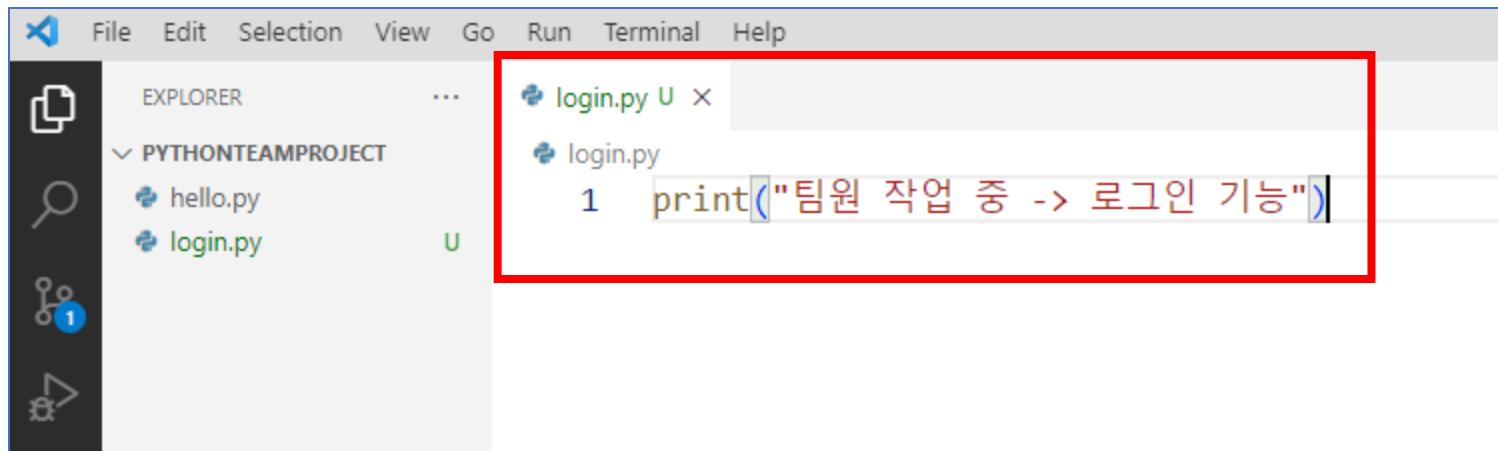
❖ 브랜치 변경(main → sooa)

```
$ git switch sooa
```

[팀원] 작업하기

❖ 코드 작성하기

- 예) login.py



The screenshot shows the Visual Studio Code interface. The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer sidebar shows a project named PYTHONTEAMPROJECT containing files hello.py and login.py. The main editor area displays the content of login.py:

```
1 print("팀원 작업 중 -> 로그인 가능")
```

A red box highlights the line of code: `1 print("팀원 작업 중 -> 로그인 가능")`.

[팀원] 작업하기

❖ 스테이지에 올리기

```
$ git add .
```

❖ 커밋하기

```
$ git commit -m "sooa first commit"
```

[팀원] 작업 완료 파일 업로드

❖ GitHub 저장소에 파일 push

```
$ git push origin 브랜치이름
```

- 예) sooa 브랜치에 파일 업로드

```
$ git push origin sooa
```

❖ GitHub 사이트 → Pull request

[팀원] pull request

This branch is 1 commit ahead of `main`.

| File | Commit Message | Time |
|------------------------|-------------------------------|---------------|
| <code>README.md</code> | Update <code>README.md</code> | 1 hour ago |
| <code>hello.py</code> | sooa first commit | 3 minutes ago |

[팀원] pull request

The screenshot shows the GitHub interface for managing pull requests. At the top, there are several navigation tabs: Code, Issues, Pull requests (which is highlighted with a red box), Actions, Projects, Wiki, Security, and Insights. Below the tabs, a modal window titled "Label issues and pull requests for new contributors" is displayed, with a "Dismiss" button in the top right corner. The main content area shows a search bar with the query "is:pr is:open" and filters for Labels (9) and Milestones (0). A prominent green button labeled "New pull request" is also highlighted with a red box. Below the search bar, it says "0 Open" and "2 Closed". A toolbar at the bottom includes filters for Author, Label, Projects, Milestones, Reviews, Assignee, and Sort. The central message states "There aren't any open pull requests." followed by a link to search all of GitHub or try an advanced search.

Code Issues Pull requests Actions Projects Wiki Security Insights

Label issues and pull requests for new contributors Dismiss

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

Filters is:pr is:open Labels 9 Milestones 0 New pull request

0 Open 2 Closed

Author Label Projects Milestones Reviews Assignee Sort

There aren't any open pull requests.

You could search [all of GitHub](#) or try an [advanced search](#).

[팀원] pull request

The screenshot shows a GitHub interface for comparing changes between two branches. At the top, there are navigation links: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. The 'Pull requests' link is highlighted.

The main title is 'Comparing changes'. Below it, instructions say: 'Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.' A red box highlights the 'compare: sooa' dropdown menu.

A message indicates: '✓ Able to merge. These branches can be automatically merged.' Below this, a text box says: 'Discuss and review the changes in this comparison with others. Learn about pull requests'.

Key statistics shown: -o 1 commit, 1 file changed, 1 contributor.

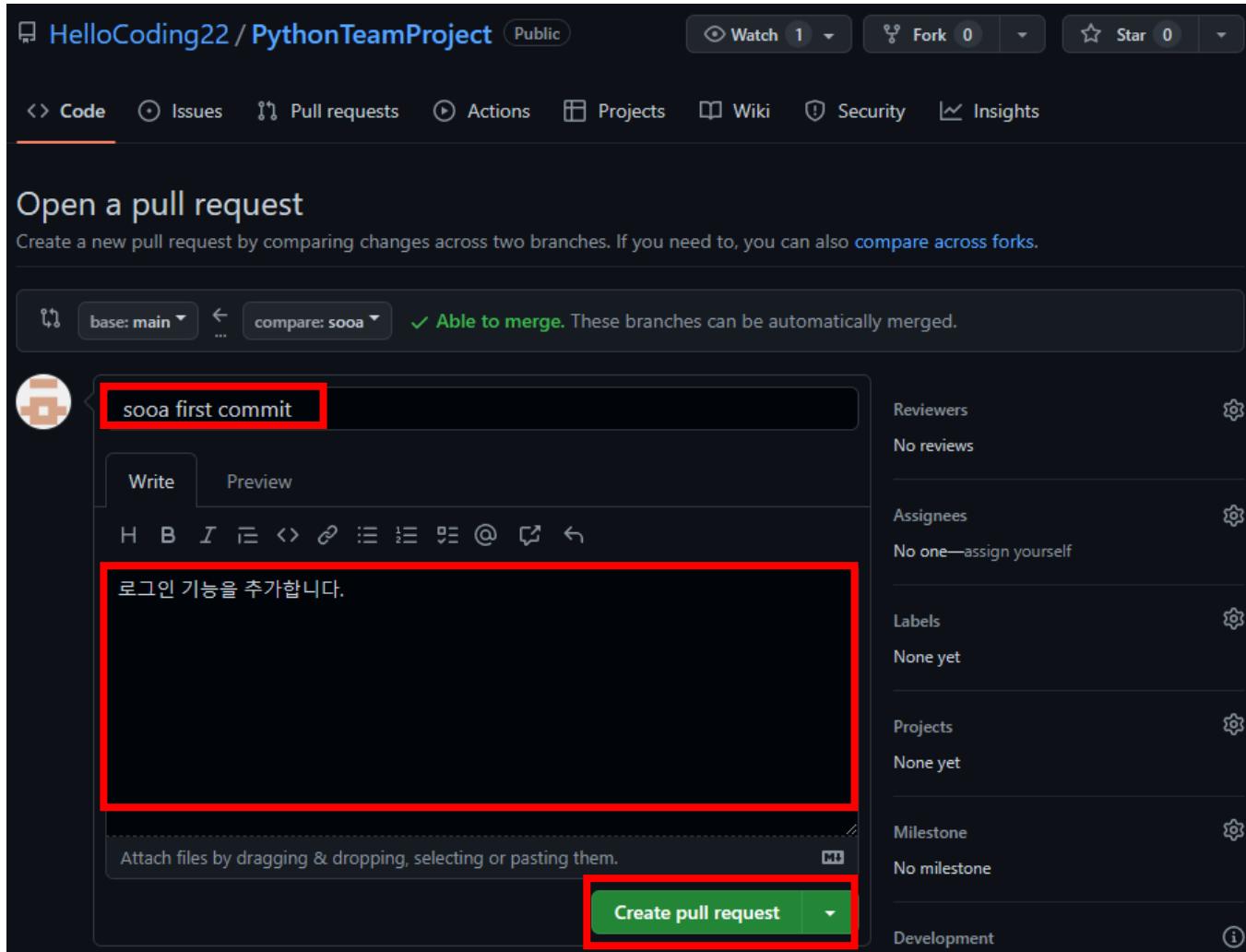
Details for the first commit: 'soo2 first commit' by 'Sooa' (authored and committed 4 minutes ago). A red box highlights the 'Create pull request' button.

The commit details show: 'Showing 1 changed file with 2 additions and 0 deletions.' A code diff is displayed for 'login.py':

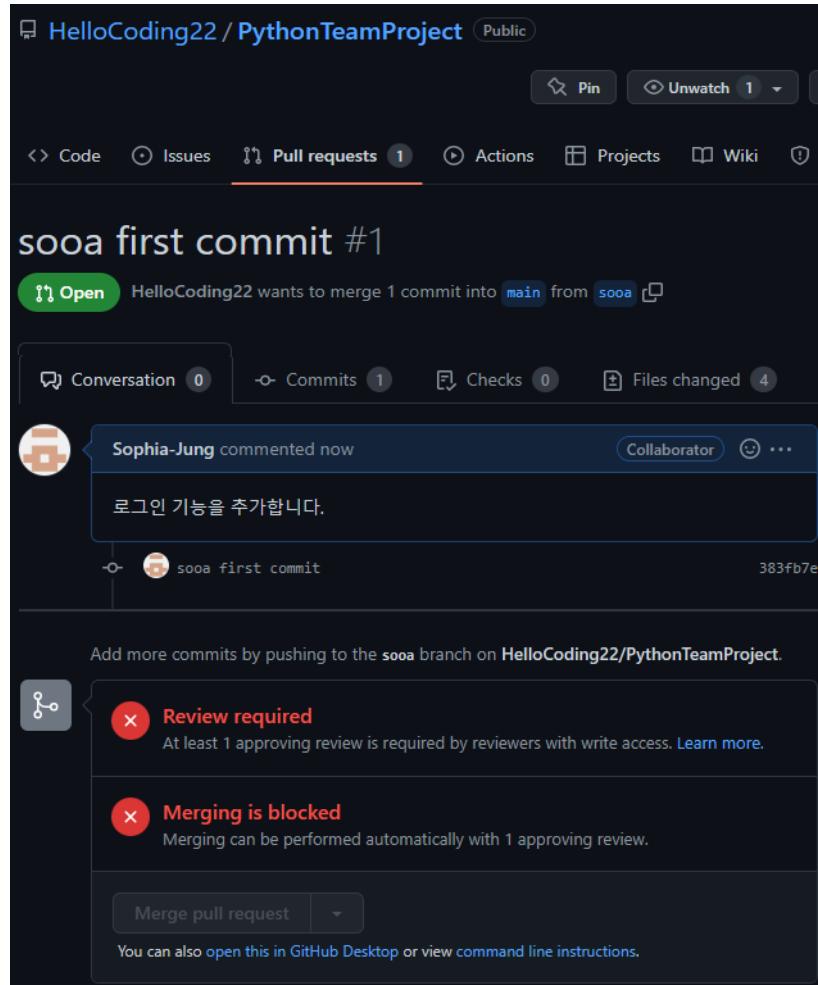
```
diff --git a/login.py b/login.py
index 1f7f3d4..f19e4d4 100644
--- a/login.py
+++ b/login.py
@@ -1,2 +1,4 @@
 1     print("팀원 작업 중 -> 로그인 가능!")
 2     print("팀원 작업 중 -> 회원가입 가능!")
+ 3     +
+ 4     print("팀원 작업 중 -> 회원수정 가능!")
```

At the bottom right, there are 'Split' and 'Unified' buttons for the diff view.

[팀원] pull request



[팀원] pull request



1명 이상의 팀원이
코드 승인을 해줘야
merge 가능

[전체] pull request 확인

The screenshot shows the GitHub interface for the repository `HelloCoding22 / PythonTeamProject`. The repository is public. The navigation bar includes options like Pin, Unwatch, Fork, Star, Code, Issues, Pull requests (which has 1 item), Actions, Projects, Wiki, Security, Insights, and Settings. A red box highlights the 'Pull requests' tab. A modal window titled 'Label issues and pull requests for new contributors' is displayed, stating: 'Now, GitHub will help potential first-time contributors discover issues labeled with [good first issue](#)'. Below the modal are filters for 'Filters', search bar ('is:pr is:open'), 'Labels' (9), 'Milestones' (0), and a green 'New pull request' button. The main area shows 1 open pull request. A red box highlights the first pull request, which is titled 'sooa first commit' and was opened 2 minutes ago by Sophia-Jung.

Dismiss

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with [good first issue](#)

Filters Labels 9 Milestones 0 New pull request

1 Open ✓ 0 Closed

Author Label Projects Milestones Reviews Assignee Sort

sooa first commit
#1 opened 2 minutes ago by Sophia-Jung

[전체] pull request 확인

The screenshot shows a GitHub pull request page for the repository "HelloCoding22 / PythonTeamProject". The pull request is titled "sooa first commit #1" and is marked as "Open". Sophia-Jung wants to merge 1 commit into the "main" branch from the "sooa" branch. The pull request has 0 conversations, 1 commit, 0 checks, and 4 files changed. The commit message is "sooa first commit" followed by the Korean text "코드 확인". A yellow box highlights the commit message. The pull request is in progress, with a review required from a reviewer with write access. Merging is blocked until at least one approving review is received. There is also an option to merge without waiting for requirements to be met (bypass branch protections). The right side of the screen displays settings for reviewers, assignees, labels, projects, milestones, and development.

sooa first commit #1

Open Sophia-Jung wants to merge 1 commit into main from sooa

Conversation 0 Commits 1 Checks 0 Files changed 4

Sophia-Jung commented 1 minute ago

로그인 기능을 추가합니다.

sooa first commit **코드 확인**

Add more commits by pushing to the sooa branch on HelloCoding22/JavaTeamProject.

Review required Add your review
At least 1 approving review is required by reviewers with write access. Learn more.

Merging is blocked
Merging can be performed automatically with 1 approving review.

Merge without waiting for requirements to be met (bypass branch protections)

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

Reviewers
No reviews—at least 1 approving review is required.
Still in progress? Convert to draft

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development

[전체] pull request 확인

The screenshot shows a GitHub pull request page for a repository named "HelloCoding22 / PythonTeamProject". The repository is public. The pull request is titled "sooa first commit #1" and is from the user "sooa" into the "main" branch. There is one commit in the pull request. The "Review changes" button is highlighted with a red box.

sooa first commit #1

Open Sophia-Jung wants to merge 1 commit into main from sooa

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -0

Review changes

sooa first commit

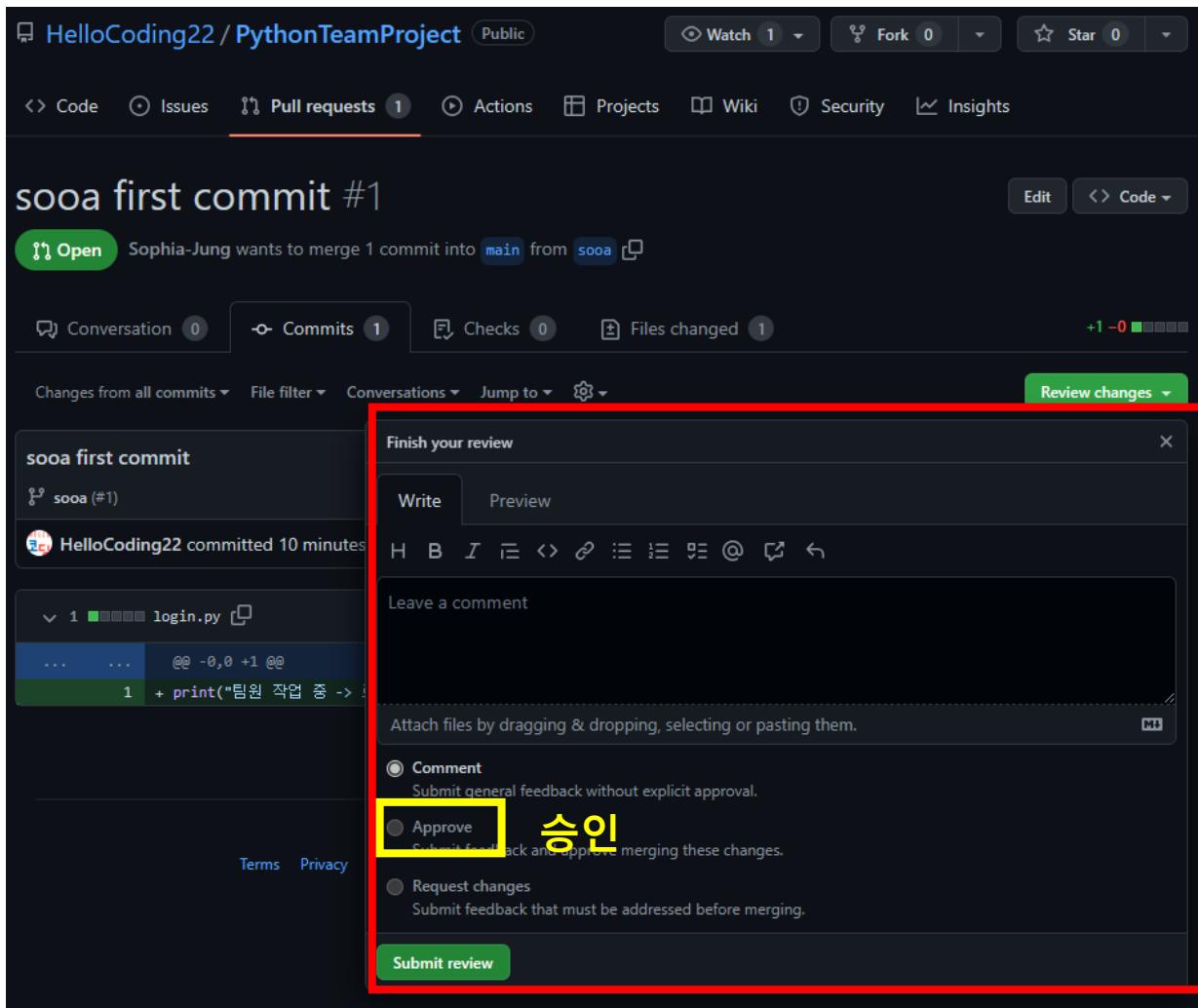
sooa (#1)

Sophia-Jung committed 12 minutes ago commit 896ca87bbb9c0e0b52fbf048b1bfd285aaaf0b4fc

login.py

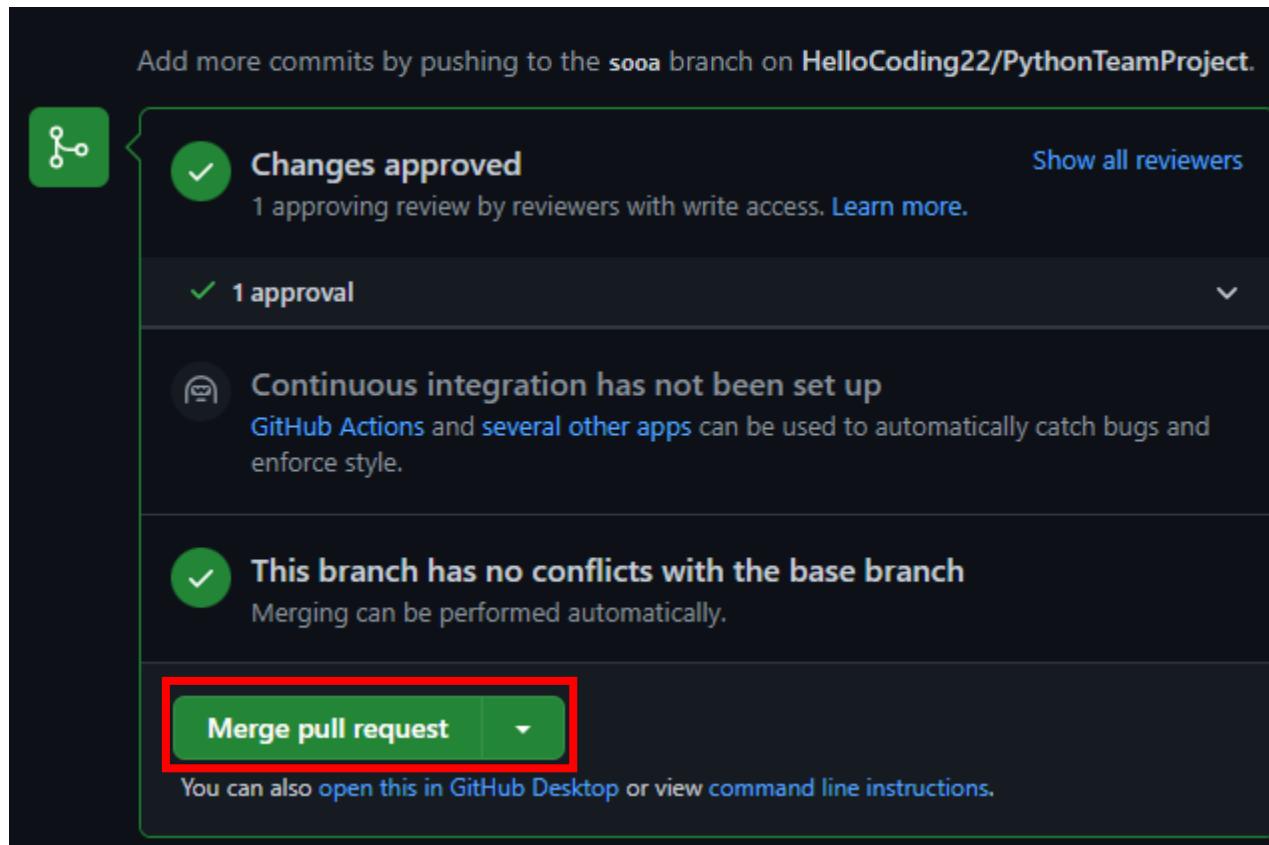
```
@@ -0,0 +1 @@
 1 + print("팀원 작업 중 -> 로그인 가능")
```

[전체] pull request 확인



[전체] pull request 승인 완료

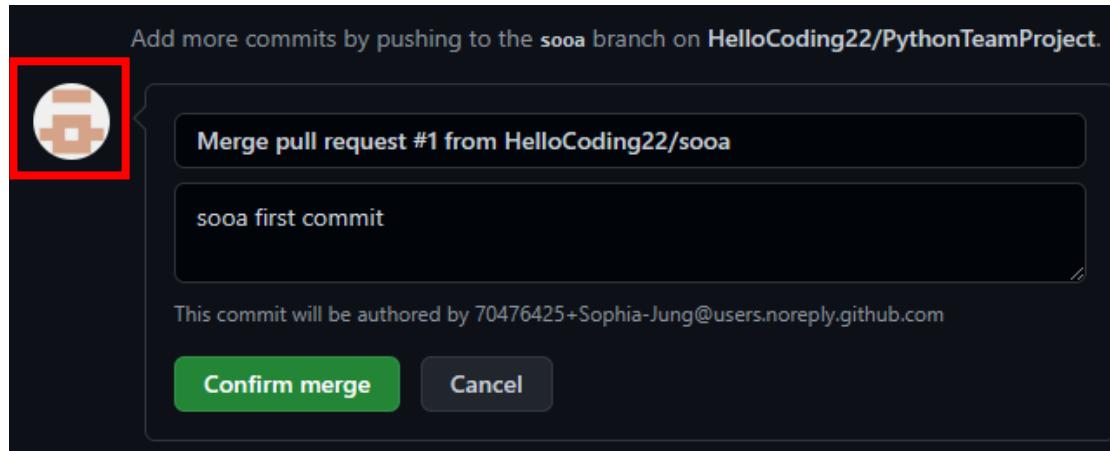
❖ Merge pull request 버튼 활성화



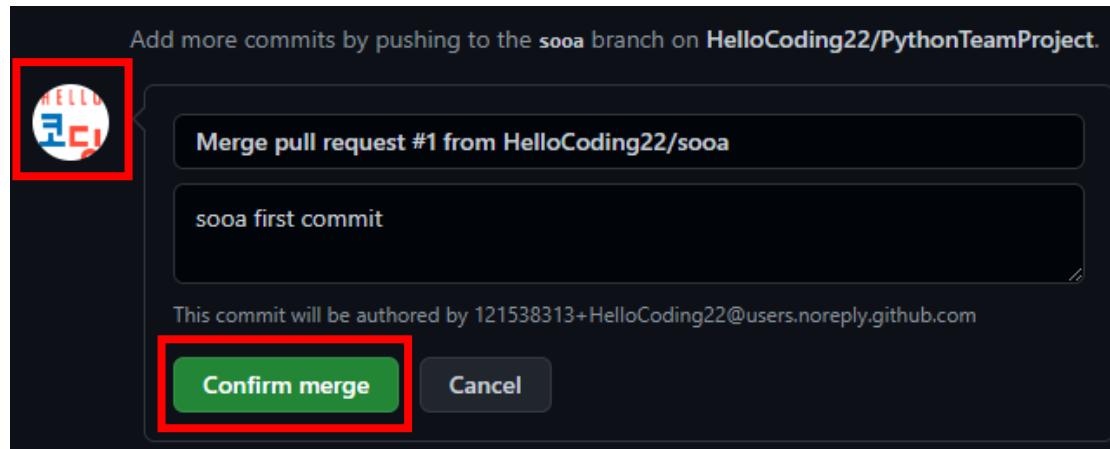
[팀장] merge 버튼 클릭

- ❖ 팀원은 merge 하지 않는 것이 좋음

팀원

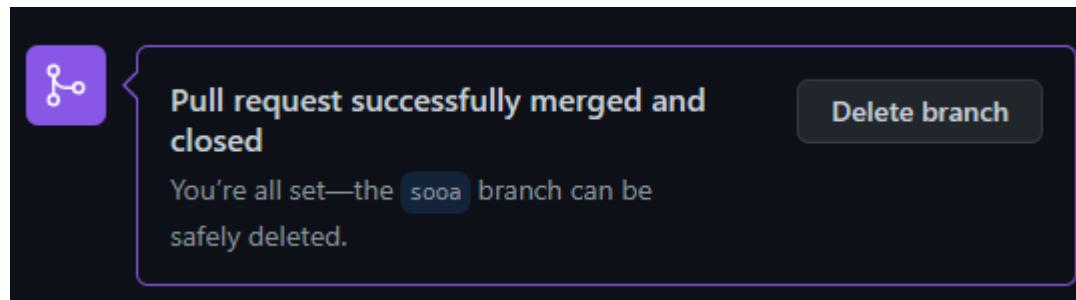


팀장



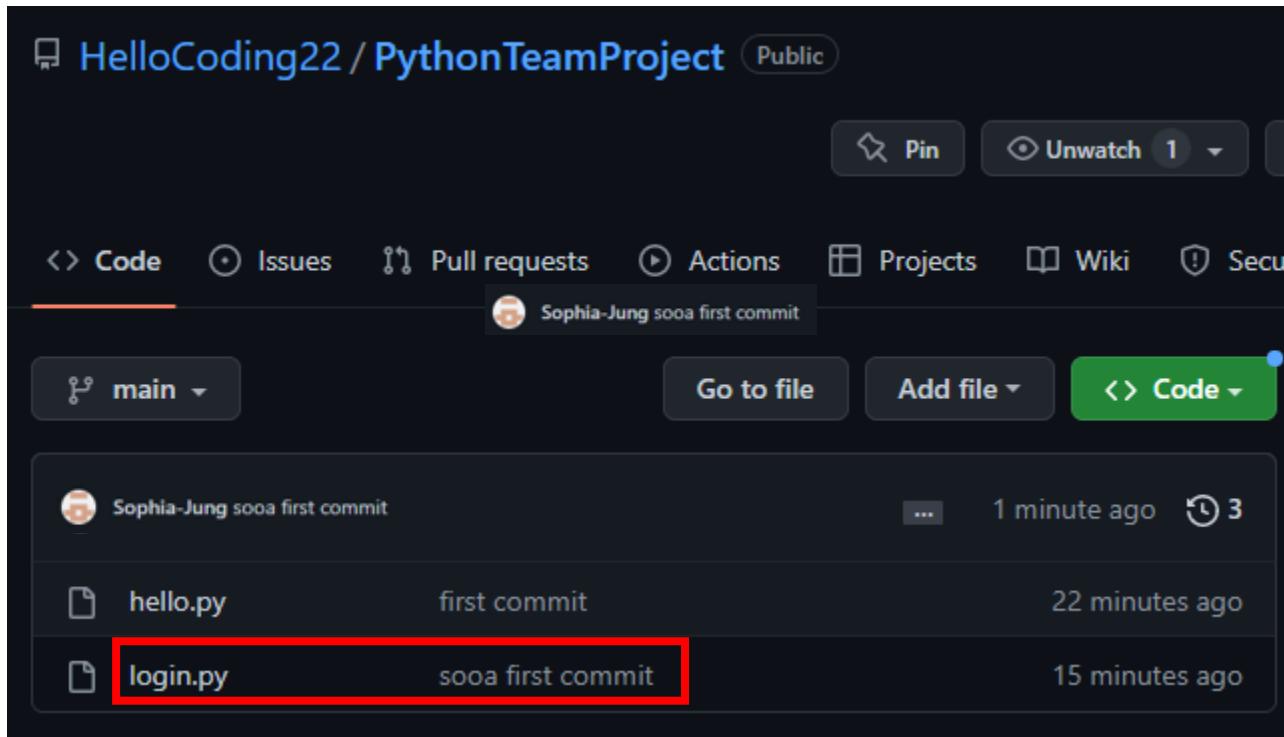
[팀장] merge 버튼 클릭

- ❖ merge 완료



merge 완료

❖ merge 완료



코드 작성 중 main 브랜치 변경 발생

❖ [전체] 기존에 작성하던 코드 저장

- hello.py

```
print("Hello!")
print("팀장 작업 -> 코드 작성 중")
```

코드 작성 중 main 브랜치 변경 발생

- ❖ [전체] 기존에 작성하던 코드 저장 및 스테이징

```
$ git add .
```

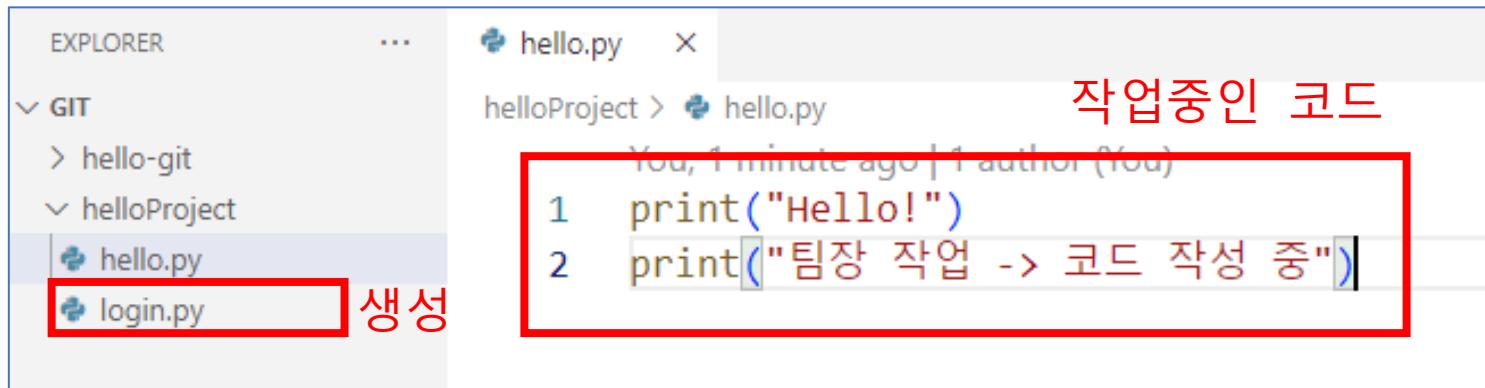
- ❖ [전체] 커밋하기

```
$ git commit -m "second commit"
```

[전체] 최신 프로젝트 pull

- ❖ main 브랜치로부터 최신 파일 다운로드(동기화)

```
$ git pull origin main
```



[전체] 작업 완료 파일 업로드

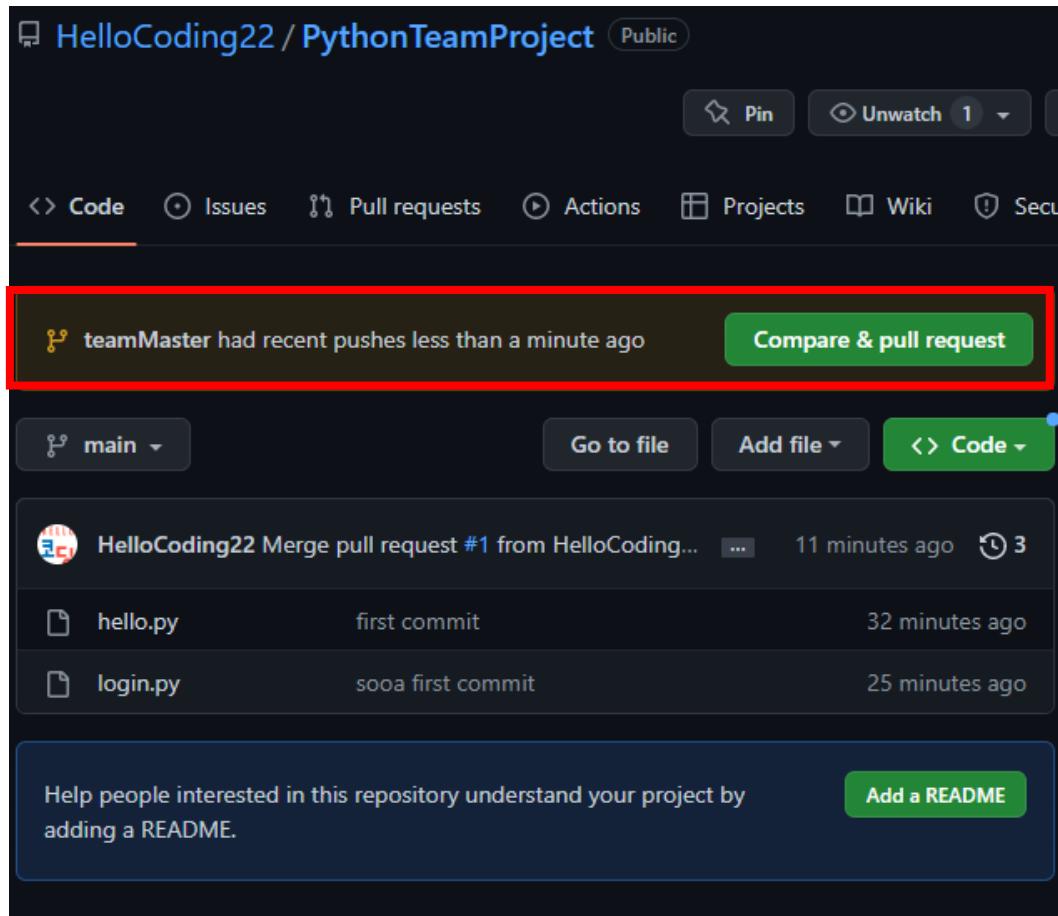
❖ GitHub 저장소에 파일 push

- 예) teamMaster(팀장) 브랜치에 파일 업로드

```
$ git push origin teamMaster
```

[전체] 작업 완료 파일 업로드

❖ Pull request 요청하기



THANKYOU