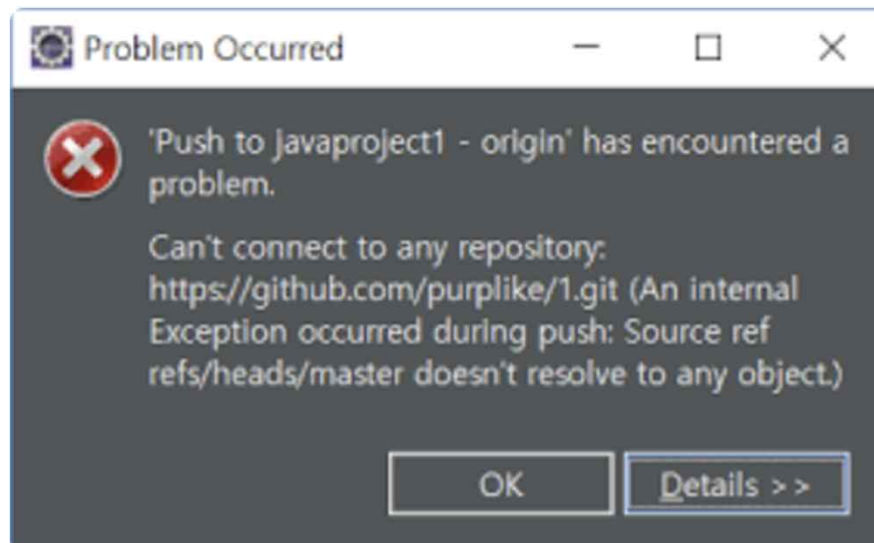


토큰, gitignore, GitHub desktop 등

GitHub 토큰 인증

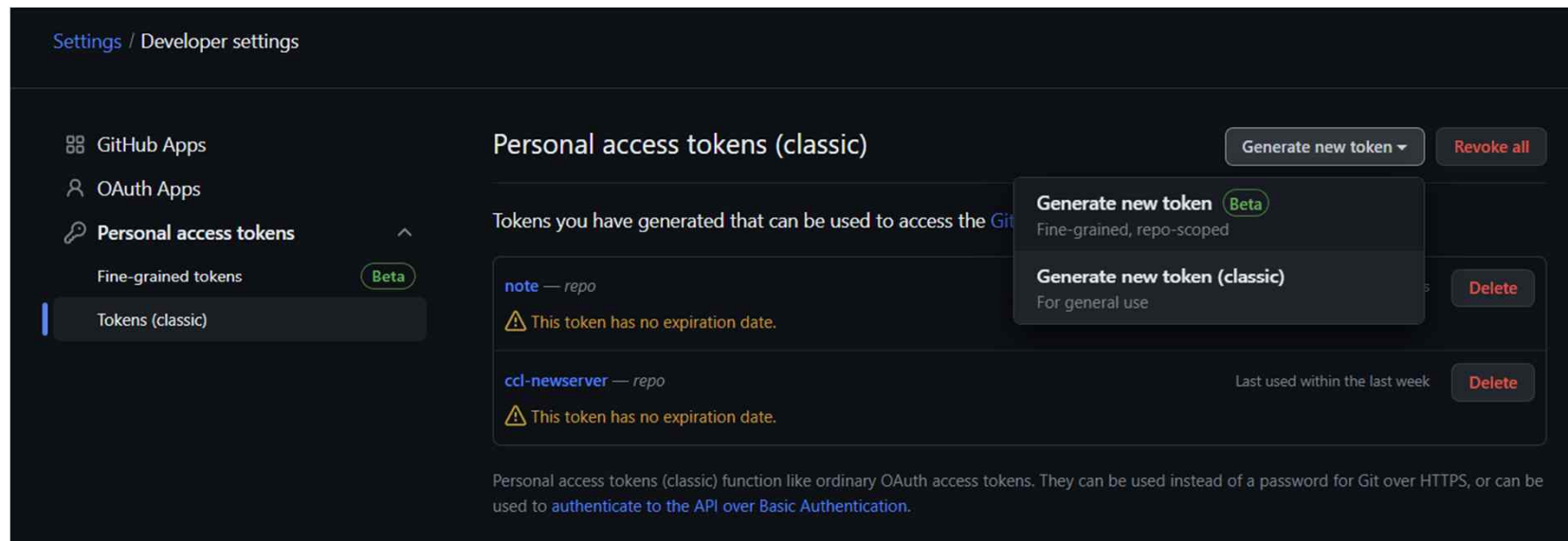
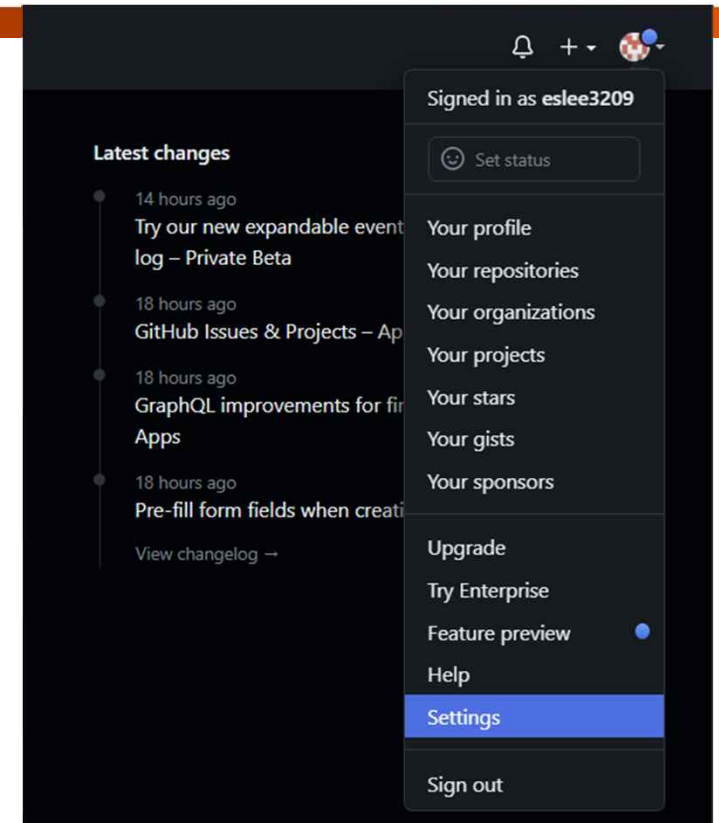
- 최근 GitHub에서는 ID/PW 기반의 basic authentication 인증을 금지하고, ID/Personal Access Token 방식의 Token Authentication 인증을 요구함
- 따라서 소스코드를 push/clone 시 오류가 뜨면서 작동하지 않는 상황 발생 가능 (ex. Linux)



<https://inpa.tistory.com/entry/GitHub-%F0%9F%8F%9B%EF%B8%8F-%EA%B9%83%ED%97%99-%ED%86%A0%ED%81%B0-%EB%B0%9B%EA%B8%B0>

GitHub 토큰 발급

- GitHub 로그인
- Settings > Developer settings > Personal access tokens
- 일단 Tokens (classic) 탭 선택
- Generate new token (classic) 클릭



GitHub 토큰 발급

- 비밀번호 입력
- Expiration date 선택
- repo, gist 체크박스 선택하고 "Generate token" 버튼 클릭

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 *

90 days The token will expire on Thu, Jul 27 2023

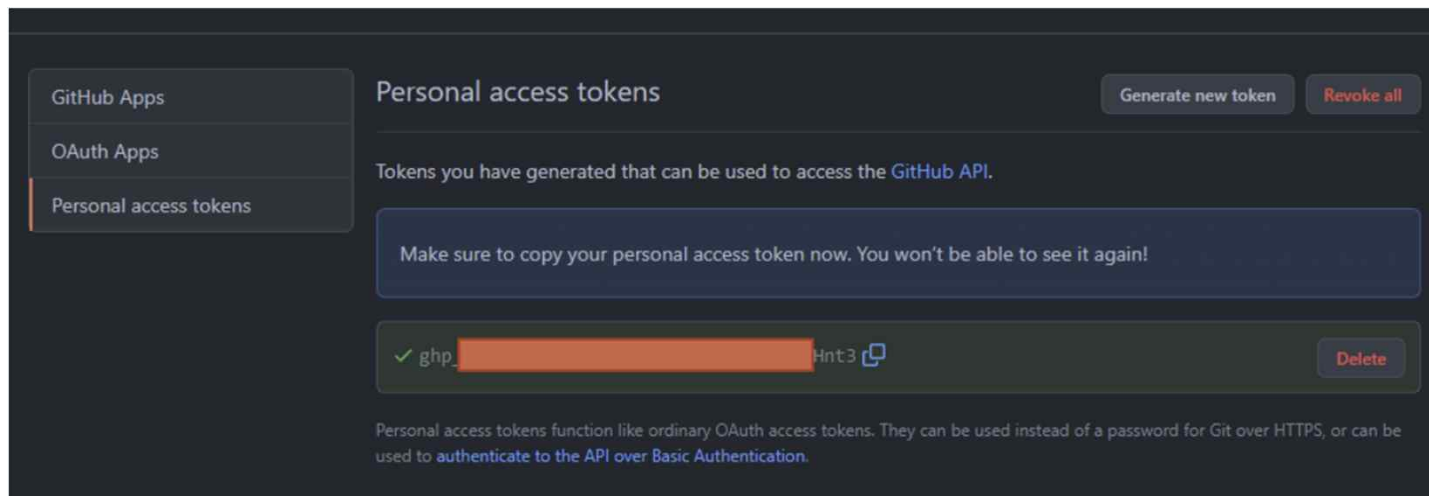
Select scopes

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

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

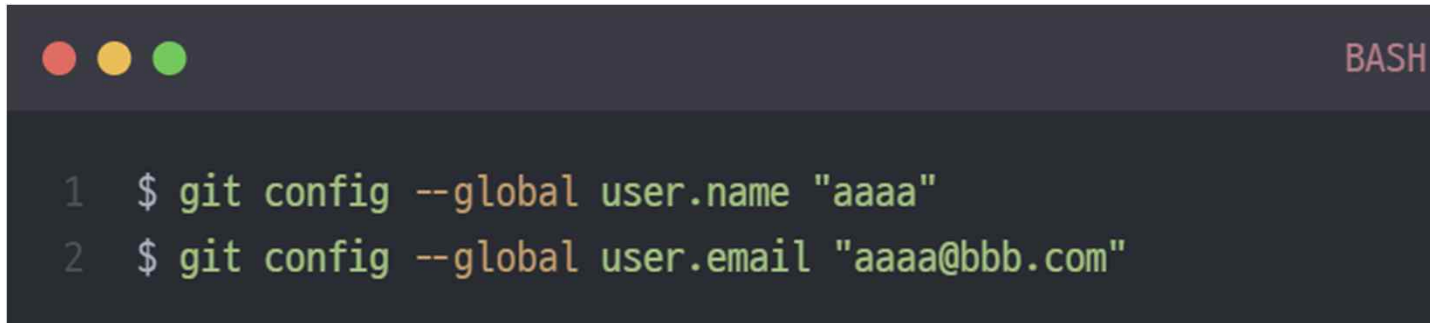
GitHub 토큰 발급

- 그러면 토큰이 발급됨
- 발급된 토큰은 메모장 등 어딘가에 저장해놓아야함. 새로고침하면 토큰을 볼 수 없기 때문
- 만약 토큰을 잃어버렸으면 다시 생성해야함
- 발급 받았으면, 이제 push 등의 작업을 수행할 때, PW 입력하라고 하는 경우 방금 생성한 token을 입력하면 됨



GitHub 토큰 등록

- 아래와 같이 user name과 user email을 설정함

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The title bar on the right says "BASH". The terminal shows two lines of commands:

```
1 $ git config --global user.name "aaaa"  
2 $ git config --global user.email "aaaa@bbb.com"
```

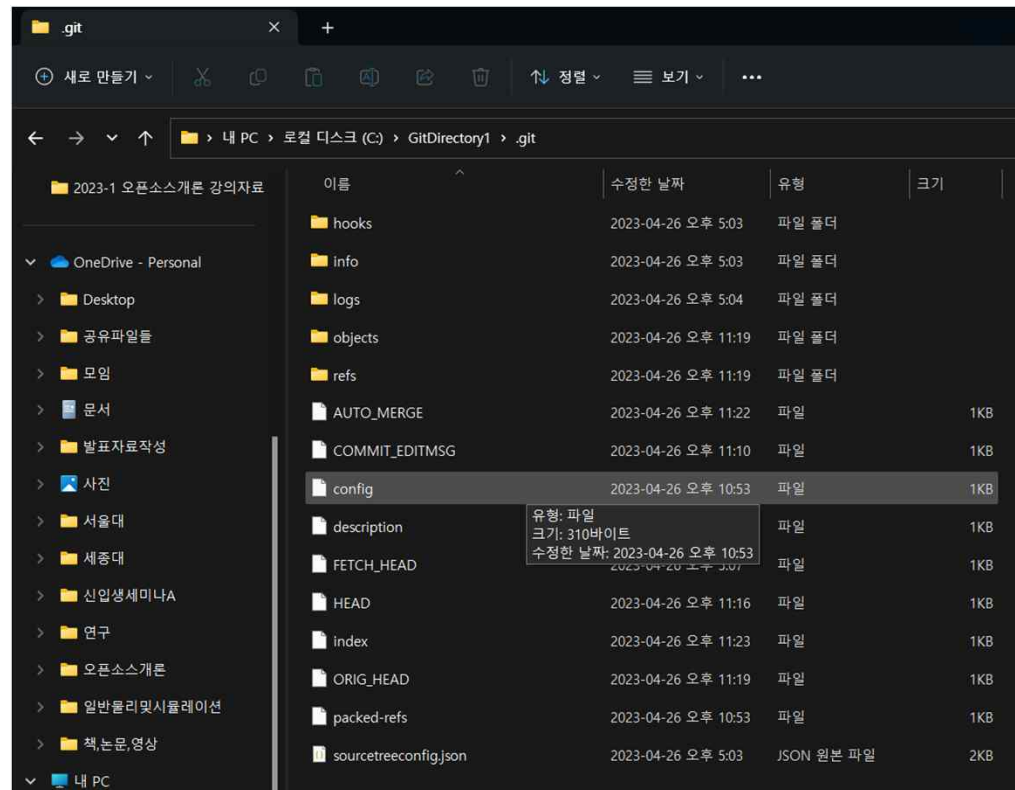
- 그리고 github.com에서 clone, pull 할 때 이메일과 pw를 물어보는데 이제 기존 pw가 아닌 방금 만든 token을 사용하면 됨
- 그렇지만 토큰은 외우기 힘들기 때문에 매번 복사해서 붙여넣기하는 것이 번거로울 수 있음

GitHub 토큰 등록

- 따라서 cache 설정을 이용하여 자신의 토큰을 해당 git repository에 저장할 수 있는데, 아래 명령어를 입력하면 다음번에 넣는 user email과 pw(토큰)을 캐싱하게 됨
 - `git config --global credential.helper cache`
- 한번 위와 같이 캐싱하게 되면 더 이상 패스워드(토큰)을 물어보는 일은 없을 것
- 만일 캐시를 지우고자 하면 다음 명령어를 입력
 - `git config --global --unset credential.helper`

GitHub 저장소에 토큰 영구 적용

- 만든 토큰을 기존에 있던 git repository에 아예 저장하여 영구 적용하고자 함
- 먼저 본인의 로컬 저장소 폴더에 들어감. 그러면 .git이라는 폴더가 있음
- .git에 들어가보면 config 파일이 있음



GitHub 저장소에 토큰 영구 적용

- 이 config 파일을 편집기로 오픈함

```
C: > GitDirectory1 > .git > ⚙ config
1  [core]
2      repositoryformatversion = 0
3      filemode = false
4      bare = false
5      logallrefupdates = true
6      symlinks = false
7      ignorecase = true
8  [remote "origin"]
9      url = https://github.com/eslee3209/GitRepository1.git
10     fetch = +refs/heads/*:refs/remotes/origin/*
11  [branch "master"]
12     remote = origin
13     merge = refs/heads/master
14
```

- 필요한 것은 [remote "origin"] 부분에 있는 url 부분임
- url을 다음과 같은 형식으로 바꾸면 됨
 - url = <https://<user-id>:<token>@github.com/eslee3209/GitRepository1.git>
- 그러면 git push 명령어 사용 시 github에 push가 잘 될 것임

.gitignore

- .gitignore 파일
 - 프로젝트에서 원하지 않는 백업 파일이나 로그파일, 컴파일된 파일 등을 git 버전관리에서 제외시키도록 하는 설정 파일
 - 파일을 제외시킴으로서 업로드 용량을 줄일 수 있음
 - 또한, 보안이 중요한 파일이 잘못 업로드 되지 않도록 할 수 있음
- .gitignore 사용법
 - 단순히 로컬저장소에 .gitignore 파일을 만들어놓기만 하면 이후 스테이징할 때 알아서 무시해야할 파일은 스테이징에서 제외함

.gitignore

- .gitignore 자동생성 사이트
 - <https://www.toptal.com/developers/gitignore>
 - 검색창에 운영체제, 개발환경(IDE), 프로그래밍 언어를 검색하면 .gitignore 파일을 자동으로 생성함
 - github에 생략되어할 파일들이 운영체제, IDE, 프로그래밍언어별로 생성이 되며, 그대로 .gitignore 파일에 복사붙여넣기 하면 됨

.gitignore 작성 규칙

- 커밋 대상에서 제외시킬 파일들
 - (1) IDE 관련 설정파일
 - (2) 언어의 빌드 결과물, 로그, 패키지 관련 파일
 - (3) 그 외 프로젝트에서 사용자가 제외하기 원하는 파일 등

<https://github.com/github/gitignore>

.gitignore 파일 규칙

- #, 빈라인
 - #은 주석을 의미하며, 빈라인은 아무런 영향을 주지 않음
- file_name
 - 특정 파일(file_name) 무시
- /file_name
 - 현재 폴더에서 특정 파일(file_name) 무시
- folder_name/file_name
 - 특정 경로의 특정 파일 제외
- *.ext
 - 확장자가 .ext인 모든 파일을 무시
- /*.ext
 - 현재 폴더에서 확장자가 .ext인 모든 파일을 무시

.gitignore 파일 규칙

- folder_name/
 - 해당 폴더의 모든 파일을 무시
- folder_name/*.ext
 - 해당 폴더의 확장자가 .ext인 모든 파일을 무시
- folder_name/**/*.ext
 - 해당 폴더 포함한 하위 모든 폴더에서 확장자가 .ext인 모든 파일을 무시
- !file_name
 - 예외 만들기
 - 특정 파일(file_name)은 무시 대상에서 제외

/: 현재 폴더 내에서만의 영역 의미

/**/*.: 현재 폴더 포함한 모든 하위 폴더 영역 의미

.gitignore 파일 예제

```
# 모든 확장자 .a 파일을 무시
*.a

# 무시하는 모든 확장자 .a 파일들 중에서 lib.a 파일은 무시하지 않음
!lib.a

# Project/
#   └ .gitignore
#   └ A/
#       └ a.txt
#       └ TODO/
#           └ tt.txt
#       └ TODO/
#           └ t.txt

# 현재 폴더 중에서 TODO 폴더에 있는 모든 파일을 무시
# (즉, t.txt 파일만 무시되고 tt.txt 파일은 무시되지 않음)
/TODO
```

.gitignore 파일 예제

```
# 프로젝트 전체 폴더 중 TODO라는 폴더명을 사용하는 TODO 폴더의 하위 파일은 모두 무시
# (즉, t.txt 파일과 tt.txt 파일 모두 무시됨)
TODO/

# Project/
#   └ .gitignore
#   └ doc/
#       └ d.txt
#       └ p.pdf
#       └ server/
#           └ ss.txt
#           └ pp.pdf

# 현재 폴더 중에서 doc 폴더 바로 밑에 있는 .txt 확장자 파일만 모두 무시
# 단, doc/server/ss.txt 와 같은 형식에서는 .txt 확장자 파일이 무시되지 않음
doc/*.txt

# 현재 폴더 중에서 doc 폴더 하위에 있는 .pdf 확장자 파일은
# doc 폴더 하위 어떤 폴더에 들어 있더라도 모두 무시
# (즉, p.pdf 파일과 pp.pdf 파일 모두 무시됨)
doc/**/*.pdf
```


GitHub Desktop

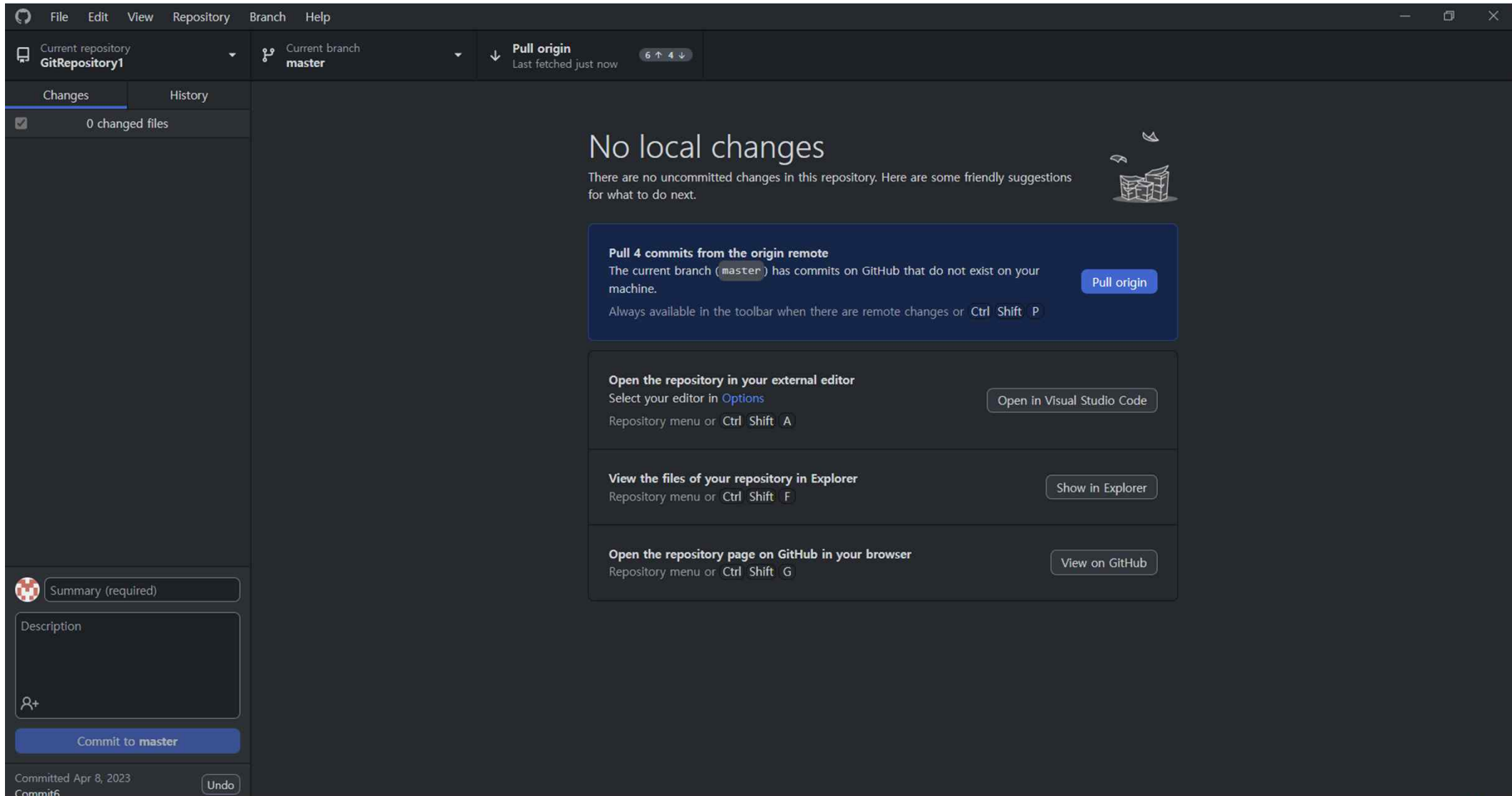
- GitHub desktop이란?
 - GitHub가 직접 개발한 공식 git 프로그램
 - Windows에서도 git 소스를 GUI로 편리하게 관리할 수 있는 툴
 - pull, push, commit 등을 도와줌

GitHub Desktop

- 설치
 - <https://desktop.github.com/>
 - 위 사이트에서 다운로드하여 설치 진행

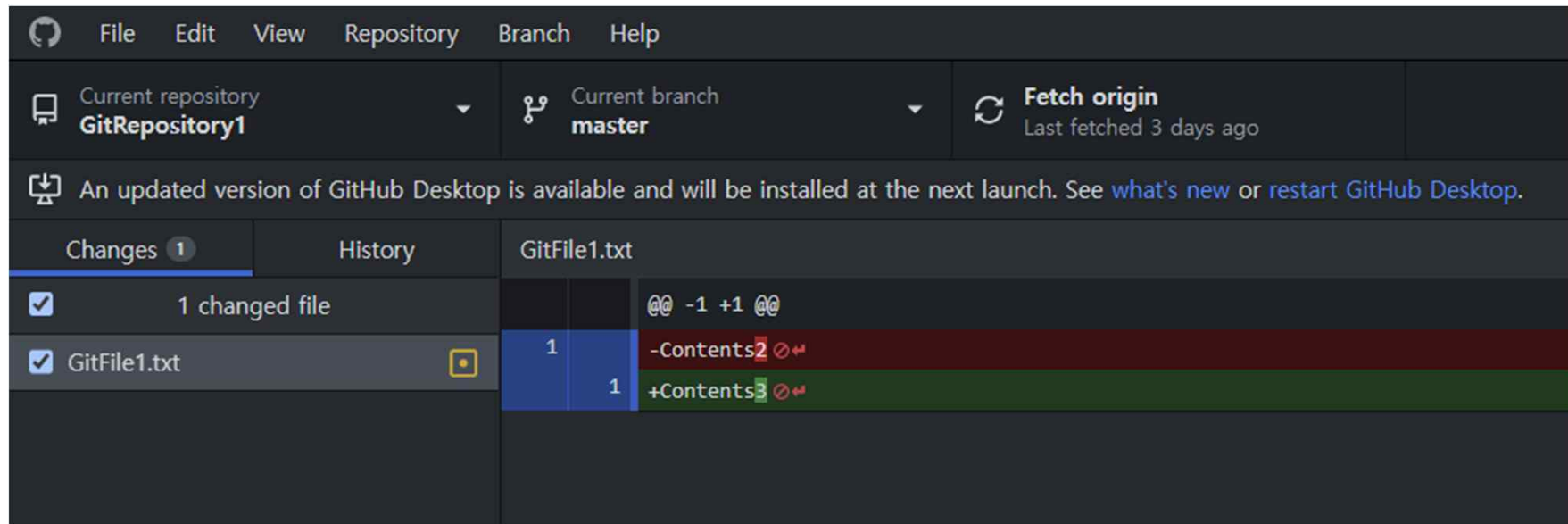
GitHub Desktop

- 메인화면



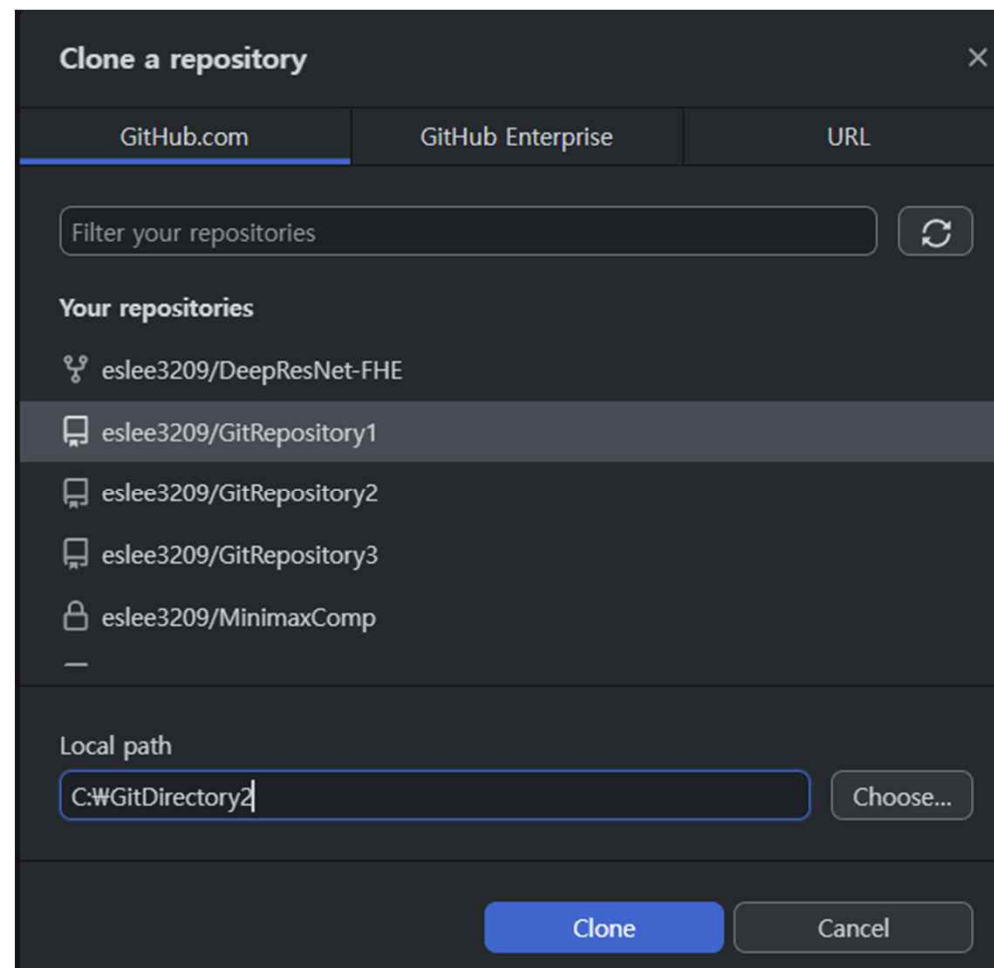
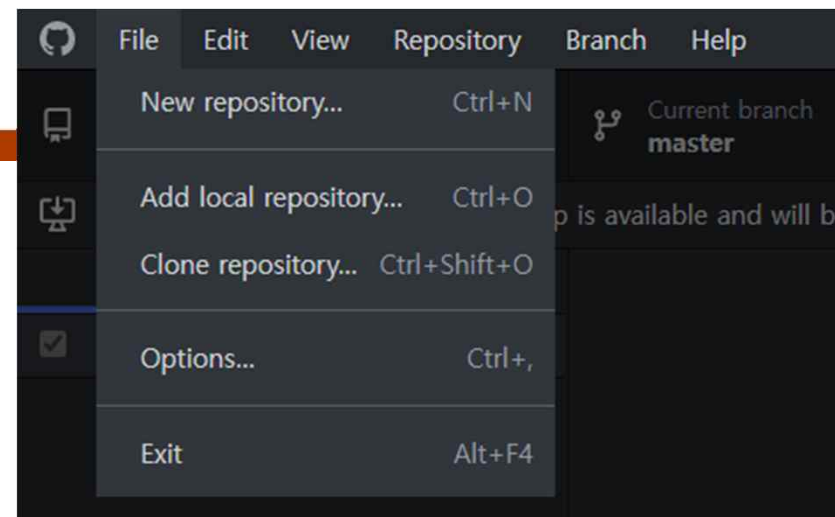
GitHub Desktop

- 수정사항
 - 왼쪽 Changes 탭을 보면 수정사항이 뜸
 - 아직 커밋하지는 않은 변경사항들
 - 체크박스에 체크를 한 것이 마치 스테이징을 한 것에 해당된다
생각할 수 있음



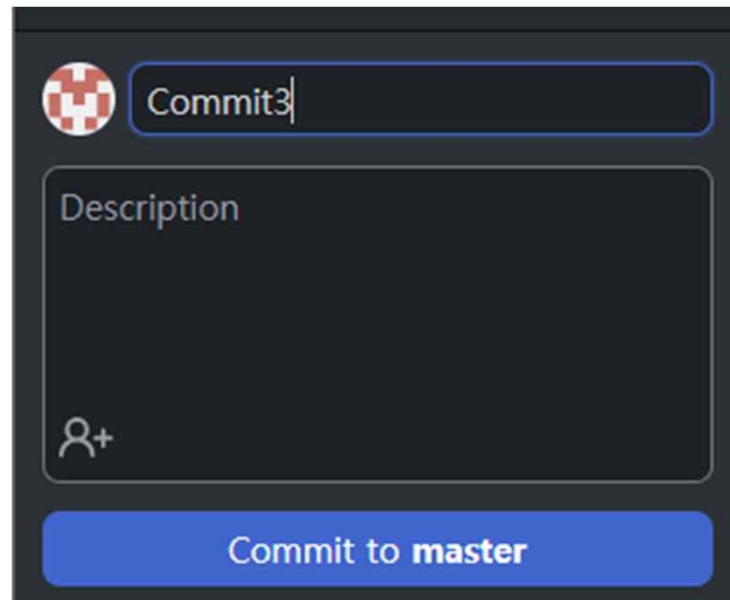
GitHub Desktop

- Clone
 - File>Clone repository... 선택
 - 클론할 원격저장소와 local folder(빈 폴더)를 선택해주고 [Clone] 버튼을 클릭하면 됨



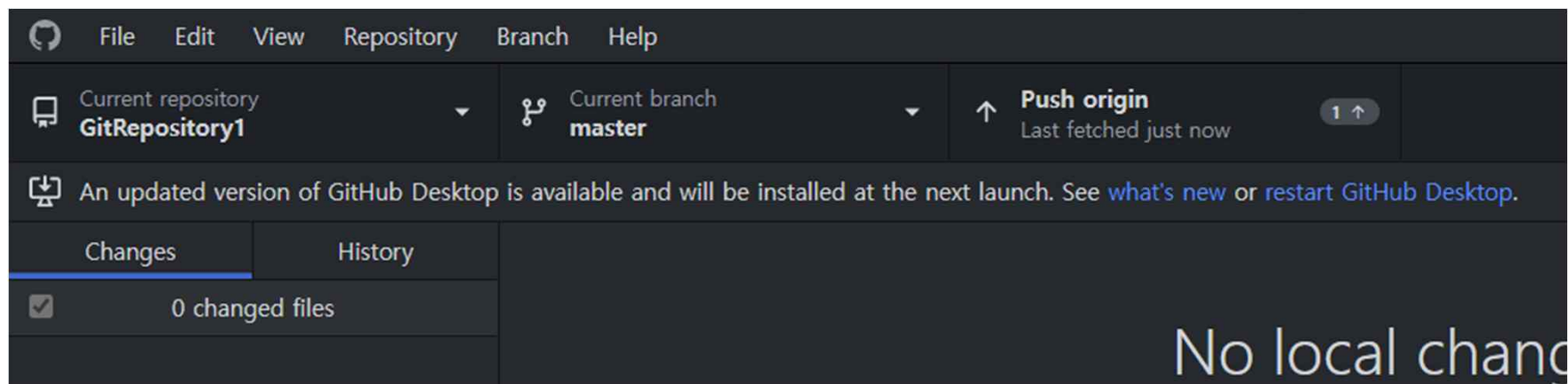
GitHub Desktop

- Commit
 - 왼쪽 아래에서 커밋 메시지를 쓴 후 [Commit to master] 버튼을 클릭함



GitHub Desktop

- Push
 - 외쪽에 [Push origin] 버튼을 클릭하면 됨



GitHub Desktop

- 타인 원격저장소에 대한 커밋 push
 - 타인의 원격저장소에 대한 커밋을 한 후 push하는 경우?
 - 자동적으로 fork를 하도록 요청함
 - fork한 후 commit 및 pull request를 수행할 수 있음

GitHub desktop v.s. Source tree

- GitHub desktop 장점
 - 소스트리는 자꾸 인증을 물어보고 어느새 인증이 풀리는 등 인증 관련하여 문제 일으키는 경우 많음
 - GitHub desktop은 GitHub 사용 한하여 인증문제로 고생할일이 없고 오류가 적음
 - GitHub에서 다운로드한 프로젝트의 소유자 별로 자동으로 분류해줌
 - 타인의 원격저장소에 대해 커밋하는 경우 자동으로 fork한 뒤 커밋하도록 함
 - 소스트리에 비해 빠름

GitHub desktop v.s. Source tree

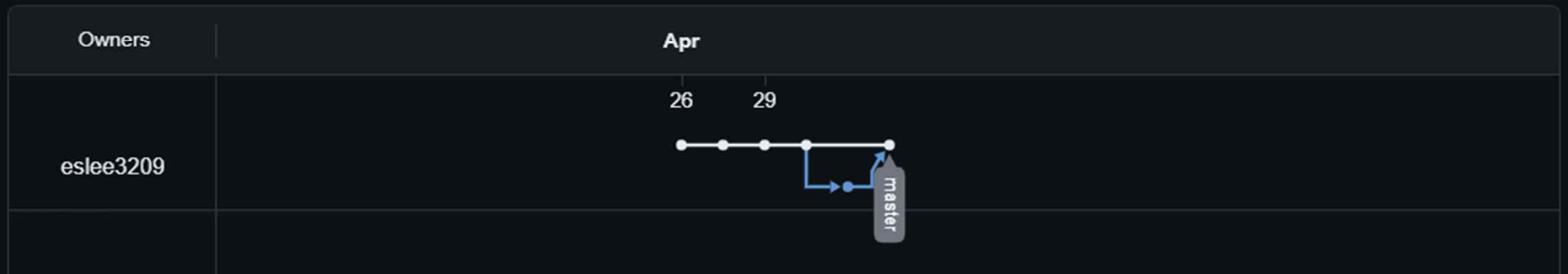
- GitHub desktop 단점
 - 소스트리보다 브랜치의 내용 파악이 어려움 (그래프x)
 - 여러 개의 체크아웃을 직접 보고 선택하기 불편함
 - 소스트리를 완전히 대체하지는 않음

커밋 history 그래프

- GitHub desktop에서 커밋 history 그래프를 예쁘게 보여 주지는 못함
- 커밋 history 그래프를 위해서는 다음 방법들을 사용할 수 있음
 - (1) 원격저장소 insights 탭
 - 원격저장소 insights 탭 > Network

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



커밋 history 그래프

- (2) Sourcetree 사용
- (3) 터미널에서 git log 와 함께 --graph 옵션 사용
 - ex) git log --oneline --graph --decorate --all

README.md

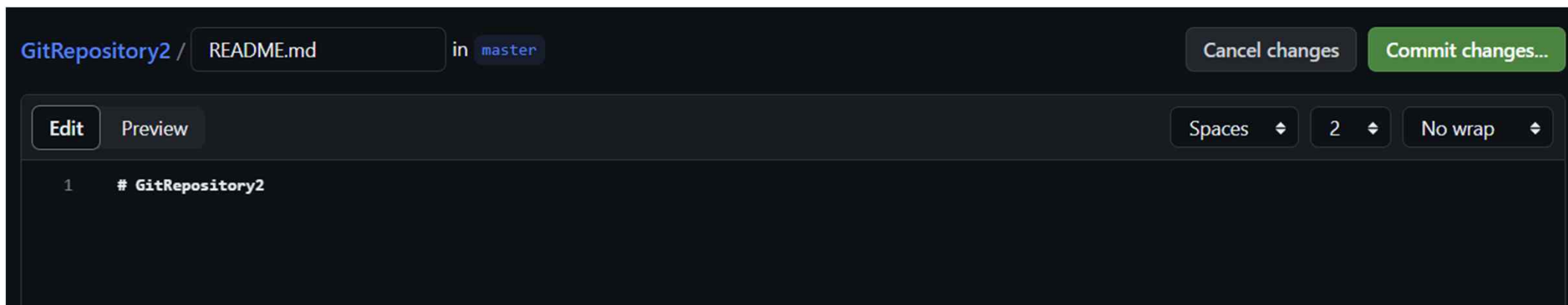
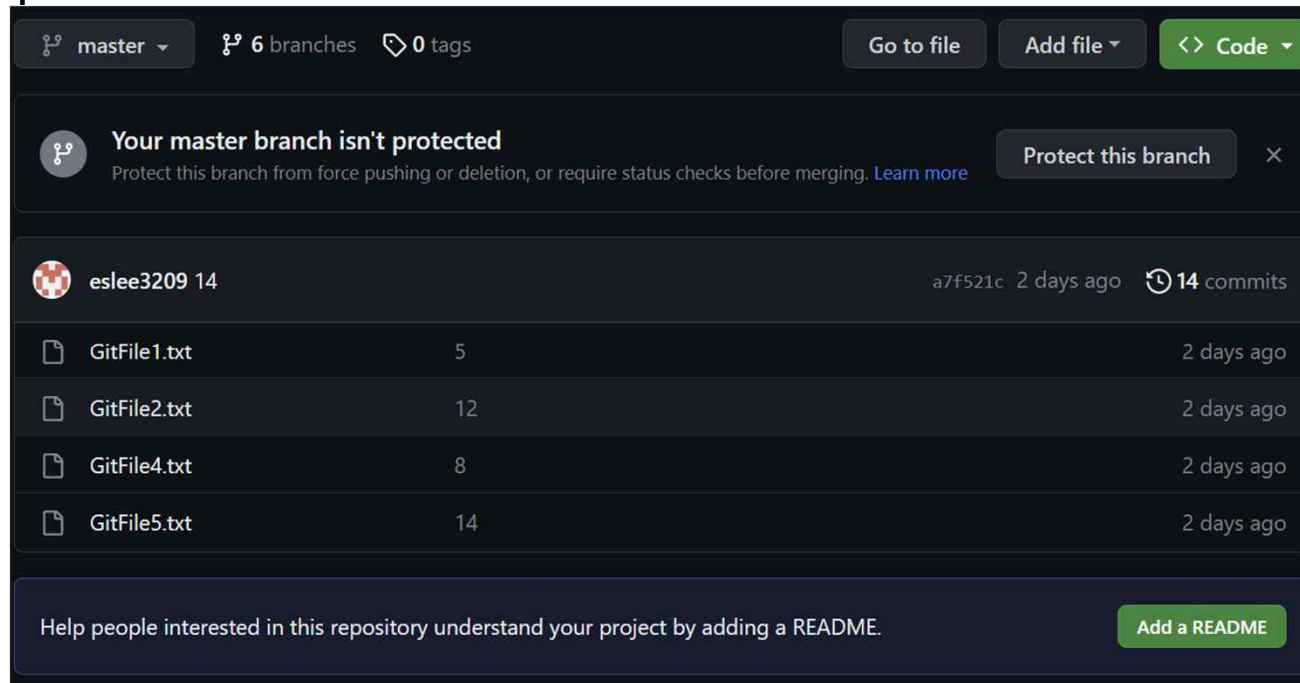
- README.md란?
 - "README.md"라는 파일은 프로젝트 디렉토리의 최상단에 위치해 있으며, 프로젝트의 개요, 설정 방법, 사용 방법 등을 설명하는 문서임
 - 이는 프로젝트에 참여하거나 해당 프로젝트를 사용하려는 사람들이 처음으로 볼 수 있는 화면이며 주로 오픈 소스 프로젝트에 사용됨
- md란?
 - "md" 확장자는 "마크다운(Markdown)"을 나타냄
 - 마크다운은 간단한 구문을 사용하여 텍스트를 구조화하는 방법으로, 웹에서 쉽게 읽고 쓸 수 있음
 - 단, 마크다운을 지원하는 프로그램이나 사이트에서만 사용이 가능하다. ex) 벨로그, 티스토리 등등

README.md

- README.md에는 보통 다음 내용들이 포함됨
 - 프로젝트의 명칭 및 로고
 - 프로젝트의 간단한 설명
 - 필요한 소프트웨어 및 하드웨어 요구 사항
 - 설치 및 설정 방법
 - 사용 방법 및 예시
 - 테스트 방법
 - 기여 방법
 - 라이선스 정보
 - 저자 및 연락처 정보
 - 참고 문헌
- 이 정보들은 다른 사람들이 프로젝트를 이해하고, 이를 올바르게 설치 및 사용하는 데 도움이 됨

README.md

- README.md 생성
 - Git repository에서 "Add a README" 버튼 클릭



README.md

- 헤더

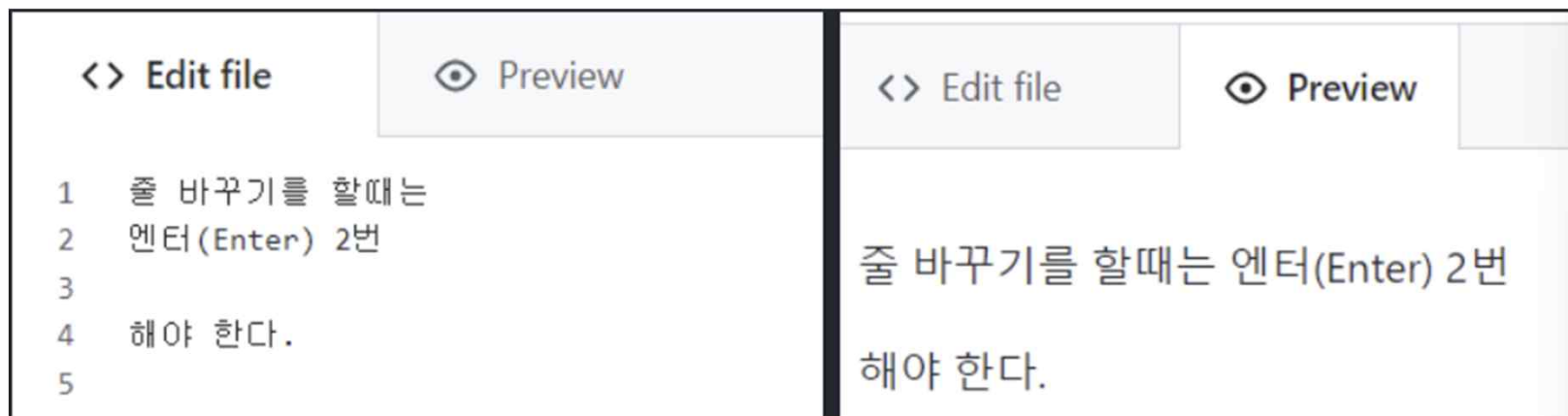
- 헤더(Header)를 입력할 때는, 텍스트 앞에 #을 붙이면 됨 (단, #과 텍스트 사이에 한 칸을 띄워야 함)
- #을 1개부터 6개까지 개수를 조절하면 위의 화면처럼 헤더 글자의 크기를 키울 수 있음



README.md

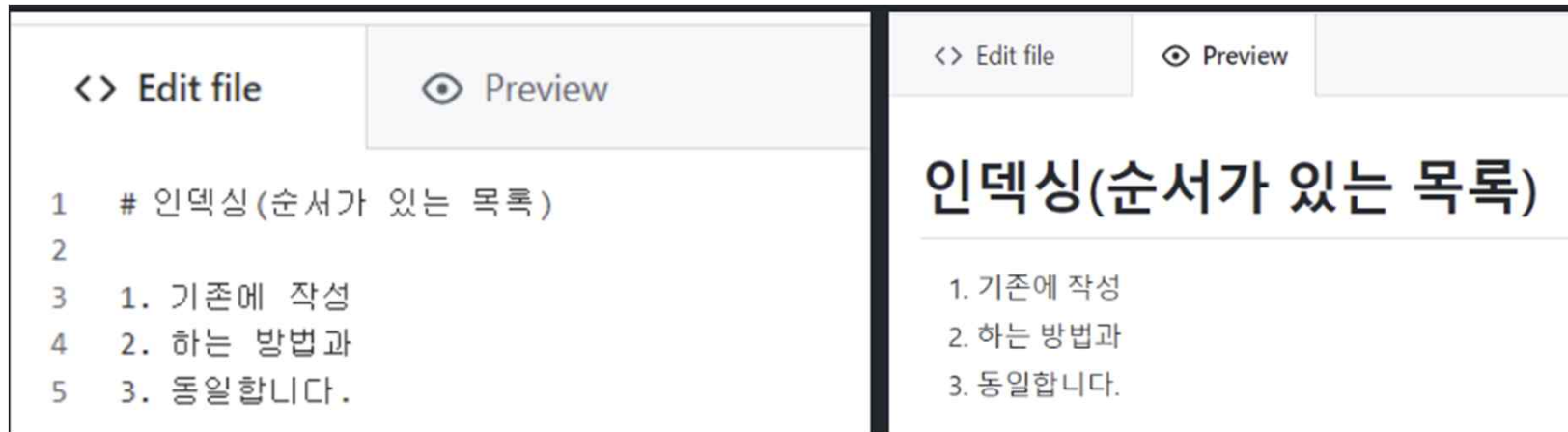
- 줄바꿈

- 줄 바꾸기(LF, Line Feed)는 엔터(Enter)를 2번 눌러서 빈 줄(Blank)을 추가하면 됨



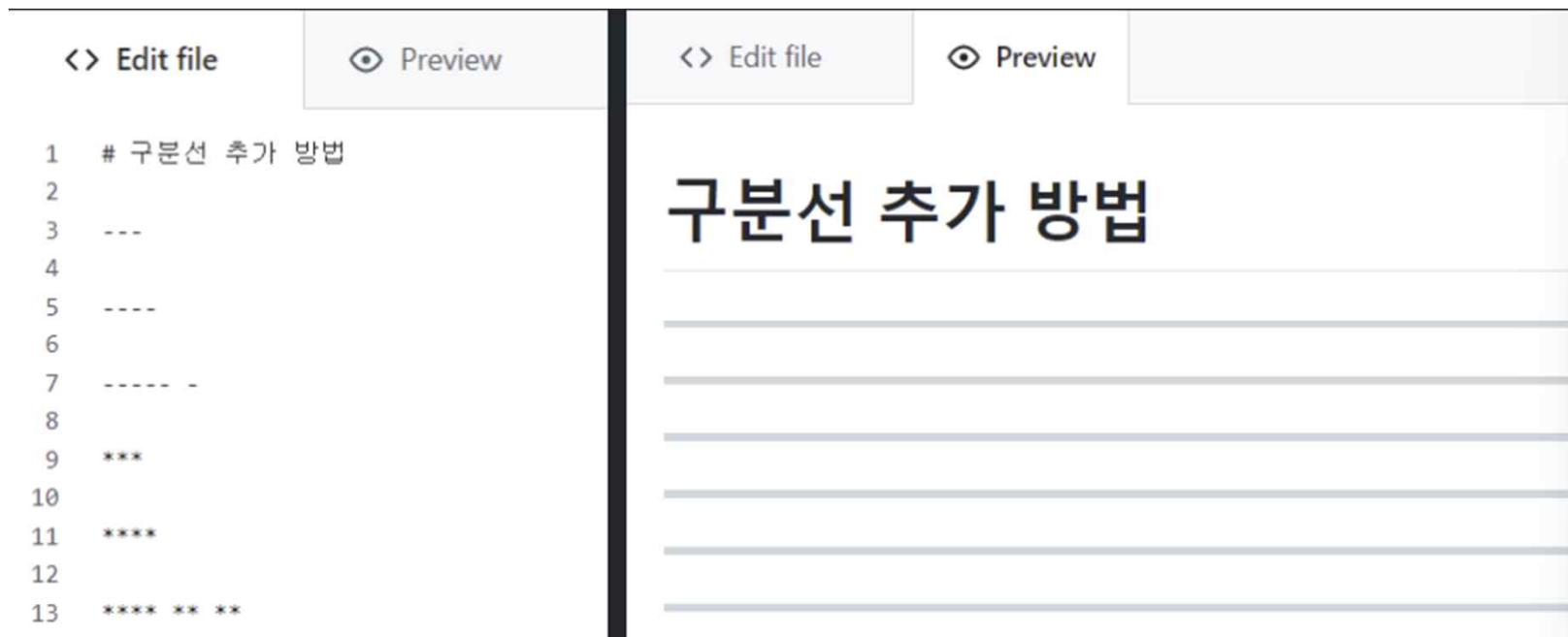
README.md

- 인덱싱
- 인덱싱(Indexing)은 순서를 매기는 것임
- 평소에 쓰는 방법과 동일 (들여 쓰기는 안 됨)



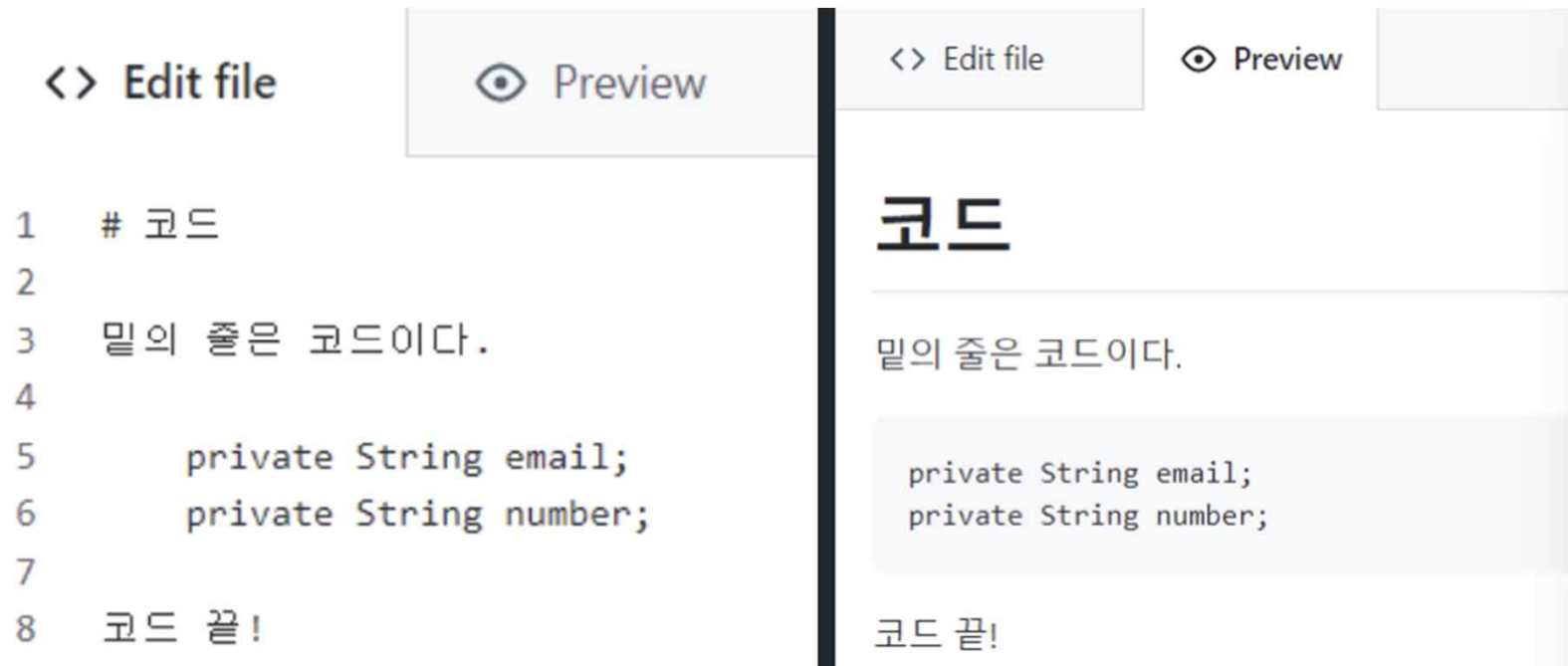
README.md

- 구분선
- 구분선(Division Line)을 추가하는 방법은 여러 가지임
- - 와 * 모두 사용이 가능하고, 3개 이상을 사용하면 됨
- 위의 화면처럼 중간에 띄어쓰기를 해도 되고, 6개 이상을 적어도 인식함



README.md

- 코드 블록
 - 코드 앞부분에 4칸을 띄어줌
 - 그리고 위아래로 한 줄씩 띄워줘야 함 (안 띄워주면 적용 안됨)



README.md

- 코드 블록
 - `를 사용하여 코드 블록 작성하는 것도 가능
 - `(백틱)는 작은따옴표가 아니라 물결(~) 표시 키가 있는 버튼의 기호
 - 인라인 코드, 즉, 작은 코드 조각의 경우 단일 백틱을 사용할 수 있음
 - 예를 들어 `print("Hello, World!")`를 렌더링하려면 마크다운에 ``print("Hello, World!")``로 작성해야 함

README.md

- 코드 블럭
 - 더 큰 코드 블록의 경우 삼중 백틱을 사용
 - 예시

```
1  # GitRepository1
2
3      def hello_world():
4          print("Hello World!")
5
6  ~~~
7  def hello_world():
8      print("Hello World!")
9  ~~~
```

GitRepository1

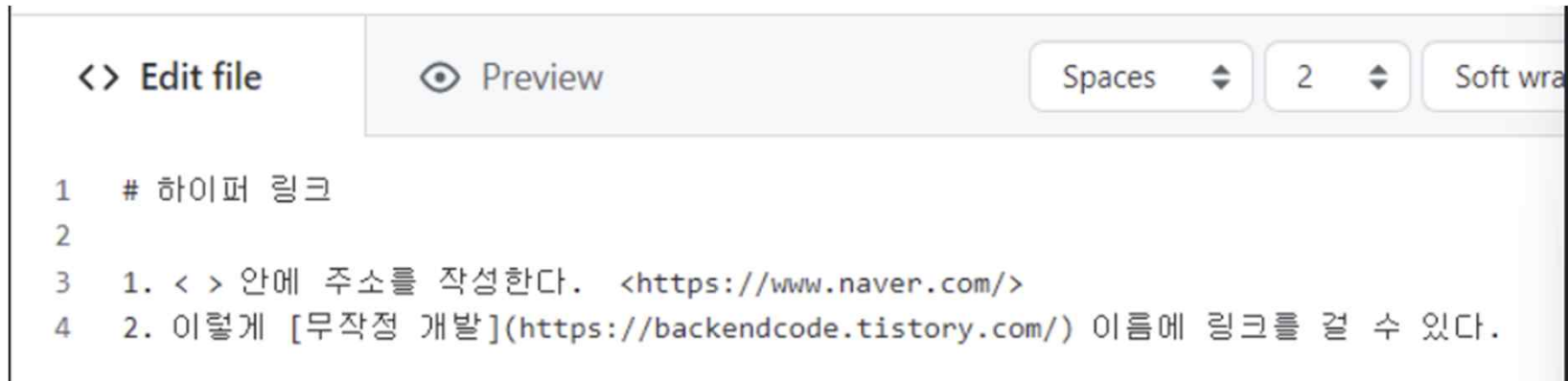
```
def hello_world():
    print("Hello World!")
```

```
def hello_world():
    print("Hello World!")
```

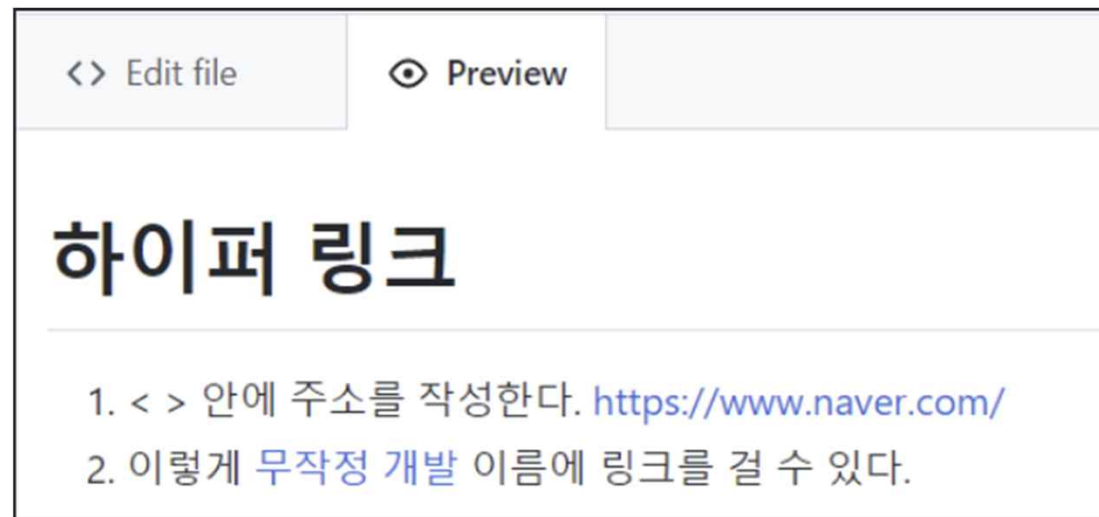
README.md

- 하이퍼링크

- 하이퍼링크(Hyperlink)의 경우 2가지 방법이 있음
- 그림처럼 꺾쇠 < > 안에 http로 시작하는 웹 주소를 넣거나,
- [링크 이름](링크 주소)를 사용할 수 있음

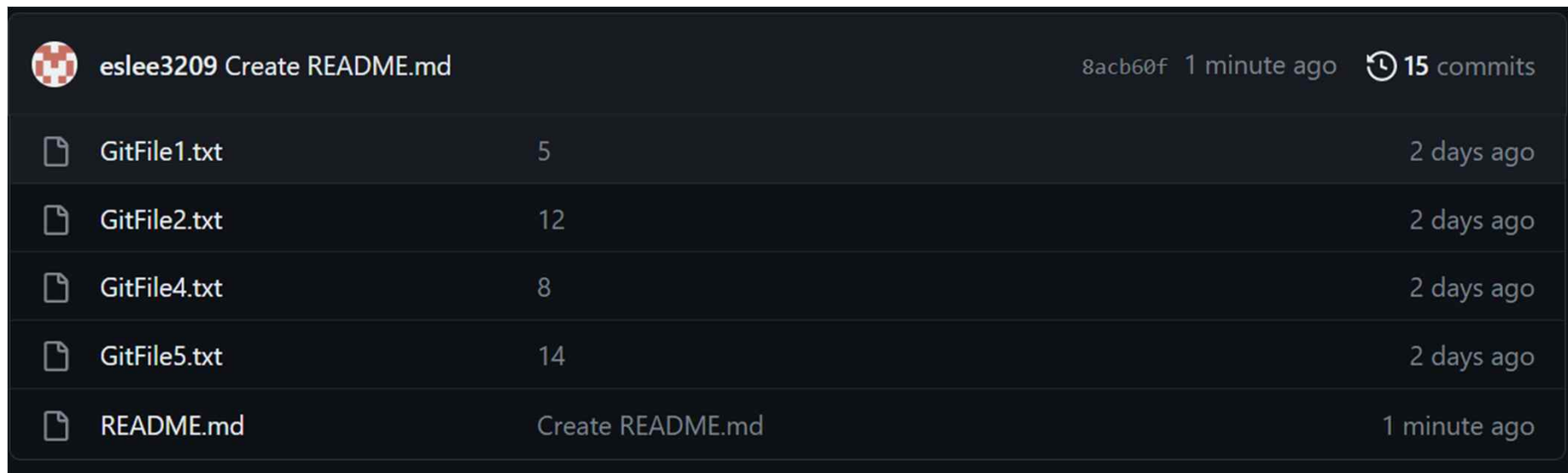


```
1 # 하이퍼 링크
2
3 1. < > 안에 주소를 작성한다. <https://www.naver.com/>
4 2. 이렇게 [무작정 개발](https://backendcode.tistory.com/) 이름에 링크를 걸 수 있다.
```








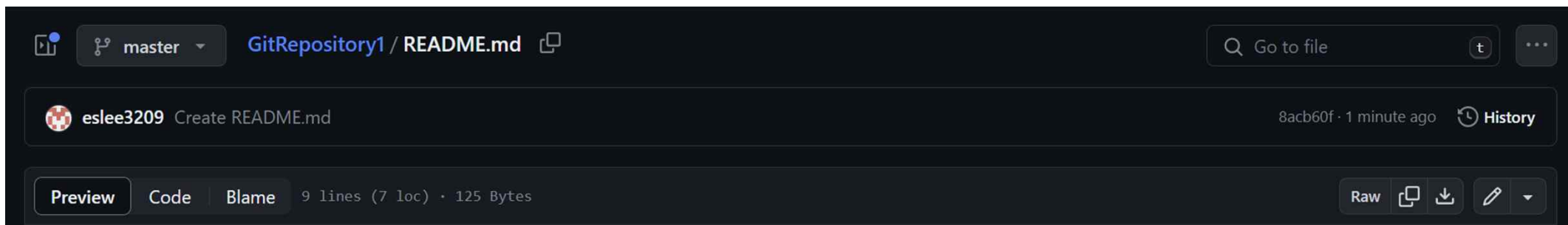
README.md

- 다른 사람 README.md 소스 복사
 - 다른 사람 repository에서 README.md 파일을 클릭
 - 오른쪽에 연필 아이콘을 클릭하면 코드 복사 가능



A screenshot of a GitHub repository interface. At the top, the repository name 'eslee3209 Create README.md' is shown, along with the commit hash '8acb60f', the time '1 minute ago', and '15 commits'. Below this is a table listing files in the repository.

	GitFile1.txt	5	2 days ago
	GitFile2.txt	12	2 days ago
	GitFile4.txt	8	2 days ago
	GitFile5.txt	14	2 days ago
	README.md	Create README.md	1 minute ago



A screenshot of a GitHub repository interface showing the 'README.md' file. The top bar shows the repository name 'GitRepository1 / README.md' and the commit hash '8acb60f · 1 minute ago'. Below this is a header for the file 'eslee3209 Create README.md' with a 'History' link. The main content area shows the file's metadata: '9 lines (7 loc) · 125 Bytes'. At the bottom, there are tabs for 'Preview', 'Code', and 'Blame'. The 'Code' tab is active, and it shows a 'Raw' button and a copy icon.

master GitRepository1 / README.md Go to file t ...

eslee3209 Create README.md 8acb60f · 1 minute ago History

Preview Code Blame 9 lines (7 loc) · 125 Bytes Raw Copy Download Edit ...