

초심자를 위한 GitLab 기초

강의 자료

■ 목차

—
초심자를 위한
GitLab 기초

- 01 특강 개요
- 02 사전 준비
- 03 Git을 활용하여 버전 관리를 해보자
- 04 실무에서 필수! Git과 GitLab 사용법
- 05 GitLab을 더 잘 활용해보자

01

특강 개요

01 특강 개요

① 이 수업을 통해

- Git을 활용하여 버전 관리 하는 방법에 대해 익힐 수 있습니다.
- 내 코드를 원격 저장소에 업로드하는 방법에 대해 익힐 수 있습니다.
- 실무에서 Git을 활용하는 방법에 대해 익힐 수 있습니다.
- GitLab을 더 잘 활용하는 방법에 대해 익힐 수 있습니다.

02

사전 준비

02 사전 준비

① Git 설치

- 실습을 진행하려면 먼저 Git이 설치되어 있어야 합니다.
- 운영체제 별로 Git을 어떻게 설치하는지 알아보겠습니다.

02 사전 준비

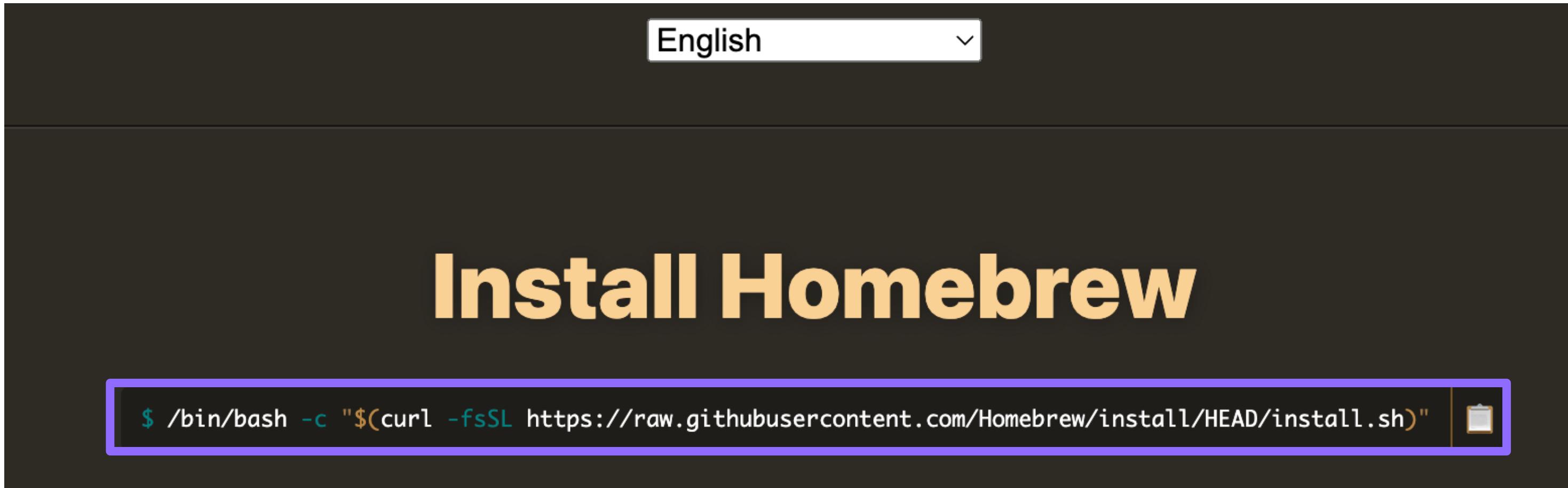
① Git 설치 - Mac OS

```
git --version  
git version 2.23.0
```

- 먼저 **git --version** 명령어를 입력하여 Git이 컴퓨터에 설치되어 있는지 확인합니다.
- 만약 git version에 대한 정보가 출력되지 않는다면, Git을 설치 해야 합니다.

02 사전 준비

⑤ Git 설치 - Mac OS



git을 설치하기 위해 먼저 Homebrew라는 Mac 용 패키지 관리 도구를 설치해야 합니다.

우측 하단에 작성된 링크를 들어가거나 아래 작성된 Homebrew를 설치하기 위해 코드를 복사합니다.

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

02 사전 준비

⑤ Git 설치 - Mac OS

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"  
==> Checking for `sudo` access (which may request your password)..  
Password:  
  
brew --version  
Homebrew 3.6.20
```

터미널 환경을 실행하여, homebrew에서 복사한 코드를 붙여넣기 후 실행합니다.

만약 비밀번호를 입력하라는 명령이 나오면 노트북 혹은 데스크탑의 비밀번호를 입력해주세요.

설치가 완료됐다면, brew --version을 입력했을 때 버전 정보가 출력된다면 설치가 마무리 됐습니다.

homebrew는 지속적으로 업데이트되는 개발 도구이기 때문에

brew의 버전이 업데이트 되면 brew --version 명령어의 결과가 달라질 수 있습니다.

02 사전 준비

① Git 설치 - Mac OS

```
brew install git  
git --version  
git version 2.23.0
```

- Homebrew 패키지 관리자를 사용하여 git 버전 관리 시스템을 설치하는 명령어인 **brew install git**을 입력하여 git을 설치합니다.
- git --version 명령어를 통해 git이 설치됐는지 확인해주세요.
- 버전 정보가 잘 노출된다면, git 설치가 완료되었습니다.

02 사전 준비

⑤ Git 설치 - Windows OS

- 먼저 Git이 컴퓨터에 설치되어 있는지 확인합니다.
- 윈도우 모양 + Q를 눌러서 GitBash라고 검색했을 때 검색이 되지 않는다면, Git을 설치 해야 합니다.
- Window의 패키지 매니저인 Chocolatey를 활용하여 Git을 설치해보겠습니다.

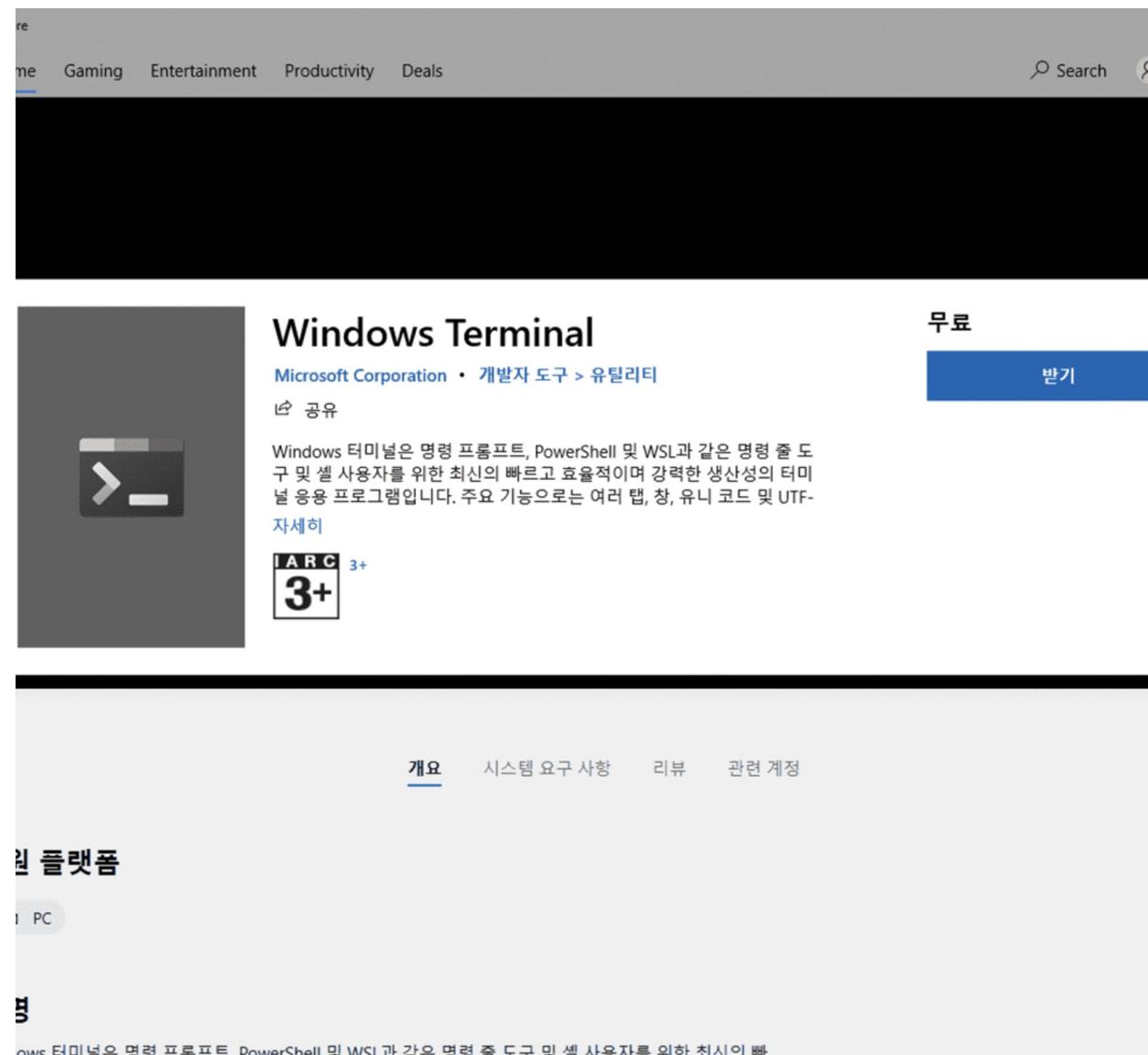
02 사전 준비

⑤ Git 설치 - Windows OS

- Chocolatey를 설치하기 앞서, 먼저 PowerShell이 설치되어 있어야 합니다.
- 최소 지원 버전은 3입니다.
- 더 낮은 버전이 설치되어 있다면 윈도우 업데이트를 통해 PowerShell을 업데이트 해야 합니다.
- 그럼 PowerShell 버전을 확인하는 방법을 알아보겠습니다.

02 사전 준비

⑤ Git 설치 - Windows OS

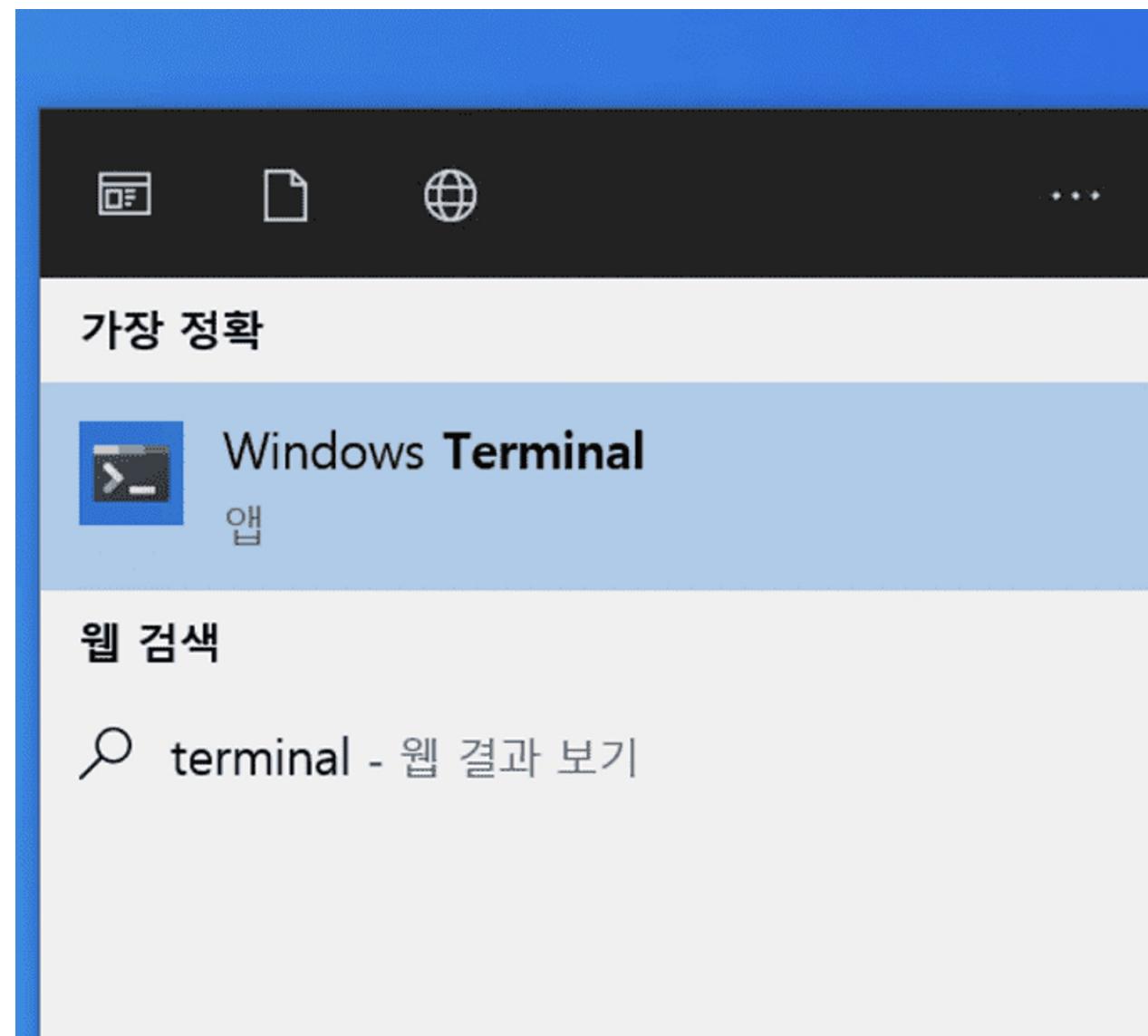


- PowerShell 버전을 확인하기 위해선 Windows Terminal에서 PowerShell을 실행합니다.
- Windows Terminal은 Microsoft Store로 설치할 수 있습니다.
- 아래 링크를 클릭하여 윈도우 스토어로 이동한 후 받기 버튼을 클릭합니다.

<https://apps.microsoft.com/store/detail/windows-terminal/9N0DX20HK701?hl=ko-kr&gl=kr&activetab=pivot%3Aoverviewtab>

02 사전 준비

⑤ Git 설치 - Windows OS



시작 버튼을 클릭하고 Terminal을 입력해
Windows Terminal 앱을 찾아 실행합니다.
Windows Terminal은 기본적으로 Windows PowerShell로 실행됩니다.

02 사전 준비

⑤ Git 설치 - Windows OS

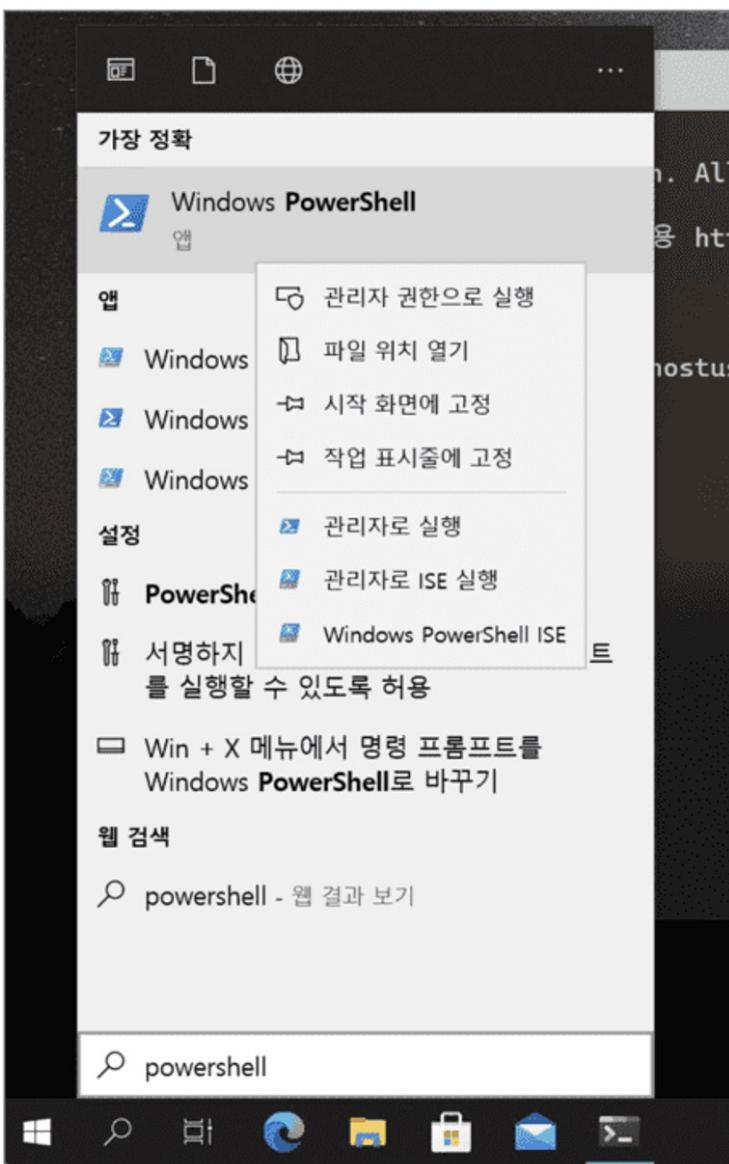
\$PSVersionTable	
Name	Value
PSVersion	5.1.19041.1023
PSEdition	Desktop
PSCompatibleVersions	{1.0, 2.0, 3.0, 4.0...}
BuildVersion	10.0.19041.1023
CLRVersion	4.0.30319.42000
WSManStackVersion	3.0
PSRemotingProtocolVersion	2.3
SerializationVersion	1.1.0.1

PowerShell을 실행했으면, 버전 확인을 위해 **\$PSVersionTable**이라고 입력합니다.

그럼 출력 내용을 통해 PowerShell 버전을 출력해서, 버전이 3이상이면 준비가 완료됐습니다.

02 사전 준비

① Git 설치 - Windows OS



다음으로, .NET Framework 4.5 이상 버전이 설치되어 있어야 합니다.

.NET Framework 버전을 확인해보겠습니다.

- Windows PowerShell 을 검색해서 실행합니다.

02 사전 준비

⑤ Git 설치 - Windows OS

```
$ (get-item 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full').GetValue("Release")  
528372
```

.NET Framework 4.0 이상 버전의 경우, PowerShell 창에서 아래 제시되어 있는 코드를 입력하면
현재 설치된 버전 정보를 확인할 수 있습니다.

```
$ (get-item 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full').GetValue("Release")  
528372 버전은 .NET Framework 4.8 버전의 예시입니다.
```

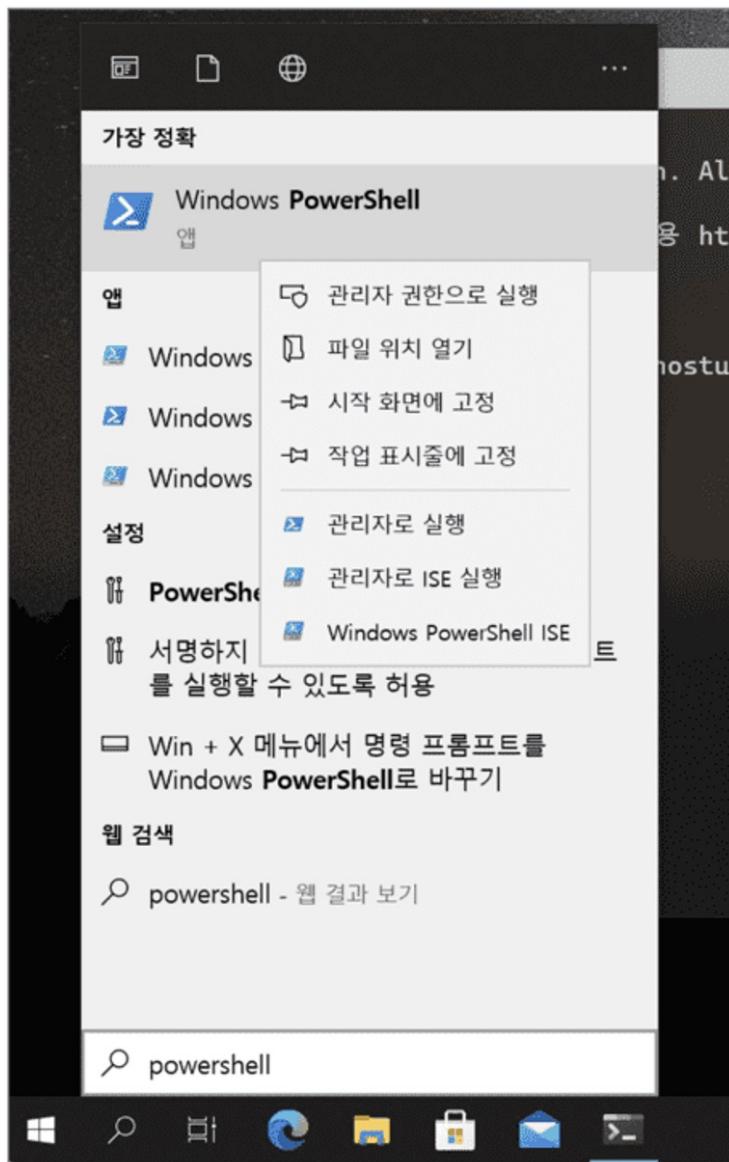
02 사전 준비

⑤ Git 설치 - Windows OS

- 만약 다른 버전의 숫자가 출력된다면, 숫자에 대응하는 버전을 확인해주세요.
- 확인 후 버전이 4.5 이상이면 .NET 버전 확인이 완료됐습니다.
 - .NET Framework 4.5, > 378389
 - 모든 Windows 운영 체제: 378389
 - .NET Framework 4.5.1, > 378675
 - Windows 8.1 및 Windows Server 2012 R2: 378675
 - 다른 모든 Windows 운영 체제: 378758
 - .NET Framework 4.5.2, > 379893
 - 모든 Windows 운영 체제: 379893
 - .NET Framework 4.6, > 393295
 - Windows 10: 393295
 - 다른 모든 Windows 운영 체제: 393297

02 사전 준비

⑤ Git 설치 - Windows OS



- PowerShell을 설치했고 .NET Framework 버전을 확인했다면 본격적으로 Chocolatey를 설치하겠습니다.
- 먼저 PowerShell에 마우스 오른쪽 클릭 후 관리자 모드로 실행합니다.

02 사전 준비

⑤ Git 설치 - Windows OS

```
Get-ExecutionPolicy
```

```
# ExecutionPolicy를 AllSigned로 설정  
$ Set-ExecutionPolicy AllSigned
```

```
# ExecutionPolicy를 Bypass로 설정  
$ Set-ExecutionPolicy Bypass -Scope Process
```

권한 확인을 위해 Get-ExecutionPolicy를 입력하여 출력 값이 Restricted가 아닌지 확인해야 합니다.

만약 Restricted라면 AllSigned나 Bypass로 설정해줍니다.

02 사전 준비

⑤ Git 설치 - Windows OS

```
$ Set-ExecutionPolicy Bypass -Scope Process -Force;
$ [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor
3072;
$ iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

...
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
```

권한 확인이 완료됐다면, Chocolatey 공식 설치 스크립트를 실행합니다. 아래 스크립트를 복사 후 실행하면 Chocolatey가 설치됩니다.

```
$ Set-ExecutionPolicy Bypass -Scope Process -Force;
$ [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072;
$ iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

Chocolatey (choco.exe) is now ready. message가 보이고, 에러 없이 완료되면 설치가 완료됐습니다.

02 사전 준비

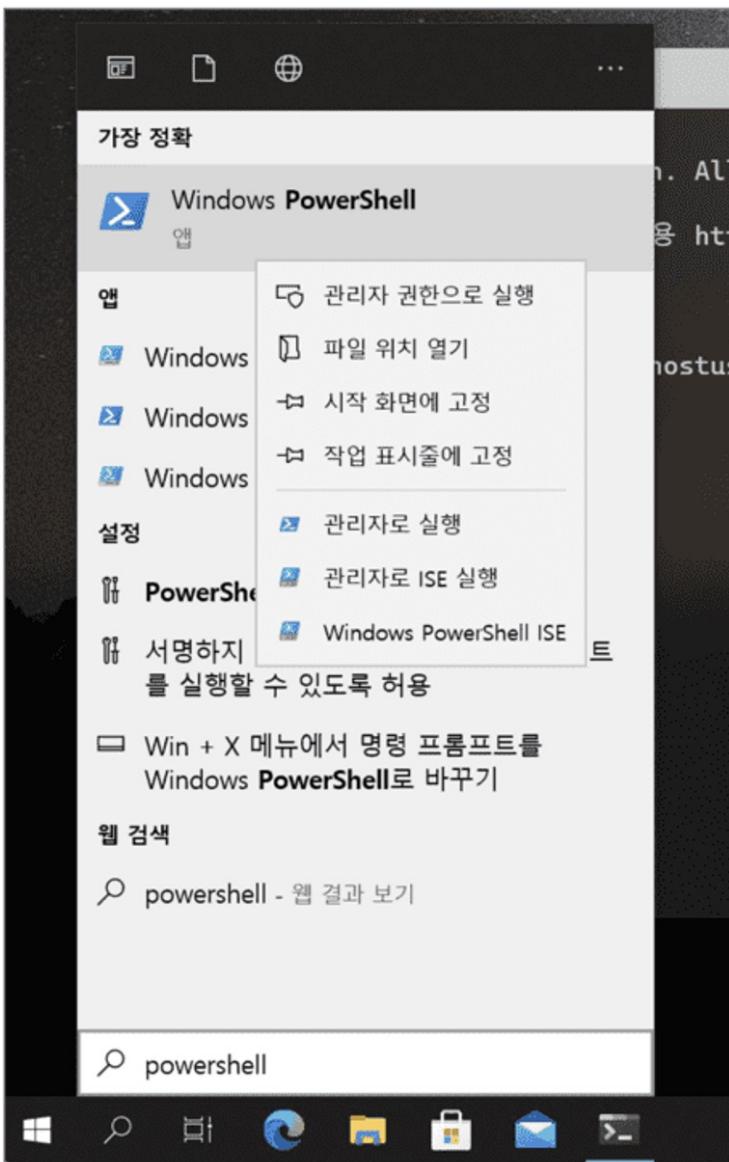
⑤ Git 설치 - Windows OS

```
$ choco
Chocolatey v0.10.15
Please run 'choco -?' or 'choco <command> -?' for help menu.
```

Chocolatey 설치가 완료됐는지 확인하기 위해 choco라는 명령어를 입력해서 버전이 출력됐다면 설치를 성공했습니다.

02 사전 준비

⑤ Git 설치 - Windows OS



- Chocolatey를 설치했다면, 다시 Windows PowerShell을 관리자 모드로 재실행해주세요.
- Windows에서는 특정 프로그램을 설치했을 때, 터미널을 재시작해주어야 확인이 되는 경우가 있기 때문에, 다시 PowerShell을 열어줍니다.

02 사전 준비

⑤ Git 설치 - Windows OS

```
$ choco install git
Chocolatey v0.10.15
Installing the following packages:
git
By installing you accept licenses for the packages.
Progress: Downloading git.install 2.32.0.2... 100%
Progress: Downloading git 2.32.0.2... 100%
...
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint):
...
```

PowerShell을 관리자 모드로 실행했다면, `choco install git` 명령어를 실행하면 Git이 설치됩니다.

중간에 정말 설치할 것인지 물어보는데, `Y`를 입력해서 설치를 진행해줍니다.

02 사전 준비

⑤ Git 설치 - Windows OS

```
$ git --version  
git version 2.32.0.windows.2
```

그 후 PowerShell을 재시작하여, **git --version** 명령어를 입력하면, git 버전 정보가 출력될 것입니다.
버전 정보가 출력됐다면, git 설치가 완료됐습니다.

03

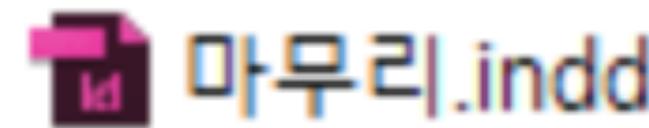
Git을 활용하여 버전 관리를 해보자

⑤ Git이란 무엇일까요?

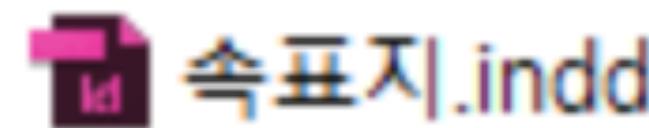
- Git은 오늘날 가장 많이 사용하는 버전 관리 시스템입니다.
- 버전이란 프로그램을 수정, 개선할 때의 변화가 결과물로 나온 것이라 말할 수 있습니다.
- 개발자에게 버전 관리를 잘 하는 것은 대단히 중요합니다.
- 개발을 공부하기 전에 여러분들은 버전 관리를 어떻게 해오셨나요?

03 Git을 활용하여 버전 관리를 해보자

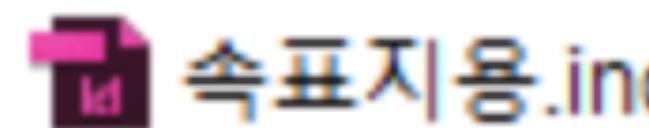
⑤ 우리는 어떻게 버전 관리를 해왔는가?



마무리.indd



속표지.indd



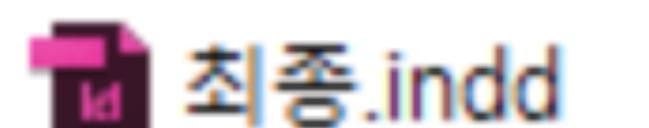
속표지용.indd



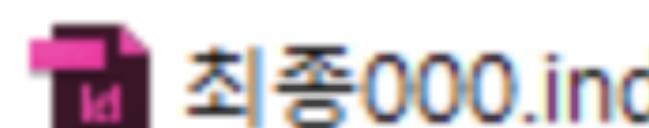
시안.indd



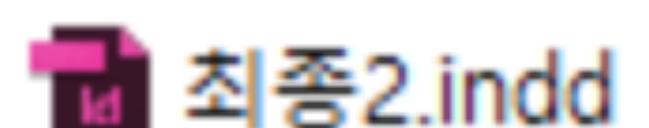
정말 끝.indd



최종.indd



최종000.indd



최종2.indd

- 최종, 최최종, 진짜 최종..
- PPT를 만든다고 생각하면, 마지막 버전을 만들기 위해 여러 버전들을 나눠서 만들어본 경험이 있을 것 입니다.
- 최종, 최최종, 리얼 최종 등의 파일 중에서 백업 버전이 필요하다면 어떤 버전으로 백업을 해야 할까요?

⑤ 버전 관리를 효율적으로 하도록 돕는 Git



- 버전 관리의 어려움을 효과적으로 해결해준 시스템이 바로 Git입니다.
- Git은 어떻게 버전 관리를 하는지 알아보겠습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전이 만들어지는 3단계

Working directory

내가 코드를 작성하는 공간
입니다.

Staging Area

버전이 될 코드가 존재하는
공간입니다.

Repository

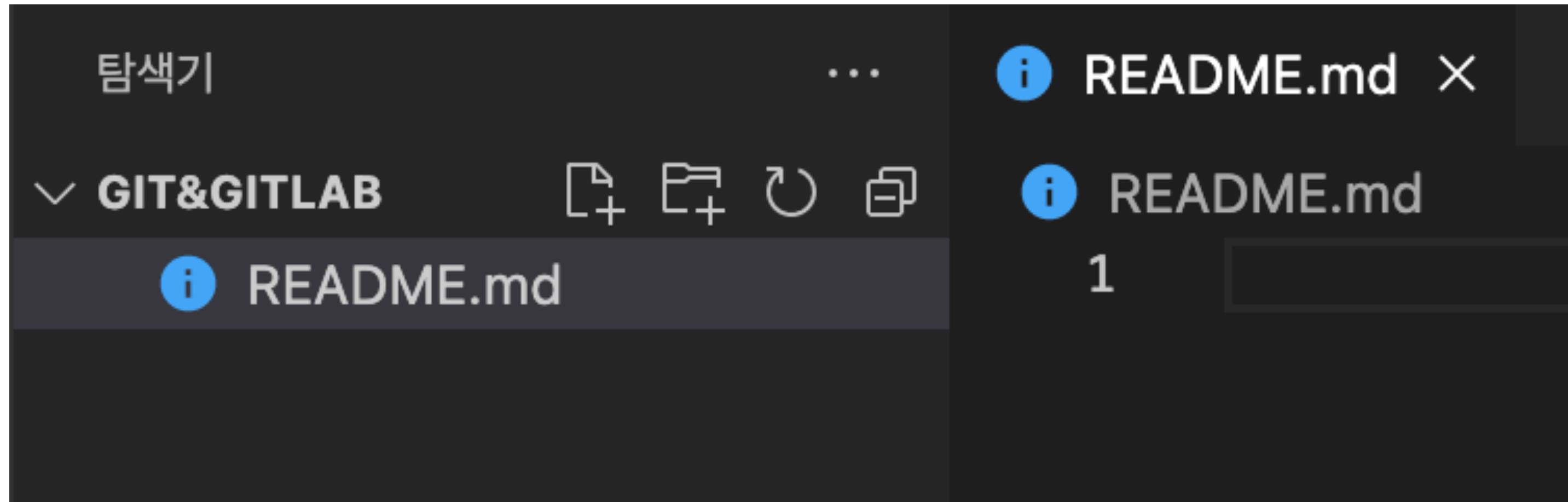
버전이 저장되어 있는 공간
입니다.

코드를 생성, 수정, 삭제되
는 공간입니다.

Working directory에서
버전으로 만들고 싶은 파일
들을 저장하는 공간입니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습



VSCode를 실행한 후, GIT&GITLAB이라는 폴더를 만들었습니다.

폴더에 README.md 파일을 생성해줍니다.

여기서 README란 무엇일까요?

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습

The screenshot shows the GitHub repository page for Nest.js. At the top, there's a large logo with a red lion icon and the word "nest". Below it, a sub-headline reads "A progressive Node.js framework for building efficient and scalable server-side applications." The main content area displays the README.md file. It includes sections for "Installation", "Running the app", and a code block for "development mode".

Installation

```
npm install
```

Running the app

```
development
npm run start

watch mode
npm run start:dev

production mode
npm run start:prod
```

- README.md 파일은 이름에서 알 수 있듯, 프로젝트 코드를 살펴보기 전에 코드가 어떻게 구성됐는지, 아키텍처는 어떻게 구성됐는지 등 프로젝트의 내용을 정리하는 문서입니다.
- README 문서를 작성하기 위해선 Markdown 언어를 작성해야 합니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습

The screenshot shows a code editor window with two tabs: 'README.md' and 'Preview README.md'. The 'README.md' tab contains the following content:

```
> ## 엘리스 특강 > ### 엘리스 Git GitLab 특강
1 # 엘리스
3 ## 엘리스 특강
5 ### 엘리스 Git GitLab 특강
6
7 엘리스 특강을 위해 README
작성중입니다.
8
9 - 리스트를 작성합니다.
10 - 리스트2를 작성합니다.
11
```

The 'Preview README.md' tab shows the rendered HTML output:

엘리스
(HTML <H1> 태그)
엘리스 특강
(HTML <H2> 태그)
엘리스 Git GitLab 특강
(HTML <H3> 태그)

엘리스 특강을 위해 README 작성중입니다.

- 리스트를 작성합니다.
- 리스트2를 작성합니다.

- Markdown에서 사용할 수 있는 다양한 문법이 있습니다.
예를 들어 아래와 같은 문법이 있습니다.
- # (HTML <H1> 태그)
(HTML <H2> 태그)
(HTML <H3> 태그)
- (번호가 없는 리스트)
 - 자세하게 알고 싶으신 분들은 Markdown 문법을 검색해서 공부해보세요!

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습

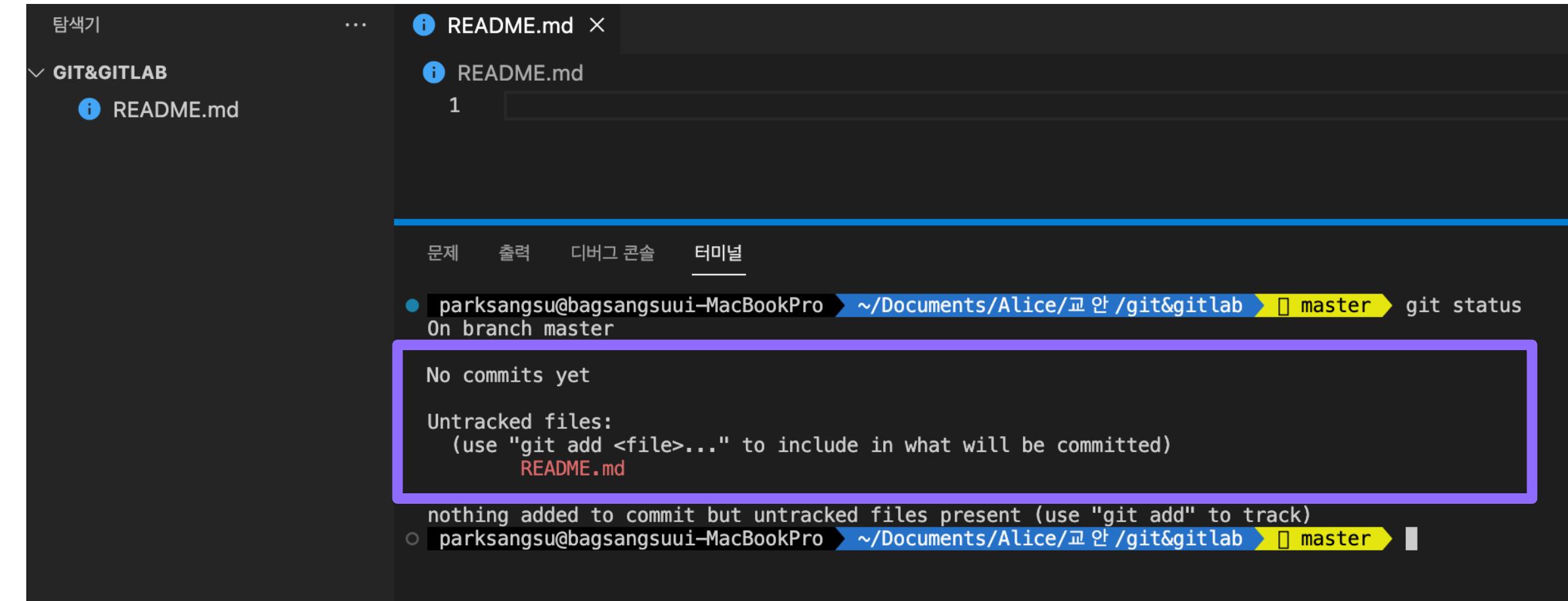
문제	출력	디버그 콘솔	터미널

```
● parksangsu@bagsangsuum-MacBookPro ~Documents/Alice/교안/git&gitlab git init  
Initialized empty Git repository in /Users/parksangsu/Documents/Alice/교안/git&gitlab/.git/  
○ parksangsu@bagsangsuum-MacBookPro ~Documents/Alice/교안/git&gitlab [master] █
```

Git 저장소를 초기화하는 명령어인 **git init**을 입력하여
현재 있는 디렉터리에서 버전 관리를 시작하겠다고 선언하겠습니다.

03 Git을 활용하여 버전 관리를 해보자

✓ 버전 만들기 실습



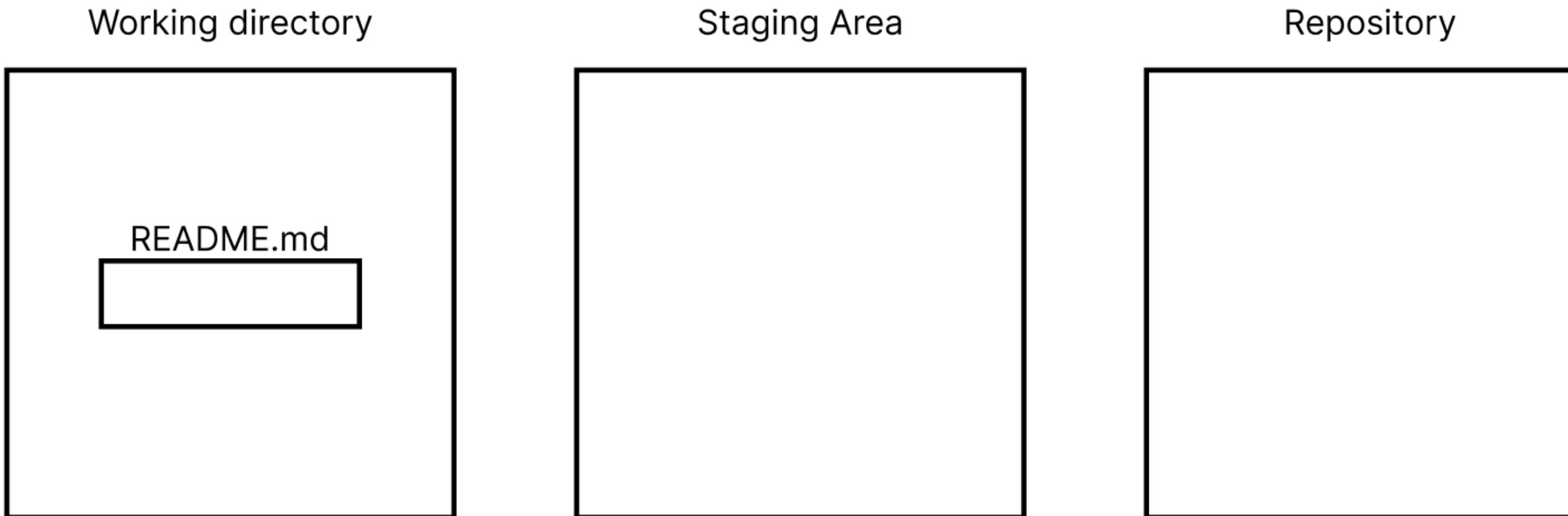
```
parksangsu@bagsangsuum-MacBookPro ~/Documents/Alice/교안/git&gitlab master git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.md
nothing added to commit but untracked files present (use "git add" to track)
```

git status 명령어를 입력하면 현재 버전 관리가 되는 폴더에 현 상황을 알려줍니다.

현재는 commit이 없다는 내용이 출력되고 있고, 트래킹 되지 않은 파일이 있다고 보여주고 있습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습



여기서 트래킹되지 않았다는 것은
Staging Area에 파일이 추가되지 않았다는 뜻입니다.



Staging Area

Staging Area는 버전이 될 코드가
존재하는 공간입니다.

Working directory에서 버전으로
만들고 싶은 파일들을 저장하는 공간
입니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습

The screenshot shows a terminal window with the following content:

```
git add .
git commit -m "Initial commit"
```

The terminal shows the command `git add .` being entered and then completed. The command `git commit -m "Initial commit"` is also visible at the bottom.

그럼 Staging Area에 버전으로 추가하고 싶은 파일을 추가하기 위해 README.md 파일에 테스트1이라고 입력하고 저장하겠습니다.
그 후 git add . 를 입력하여 Staging Area에 파일을 올려보겠습니다.



add

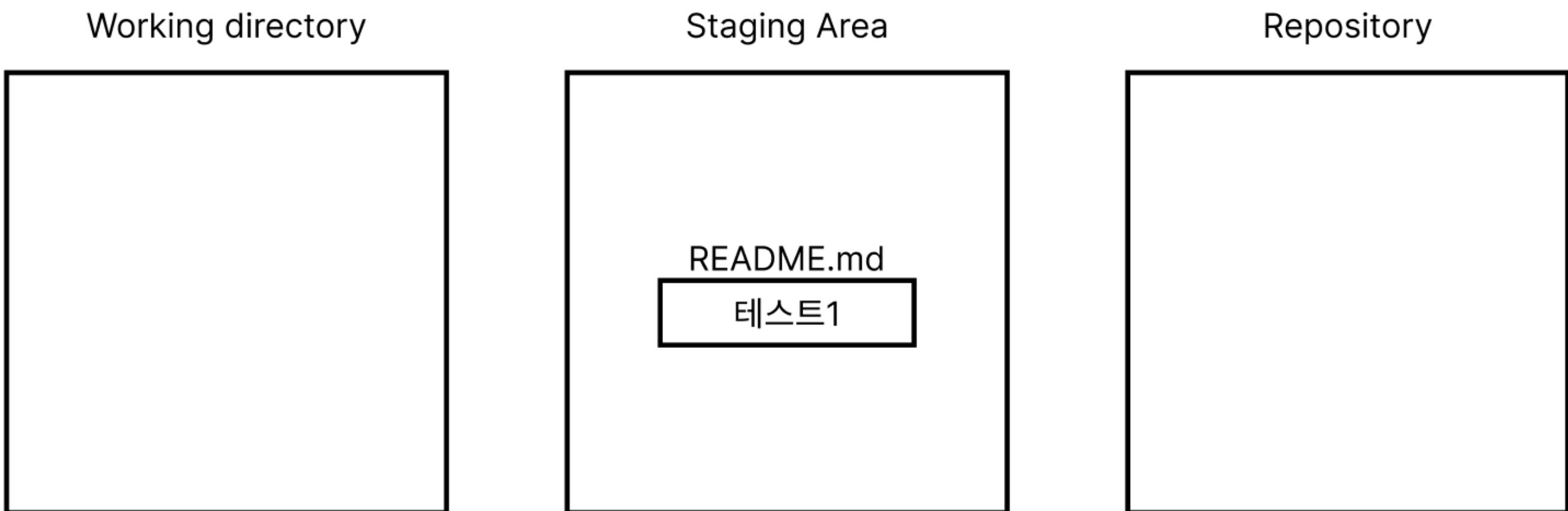
add 명령어 뒤에 . 을 붙이면 현재 경로의 모든 파일들을 Staging Area로 추가하겠다는 뜻입니다.

만약 특정 파일만 Staging Area로 추가하고 싶다면, add 뒤에 파일명을 붙입니다.

ex) git add README.md

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습



git add . 명령어를 활용하여
방금 생성한 파일의 변경 사항을 Staging Area에 업로드했습니다.



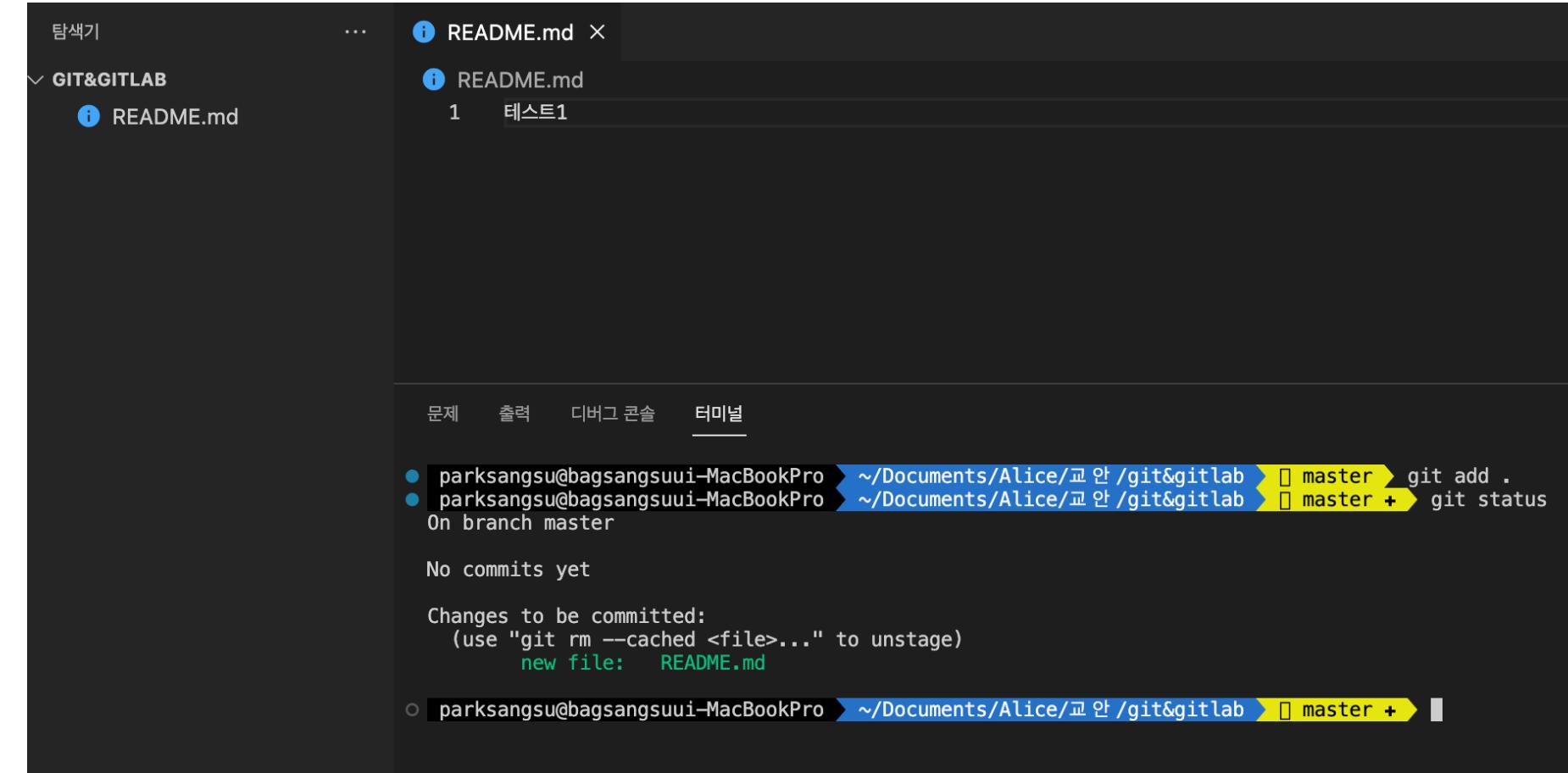
Staging Area

Staging Area는 버전이 될 코드가
존재하는 공간입니다.

Working directory에서 버전으로
만들고 싶은 파일들을 저장하는 공간
입니다.

03 Git을 활용하여 버전 관리를 해보자

✓ 버전 만들기 실습



The screenshot shows a terminal window with a dark background. On the left, there's a file tree view with a single file named 'README.md' under a folder named 'GIT&GITLAB'. The main area of the terminal shows the following text:

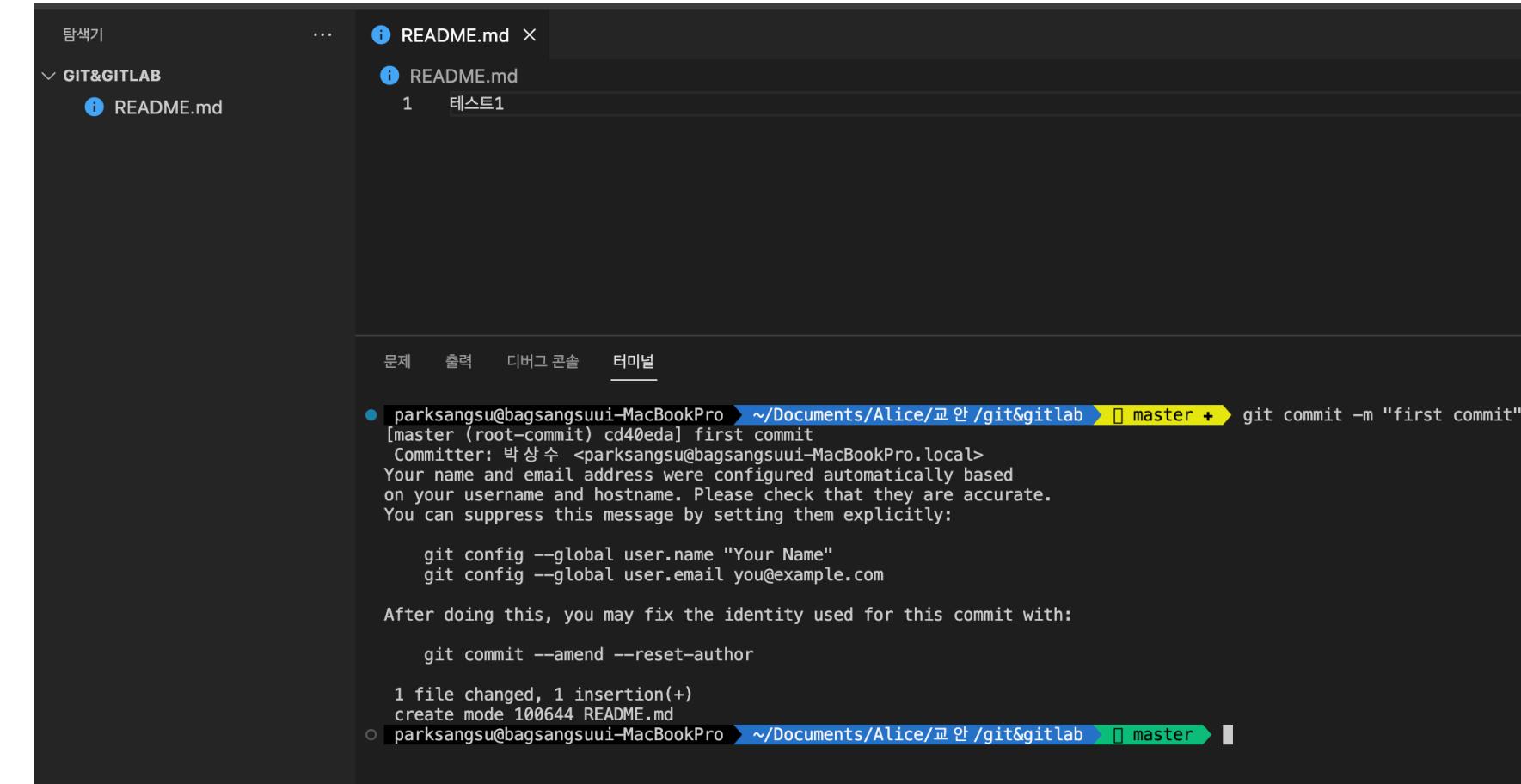
```
터미널
parksangsu@bagsangsuum-MacBookPro ~Documents/Alice/교안/git&gitlab master git add .
parksangsu@bagsangsuum-MacBookPro ~Documents/Alice/교안/git&gitlab master + git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file: README.md

```

git status 명령어를 입력해보면,
현재 README.md 파일이 트래킹되고 있는 것을 확인할 수 있습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습



The screenshot shows a terminal window with a dark theme. At the top, there's a file browser sidebar with a single item: 'GIT&GITLAB' containing 'README.md'. The main area is a terminal window with the following text:

```
parksangsu@bagsangsuum-MacBookPro ~/Documents/Alice/교안/git&gitlab [master] + git commit -m "first commit"
[master (root-commit) cd40eda] first commit
Committer: 박상수 <parksangsu@bagsangsuum-MacBookPro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
git config --global user.name "Your Name"
git config --global user.email you@example.com
After doing this, you may fix the identity used for this commit with:
git commit --amend --reset-author
1 file changed, 1 insertion(+)
create mode 100644 README.md
parksangsu@bagsangsuum-MacBookPro ~/Documents/Alice/교안/git&gitlab [master]
```

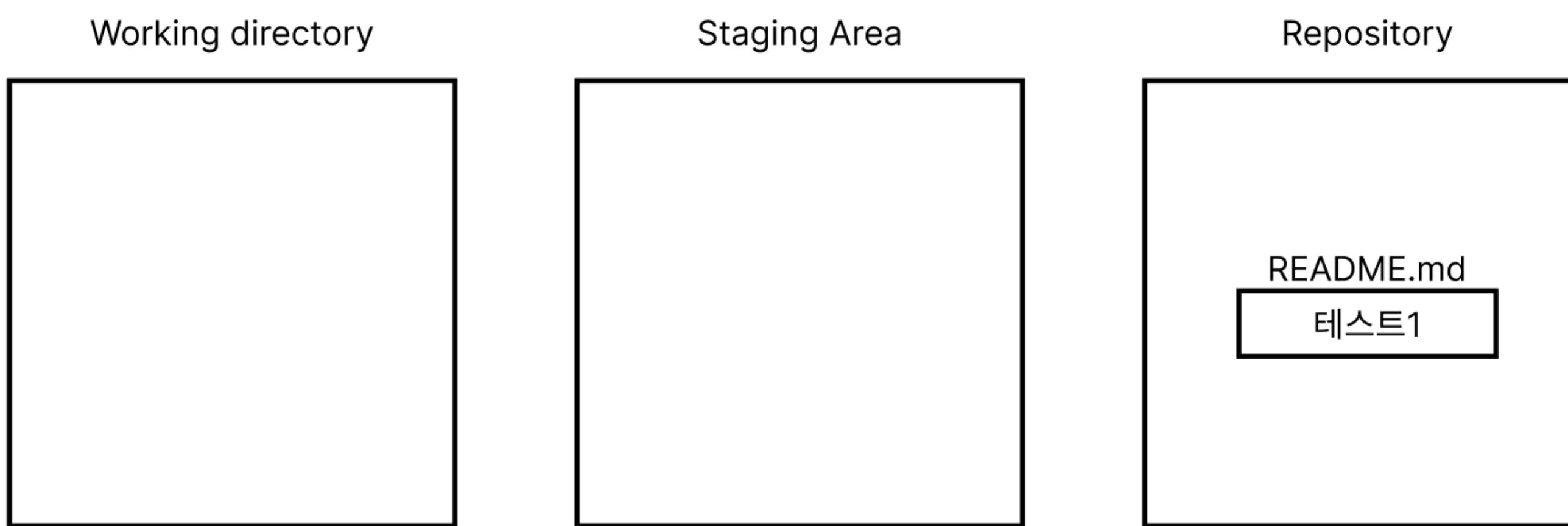
테스트1 이라고 적혀진 파일을 버전으로 저장하기 위해 commit 명령어를 사용해보겠습니다.

git commit 명령어를 활용할 땐 반드시 commit message를 입력해줘야 합니다.

commit message에는 버전에 대한 정보를 입력해줍니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습



git commit 명령어를 통해
버전이 저장되어 있는 공간인 Repository에 파일의 버전을 저장했습니다.

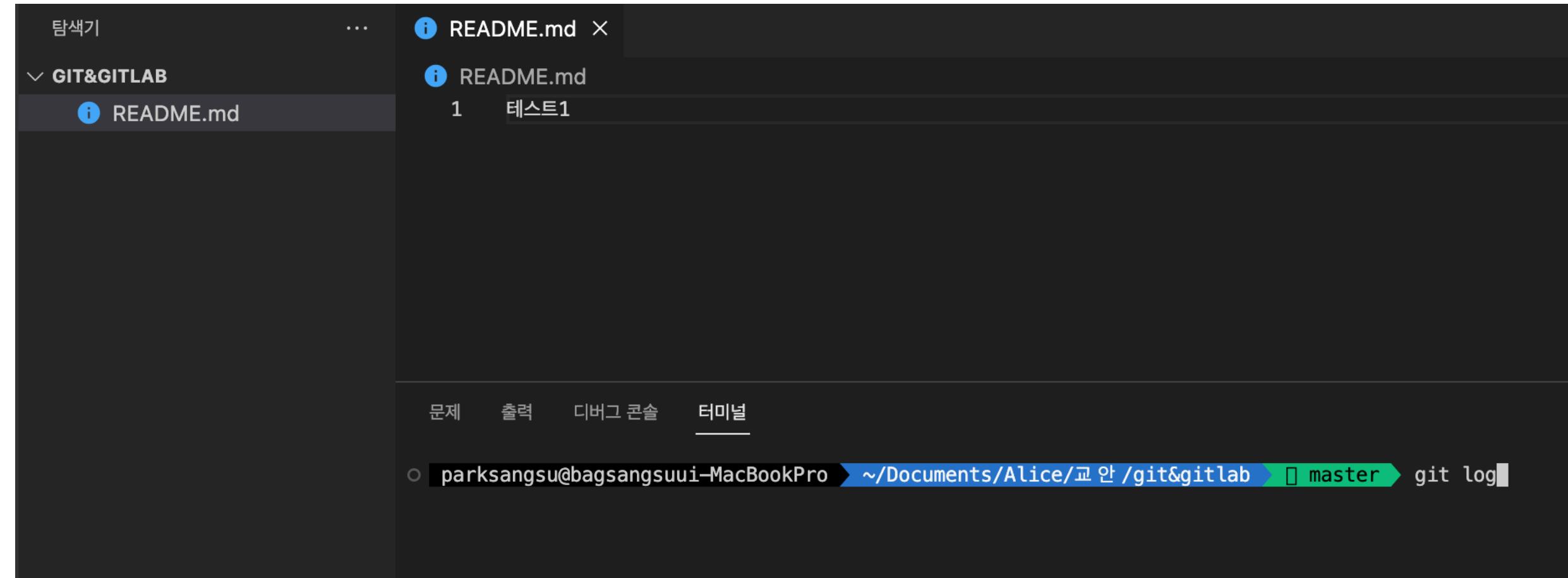


Repository

Repository는 버전이 저장되어 있는 공간입니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습



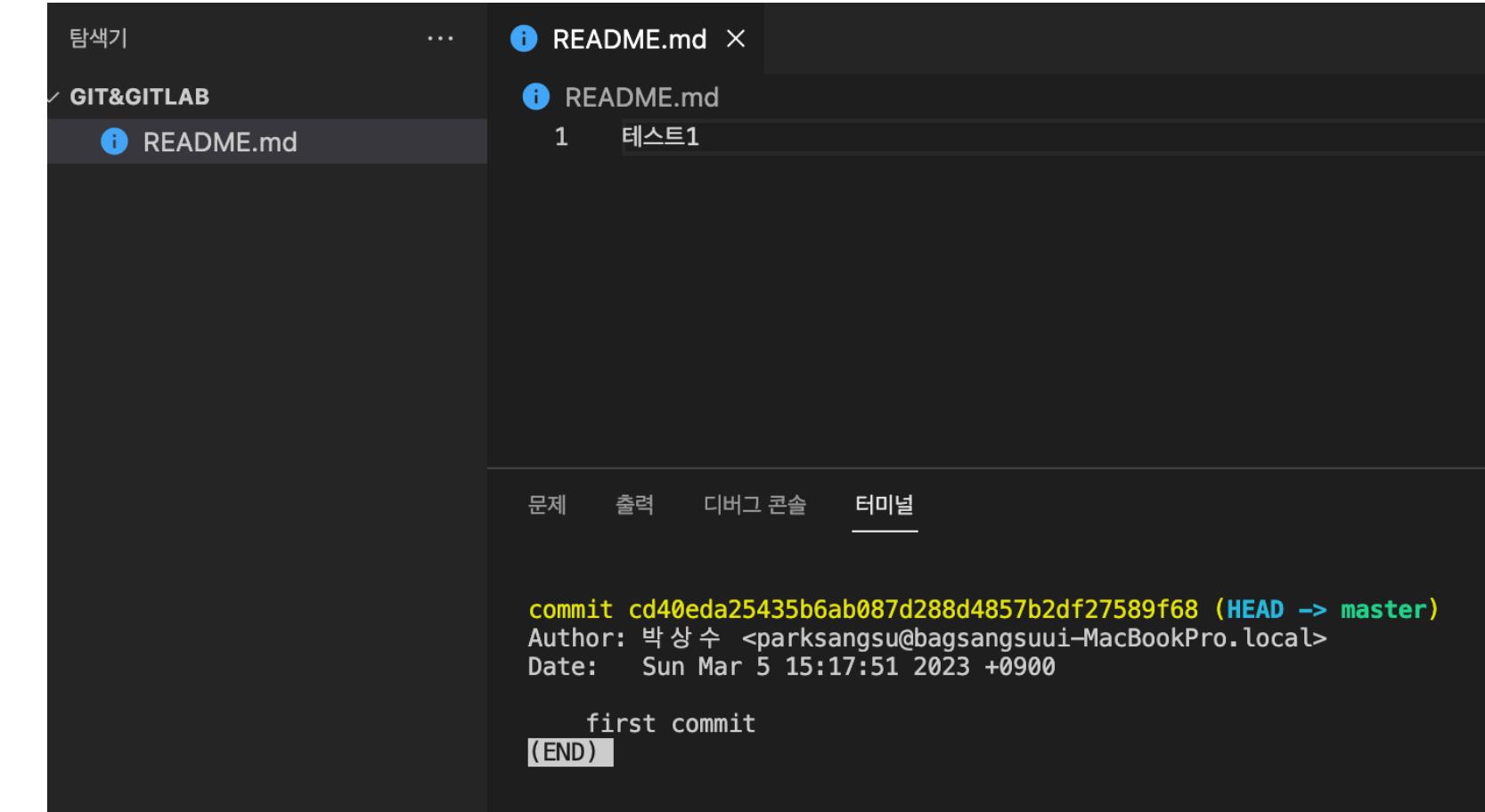
The screenshot shows a terminal window with a dark theme. On the left, there's a file explorer sidebar with a single file named 'README.md' under a folder named 'GIT&GITLAB'. The main area of the terminal shows the output of the 'git log' command:

```
parksangsu@bagsangsuum-MacBookPro ~/Documents/Alice/교안/git&gitlab master git log
```

commit이 제대로 되었는지, 버전이 제대로 만들어졌는지 확인하기 위해
git log 명령어를 입력해보겠습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 버전 만들기 실습



A screenshot of a terminal window showing a git commit log. The terminal has a dark theme. At the top, there's a file browser sidebar with a tree view showing 'GIT&GITLAB' and a file list with 'README.md'. The main area shows a commit message:

```
commit cd40eda25435b6ab087d288d4857b2df27589f68 (HEAD -> master)
Author: 박상수 <parksangsu@bagsangsui-MacBookPro.local>
Date:   Sun Mar 5 15:17:51 2023 +0900

    first commit
(END)
```

git log 명령어를 입력하면, 터미널에 내가 작성한 버전 정보들이 출력됩니다.

commit 옆에 있는 'cd40eda25 ~~~' 같은 commit 버전을 식별할 수 있는 고유값입니다.

만약 특정 버전으로 복구하고 싶을 때 저 고유값을 사용하여 복구할 수 있습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기

- 지금까지 내 로컬 환경에서 버전을 생성하고, 버전을 저장했습니다.
- 이 말은 즉 내 컴퓨터에서 생성한 코드의 버전은 아직은 나만 사용할 수 있다는 뜻입니다.
- 내가 생성한 코드 버전을 다른 개발자들도 사용할 수 있으려면 어떻게 해야 할까요?
- 이 때 사용하는 것이 바로 원격 저장소입니다.
- 원격 저장소에 내가 생성한 코드를 업로드함으로써 다른 개발자들에게 내가 생성한 코드를 공유할 수 있습니다.
- 그럼 지금부터 원격 저장소가 무엇인지, 어떻게 활용할 수 있는지 알아보겠습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기



GitLab



GitHub

많은 개발자들이 사용하는 원격 저장소의 예시로는 GitLab과 GitHub이 있습니다.

GitLab과 GitHub의 차이에 대해 잠시 알아보겠습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기



- Github는 세계에서 가장 큰 개발자 커뮤니티 중 하나여서, 다양한 오픈 소스 프로젝트를 찾아볼 수 있습니다.
- 사용자 경험 우수하다는 장점이 있습니다.
- Private Repository 활용을 하려면 유료로 이용해야 합니다. 만약 유료 저장소를 무료로 사용하려고 한다면 일부 기능만 제공됩니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기



- GitLab은 무료로 Private Repository을 사용 가능합니다.
- 수많은 CI/CD 파이프라인 도구 활용할 수 있습니다.
- 온프레미스 설치 지원을 통해 GitLab 자체 호스팅을 하여 데이터 보안을 강화할 수 있습니다.
- GitLab의 이슈 트래커를 활용하여 이슈를 체계적으로 관리 가능합니다.
- 내가 만든 버전을 다른 팀원과 공유하기 위해 GitLab을 활용해보겠습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기

The screenshot shows a GitHub profile for a user named [코치] 박상수 (@constP). The profile includes a photo, bio, and stats: Member since May 19, 2022, 0 followers, and 0 following. Below the bio is an activity heatmap showing commits over time (Mar to Mar) and day of the week (M, W, F). A tooltip indicates the heatmap represents issues, merge requests, pushes, and comments. To the right, there are sections for 'Activity' (with a blurred timeline) and 'Personal projects'. The 'Personal projects' section is highlighted with a purple rectangle and contains the message: 'You haven't created any personal projects. Your projects can be available publicly, internally, or privately, at your choice.' A 'New project' button is visible at the bottom of this section.

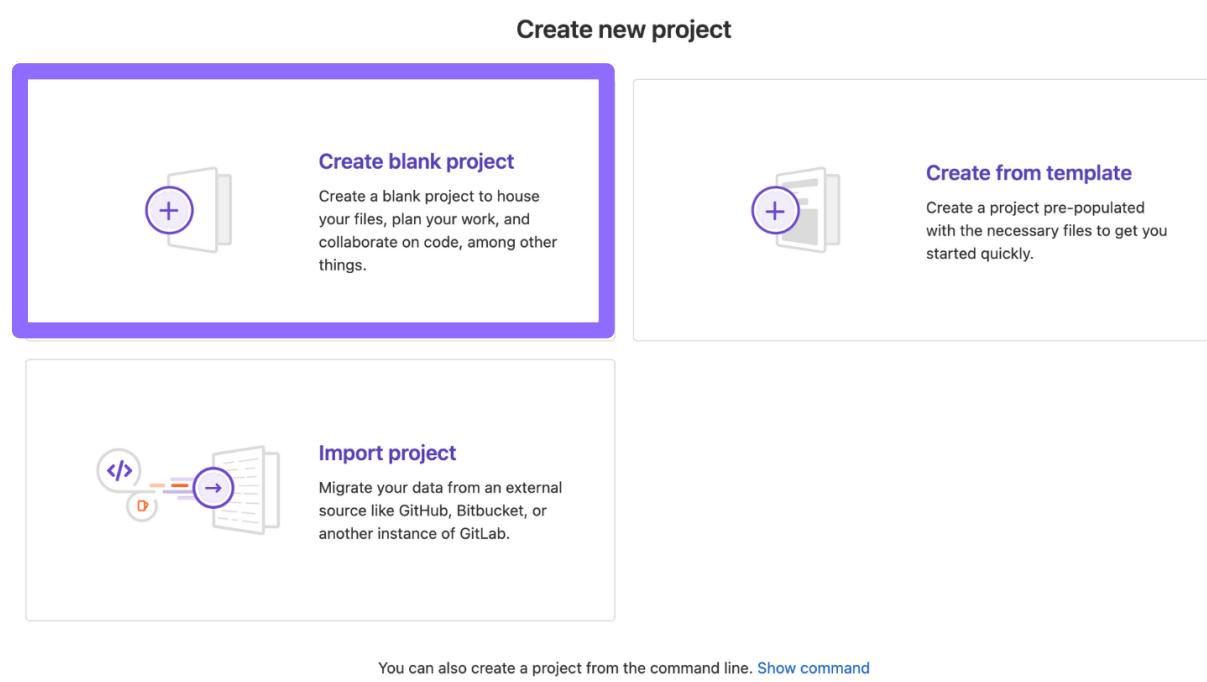
실습을 위한 Repository 환경을 생성해보겠습니다.

GitLab의 Personal projects에서 New project 버튼을 클릭하겠습니다.

03 Git을 활용하여 버전 관리를 해보자

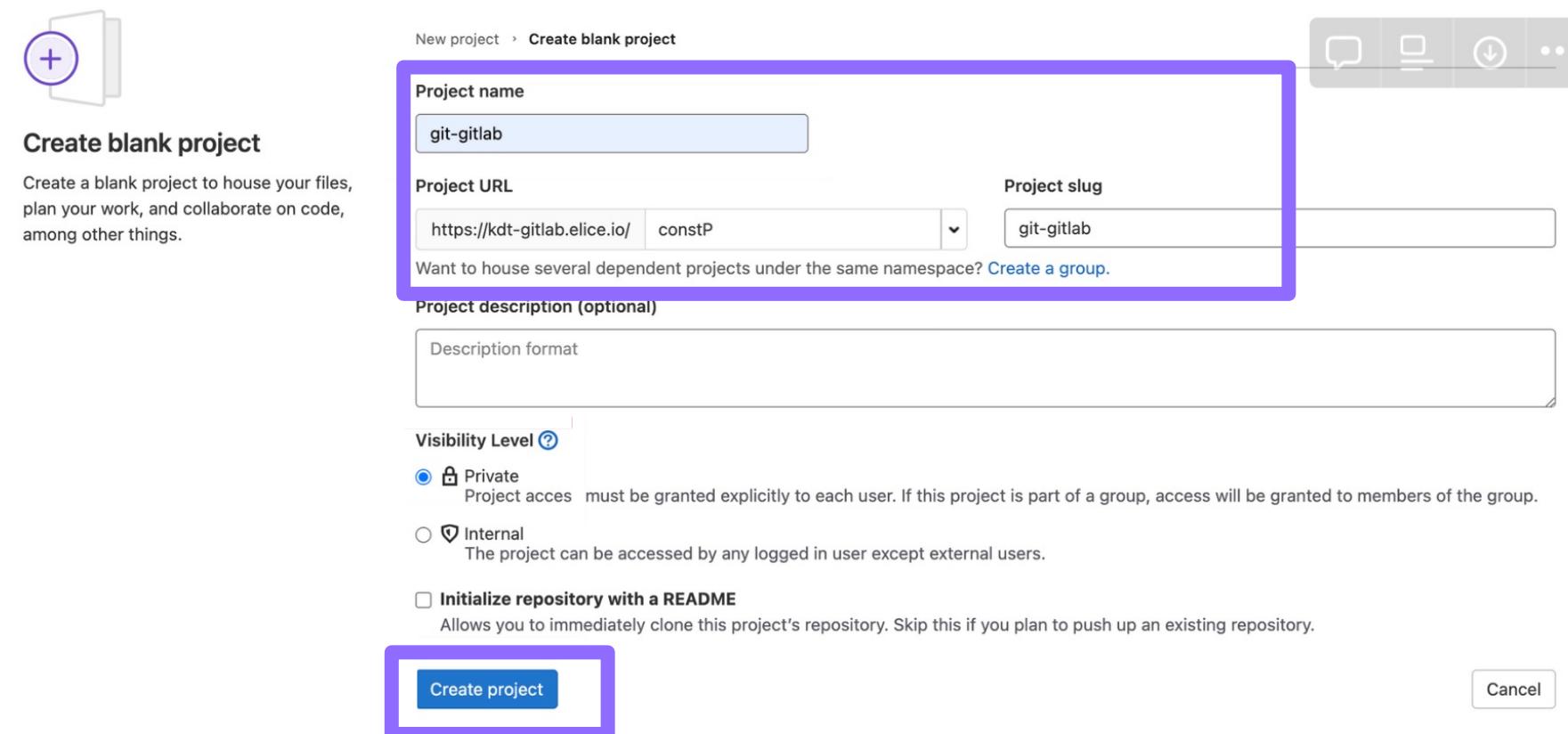
⑤ 원격 저장소 활용하기

그 후 Create blank project 버튼을 클릭하겠습니다.



03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기



project의 정보를 입력하고, Create project 버튼을 클릭합니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기

The screenshot shows the 'Project overview' page for a newly created project named 'git-gitlab'. A blue banner at the top indicates that the project was successfully created. The main content area displays the project's name, ID (5344), and a message stating that the repository is currently empty. It provides options to clone the repository or add files. Below this, there are sections for 'Command line instructions' and 'Git global setup', each containing command-line snippets. At the bottom, there is a section for creating a new repository with its own command-line instructions.

Project 'git-gitlab' was successfully created.

git-gitlab Project ID: 5344

The repository for this project is empty

You can get started by cloning the repository or start adding files to it with one of the following options.

Clone New file Add README Add LICENSE Add CHANGELOG Add CONTRIBUTING

Command line instructions

You can also upload existing files from your computer using the instructions below.

Git global setup

```
git config --global user.name "[코치] 박상수"  
git config --global user.email "tkdtn800@naver.com"
```

Create a new repository

```
git clone git@kdt-gitlab.elice.io:constP/git-gitlab.git  
cd git-gitlab  
touch README.md  
git add README.md
```

화면 처럼 나온다면 Repository가 성공적으로 생성된 것입니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기

Command line instructions

You can also upload existing files from your computer using the instructions below.

Git global setup

```
git config --global user.name "[코치] 박상수"  
git config --global user.email "tkdtn800@naver.com"
```

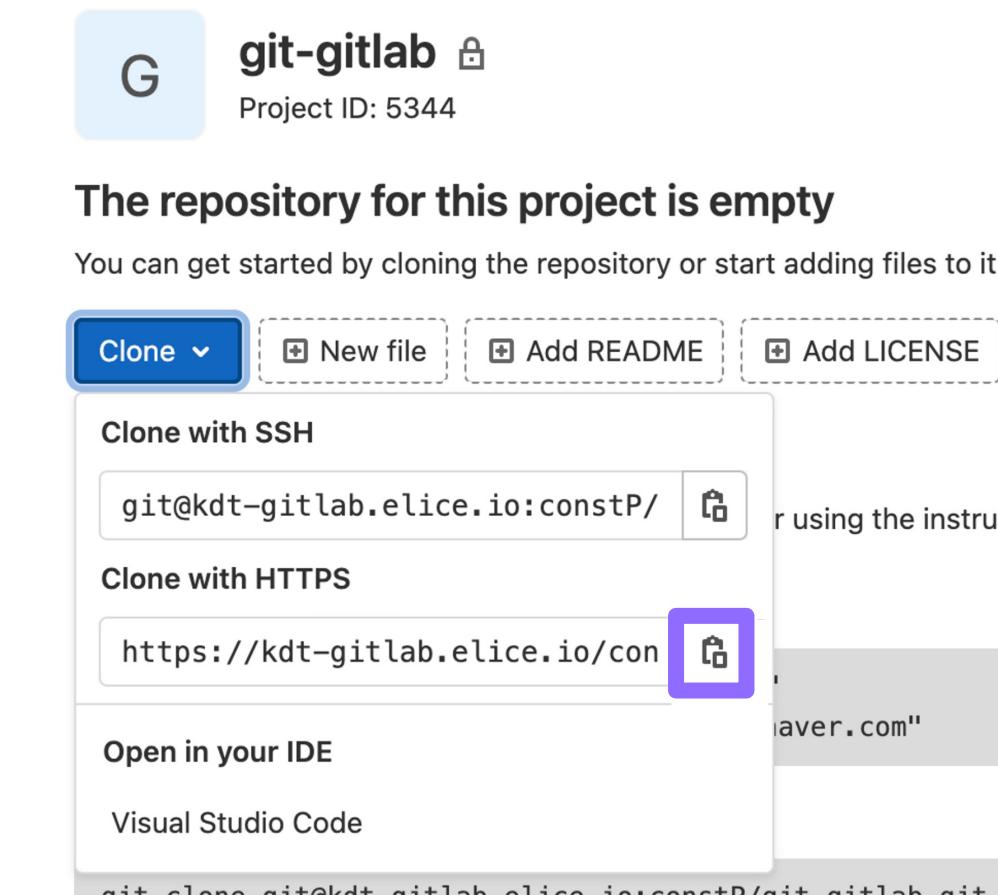
원격 저장소의 Repository를 clone 받기 위해

Repository에 나온 config 설정을 vscode 터미널에서 입력합니다.

여러분들이 생성한 Repository에 나온 코드를 터미널에 입력합니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기



본격적으로 원격 저장소의 Repository를 로컬 환경으로 clone 해오겠습니다.

먼저 Clone이라는 파란색 버튼을 클릭하고, Clone with HTTPS에 나와 있는 버튼을 클릭하여 Repository의 주소를 복사합니다.

⑤ 원격 저장소 활용하기



비밀 번호 입력

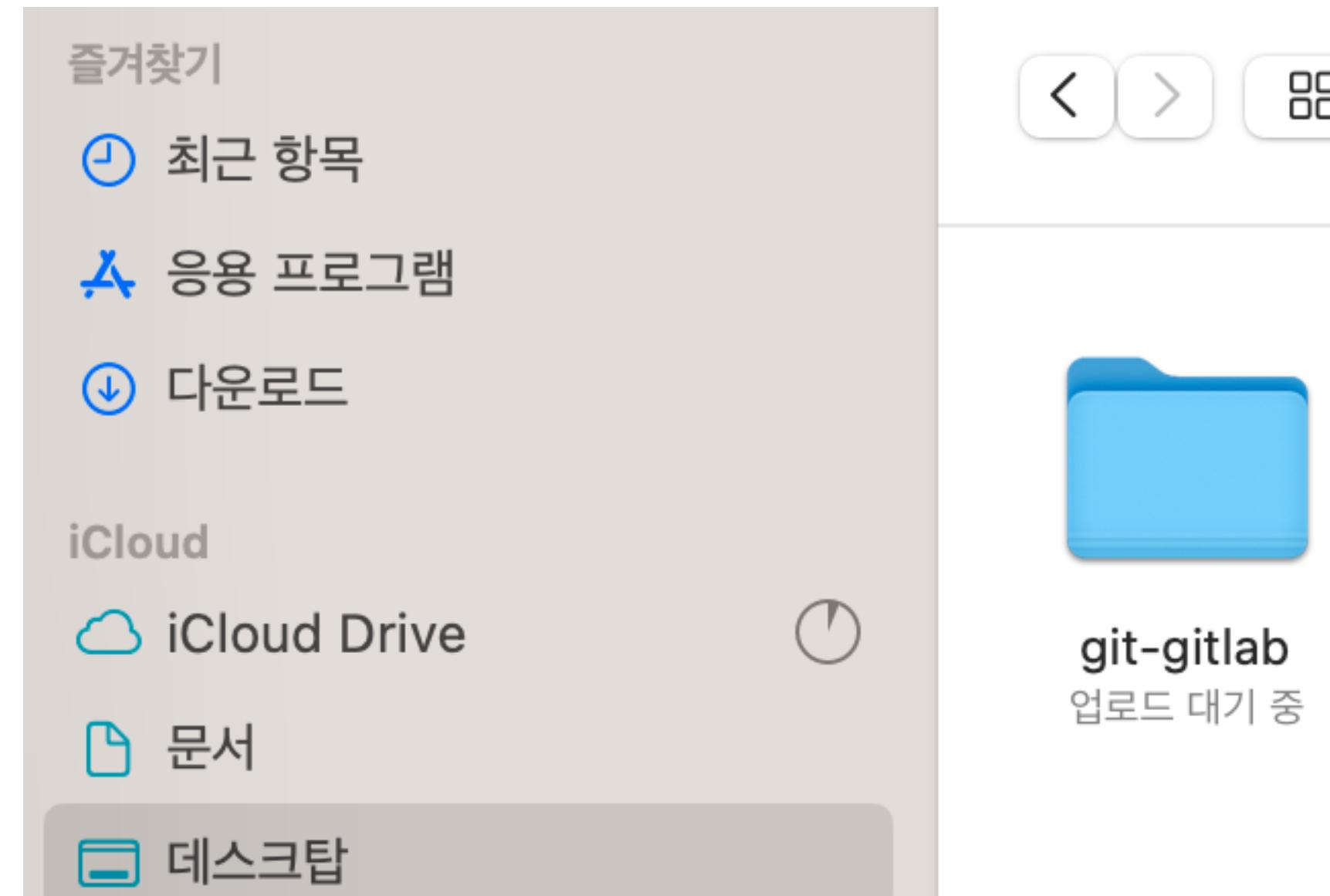
처음 clone을 받으면 비밀번호를 입력하라는 화면이 나올 수 있는데, 비밀번호는 여러분들의 gitlab 계정 비밀번호를 입력해주세요.

```
● parksangsu@bagsangsuum-MacBookPro ~/Desktop ➔ git clone https://kdt-gitlab.elice.io/constP/git-gitlab.git  
Cloning into 'git-gitlab'...  
warning: You appear to have cloned an empty repository.  
○ parksangsu@bagsangsuum-MacBookPro ~/Desktop ➔
```

그 후 vscode에서 프로젝트를 clone 받을 위치에 경로를 설정하고
git clone (복사한 내용)을 입력합니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기



실습에서 저는 데스크탑 경로로 Repository를 clone 받았기 때문에,
현재 데스크탑에 Repository가 존재하는 것을 볼 수 있습니다.

vscode에서 git-gitlab 폴더를 import 하겠습니다.



Repository

Repository는 버전이 저장되어 있는 공간입니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기

The screenshot shows a terminal window with four tabs at the top: 문제, 출력, 디버그 콘솔, and 터미널. The 터미널 tab is active, displaying the following command and its output:

```
● parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab ▶ master git remote -v  
origin https://kdt-gitlab.elice.io/constP/git-gitlab.git (fetch)  
origin https://kdt-gitlab.elice.io/constP/git-gitlab.git (push)  
○ parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab ▶ master
```

The output shows two entries for 'origin' with their respective URLs and status (fetch and push). The first entry is highlighted in blue, and the second is highlighted in green.

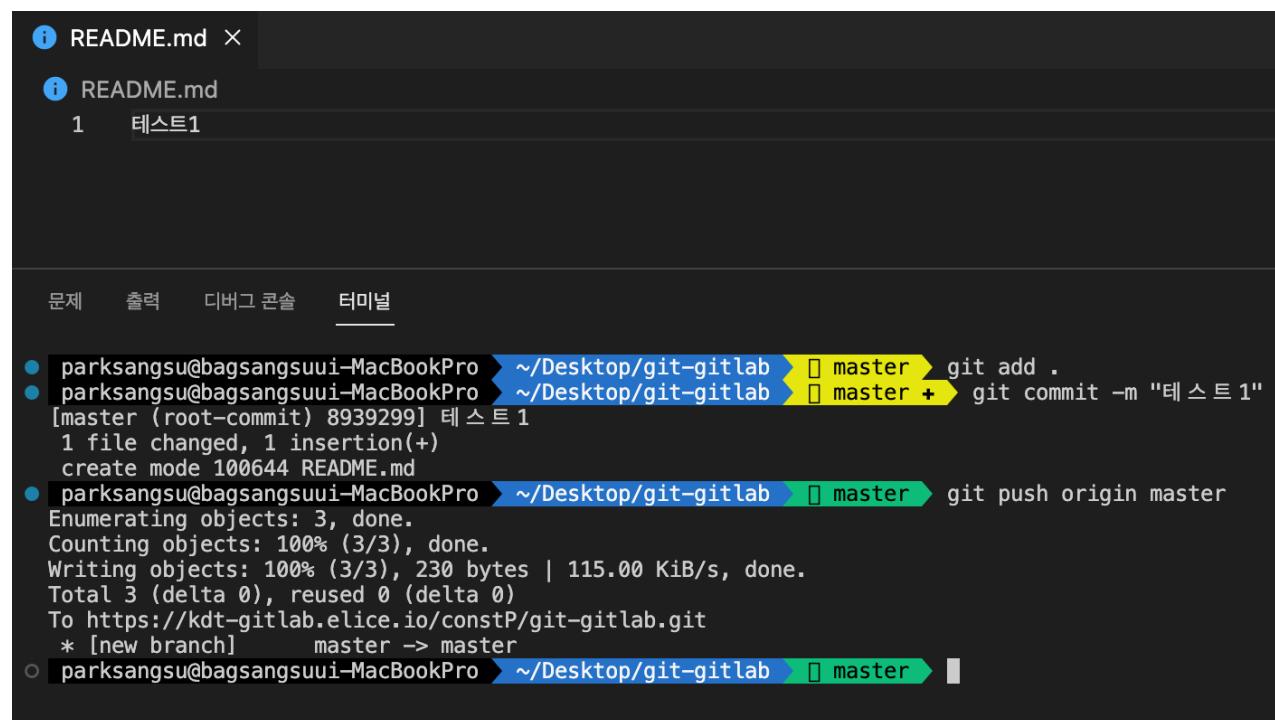
성공적으로 원격 저장소의 Repository를 clone 받았는지 확인하기 위해 git remote -v 명령어를 입력합니다.

명령어를 입력했을 때, origin이라는 원격 저장소의 이름이 출력되고,

옆에는 원격 저장소의 URL이 출력된다면 성공적으로 clone 받은 것입니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기



The screenshot shows a terminal window with a file named '테스트1' in the current directory. Below it, a git commit history is displayed:

```
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab master git add .
[parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab master +] git commit -m "테스트 1"
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 230 bytes | 115.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://kdt-gitlab.elice.io/constP/git-gitlab.git
 * [new branch] master -> master
```

원격 저장소에 내 로컬 저장소의 버전을 업로드해보겠습니다.

복습 차원에서 방금 배웠던 add, commit 명령어를 입력하여
버전을 생성해보겠습니다.

그 후 push 명령어를 입력하여 원격 저장소에 코드를 업로드하겠습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소 활용하기

The screenshot shows a GitLab repository page for a project named 'git-gitlab'. The top navigation bar includes 'History', 'Find file', 'Web IDE', and a download icon. A prominent blue button labeled 'Copy SSH clone URL' is visible. The main content area displays a single commit from a user named '테스트1' (Test1) made 25 seconds ago. The commit message is '[코치] 박상수 authored 25 seconds ago'. Below the commit, there are several buttons for managing files: 'README', 'Auto DevOps enabled', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', and 'Add Kubernetes cluster'. A table below lists the file 'README.md' with details: Name (README.md), Last commit (테스트1), and Last update (25 seconds ago). The file content is shown as '테스트1'.

원격 저장소에 성공적으로 버전이 업로드 됐는지 확인하기 위해 GitLab을 확인해보겠습니다.
생성했던 Repository를 확인해보면, 성공적으로 버전이 업로드 된 것을 볼 수 있습니다.

03 Git을 활용하여 버전 관리를 해보자

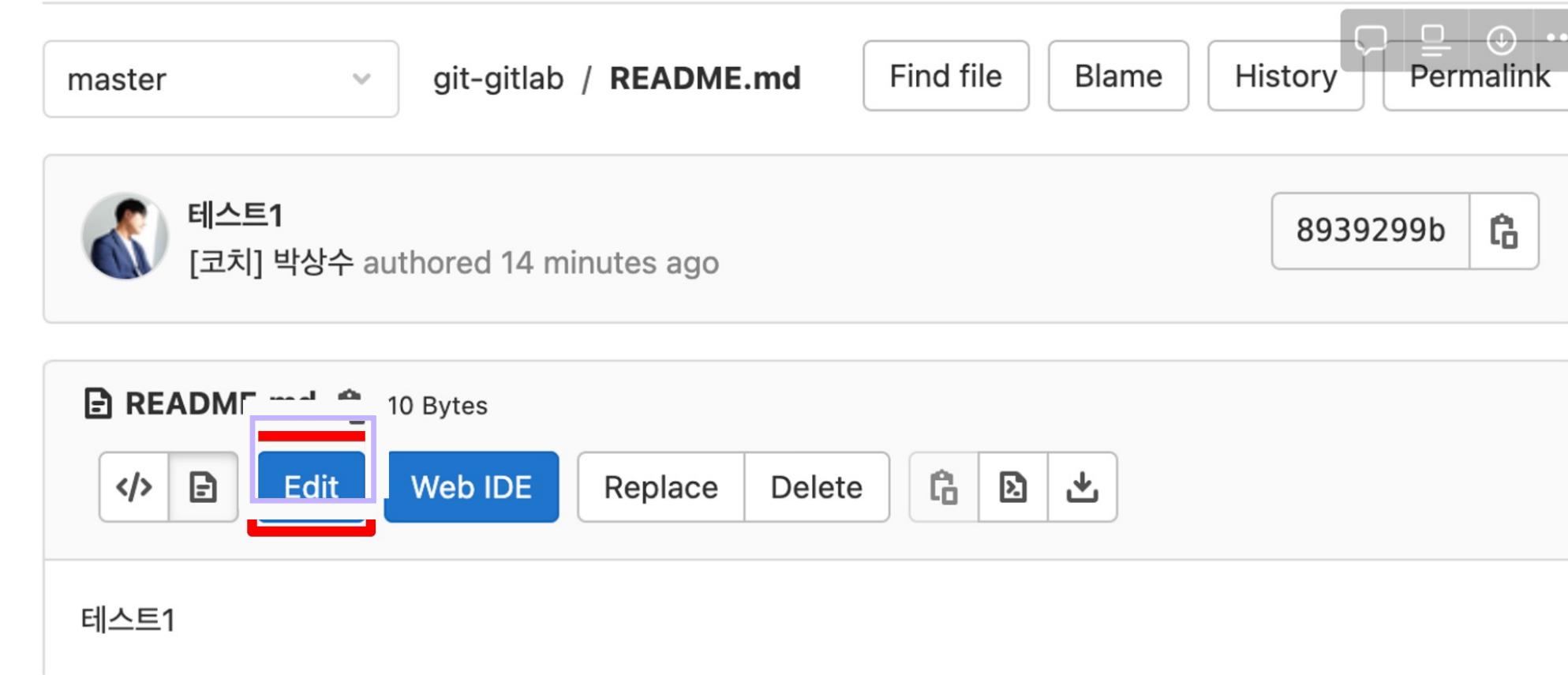
⑤ 원격 저장소의 변경 사항을 로컬로 가져오기

- 지금까지 내 로컬 환경에서 버전을 생성하고, 생성한 버전을 원격 저장소에 반영했습니다.
- 만약 협업하고 있는 다른 팀원이 코드를 변경해서 원격 저장소에 반영했다면,
원격 저장소의 변경 내역을 로컬 저장소로 가져오려면 어떻게 해야 할까요?
- 이 때 활용할 수 있는 명령어는 바로 git pull 명령어입니다.

지금부터 원격 저장소의 변경 사항을 로컬로 반영해보도록 하겠습니다.

03 Git을 활용하여 버전 관리를 해보자

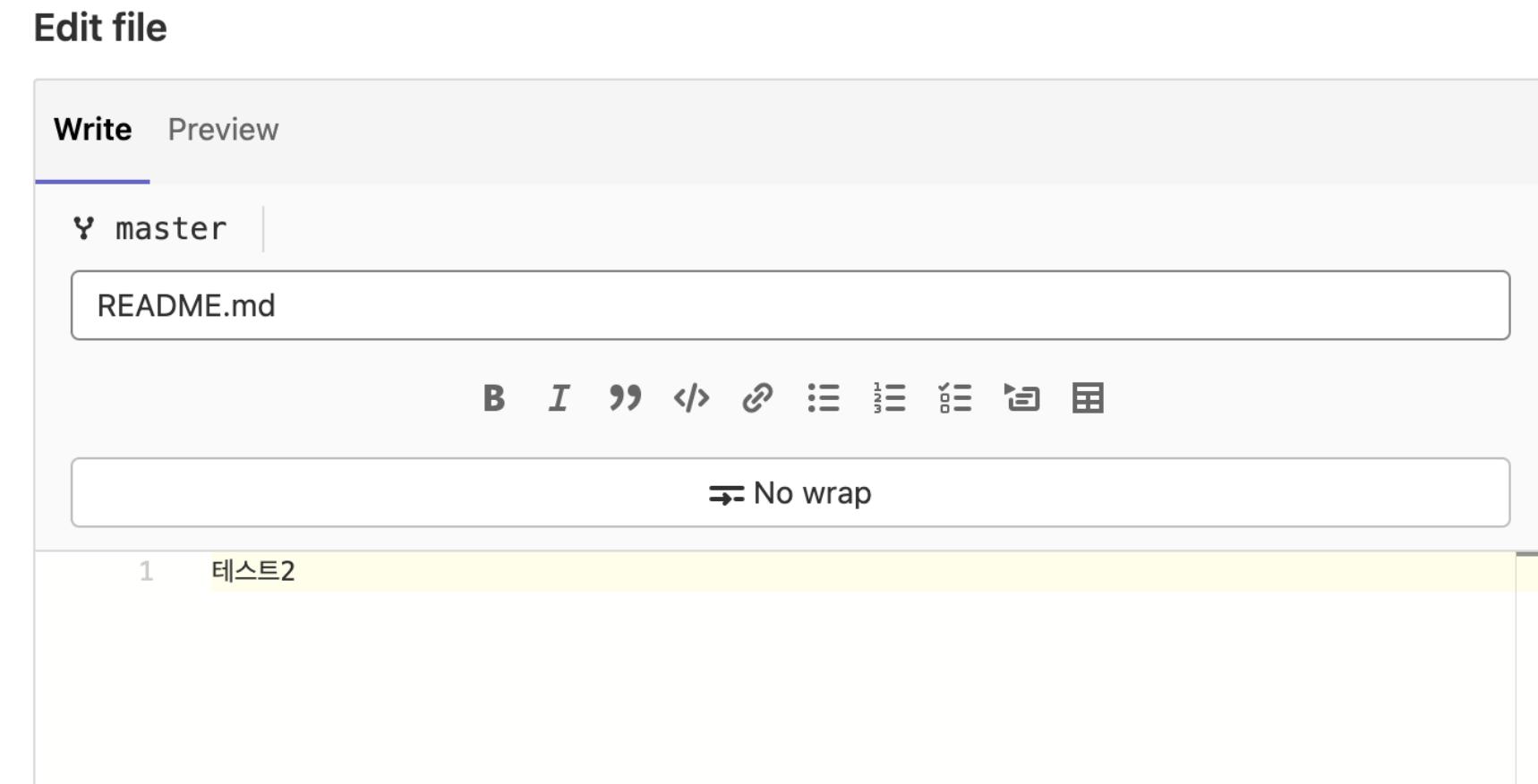
⑤ 원격 저장소의 변경 사항을 로컬로 가져오기



실습 구성을 위해 먼저 GitLab에 들어가서,
방금 생성했던 README.md 파일을 클릭하고, README.md의 Edit 버튼을 클릭합니다.

03 Git을 활용하여 버전 관리를 해보자

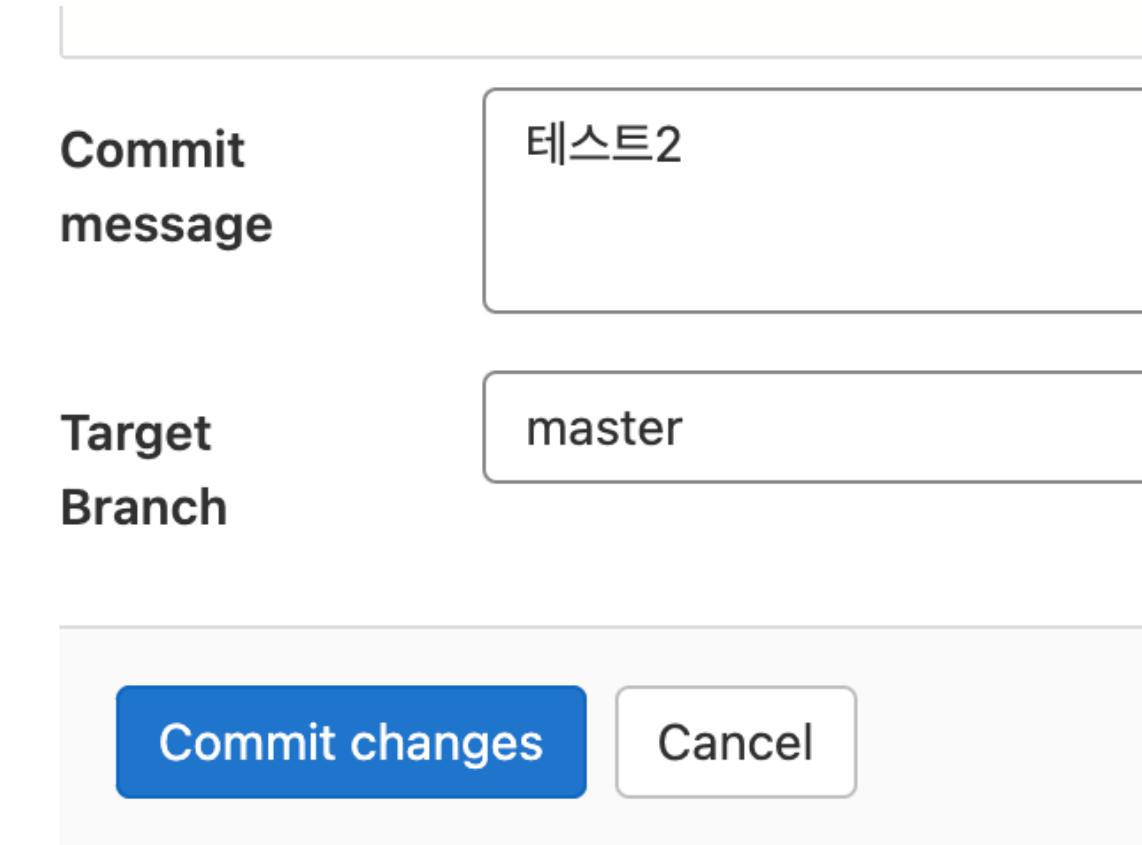
⑤ 원격 저장소의 변경 사항을 로컬로 가져오기



다른 개발자가 README.md 파일의 내용을 테스트2라고 입력하여
원격 저장소에 버전을 업로드했다고 가정하여 수정하겠습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소의 변경 사항을 로컬로 가져오기



README.md 파일의 내용을 변경했다면, 아래 Commit message, Target Branch를 설정해야 합니다.

Commit message는 테스트2라고 입력하고, Target Branch는 master를 선택한 후 Commit changes 버튼을 클릭합니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소의 변경 사항을 로컬로 가져오기

The screenshot shows a Git commit history page. At the top, there is a navigation bar with 'master' selected, a repository name 'git-gitlab / +', and buttons for 'History', 'Find file', 'Web IDE', and a download icon. A prominent blue button labeled 'Copy SSH clone URL' is also visible.

Below the navigation bar, a commit message is displayed: '테스트2 [코치] 박상수 authored just now'. To the right of the message is a commit ID '41532036' and a copy icon. A small profile picture of a person in a suit is next to the name.

Underneath the commit message, there is a table showing file details:

Name	Last commit	Last update
README.md	테스트2	just now

At the bottom of the table, there is a preview of the 'README.md' file content, which shows the text '테스트2'.

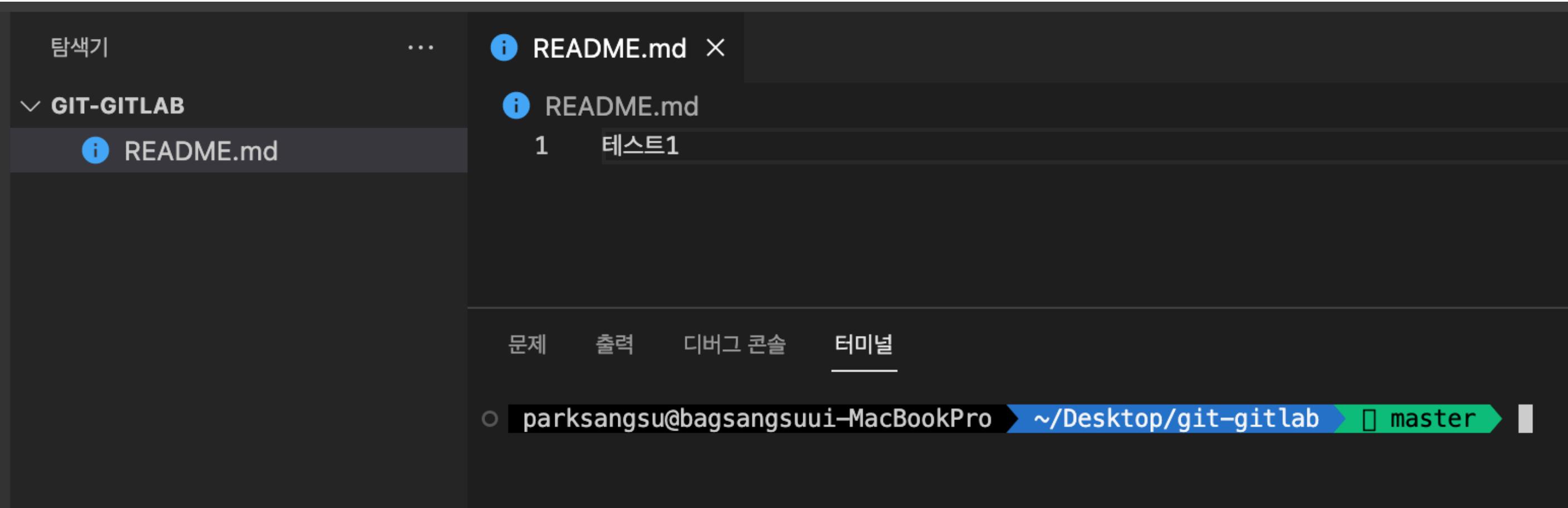
Commit changes를 입력했다면 성공적으로 README.md 파일이 테스트2라는 내용으로 수정된 것을 확인할 수 있습니다.

여기까지 완료됐다면, 협업하고 있는 다른 팀원이 원격 저장소에 코드를 변경한 상황이라고 생각할 수 있습니다.

그럼 다시 로컬 환경으로 돌아가보겠습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소의 변경 사항을 로컬로 가져오기



The screenshot shows a terminal window with the following content:

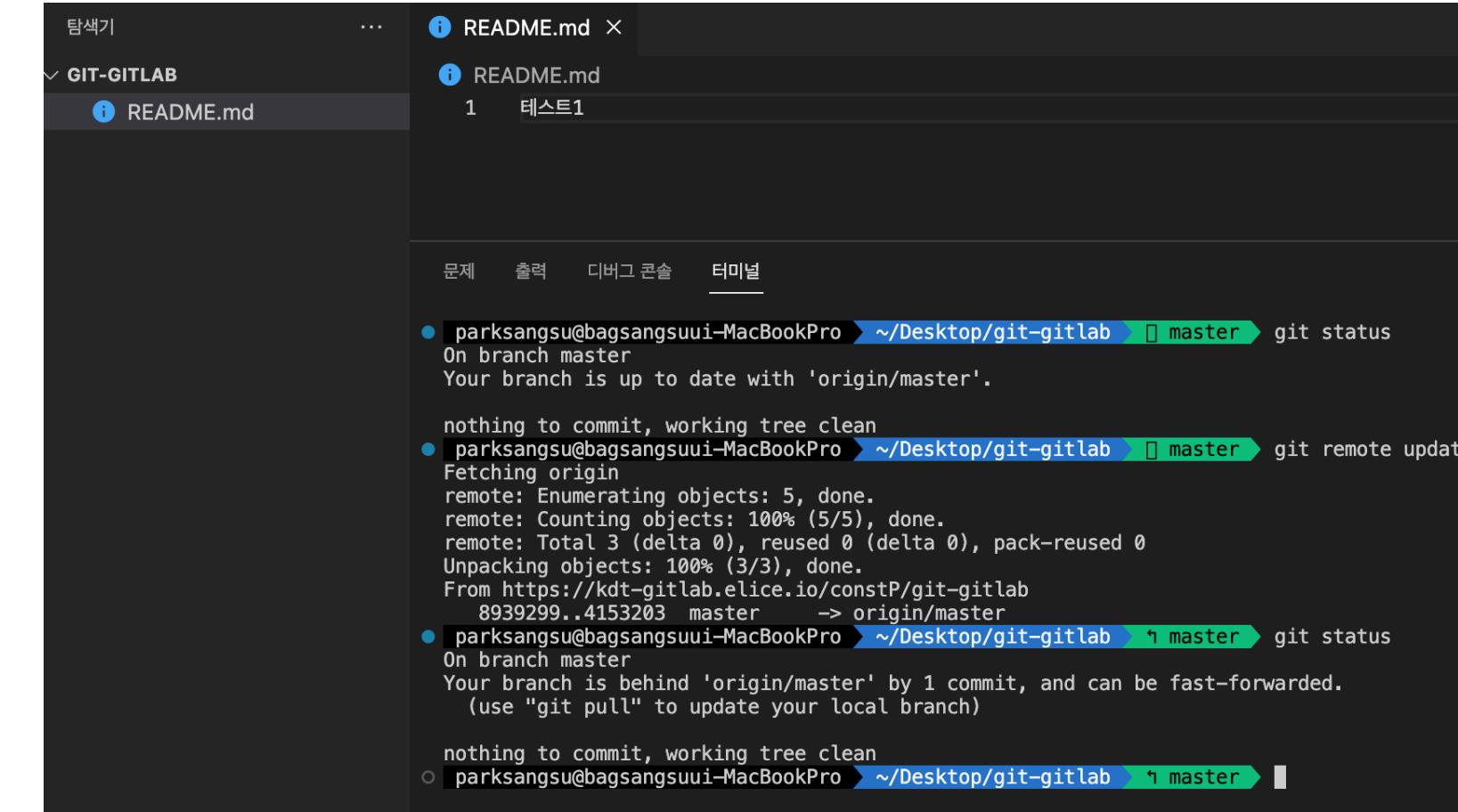
```
탐색기 ...  
└ GIT-GITLAB  
    README.md  
  
i README.md ×  
i README.md  
1 테스트1  
  
문제 출력 디버그 콘솔 터미널  
  
parksangsang@bagsangsangui-MacBookPro ~/Desktop/git-gitlab master
```

The terminal shows the output of a git pull command, indicating that the local repository has been updated from the remote repository. The local README.md file now contains the content "테스트1".

아직 로컬 환경에서의 README.md 파일은 테스트1로 구성되어 있습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소의 변경 사항을 로컬로 가져오기



The screenshot shows a terminal window with the following session:

```
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab master git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab master git remote update
Fetching origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://kdt-gitlab.elice.io/constP/git-gitlab
 8939299..4153203 master -> origin/master
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab master git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab master
```

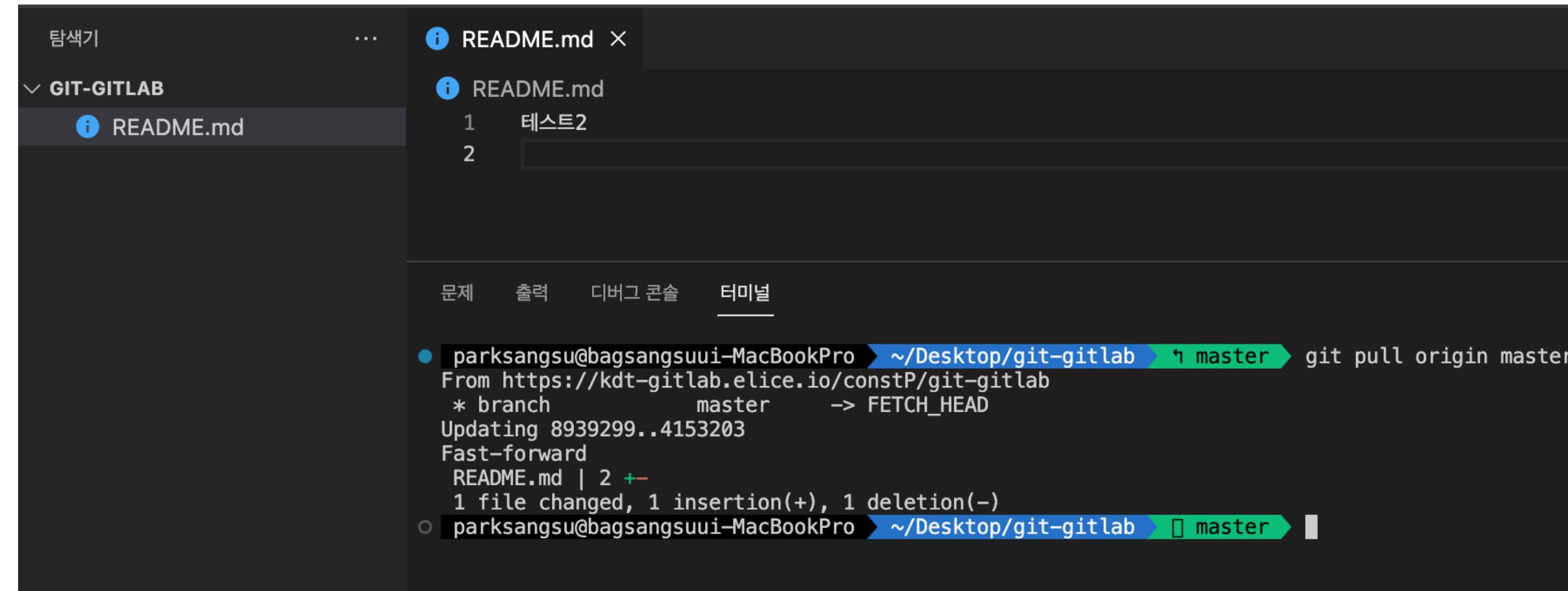
git status 명령어를 입력하면, 아직 commit 된 내역이 없다고 나오고 있습니다.

git remote update 명령어를 입력하여 현재 원격 저장소의 변경 사항 내역을 업데이트 해줍니다.

그리고 다시 git status 명령어를 입력해보면, origin/master에 commit이 하나 추가됐다는 내역이 나오고 있습니다.

03 Git을 활용하여 버전 관리를 해보자

⑤ 원격 저장소의 변경 사항을 로컬로 가져오기



The screenshot shows a terminal window with a dark theme. On the left, there's a file explorer sidebar with a single file named 'README.md' under a folder 'GIT-GITLAB'. The main area shows a code editor with the same 'README.md' file containing the text '테스트2'. Below the code editor is a tab bar with '문제', '출력', '디버그 콘솔', and '터미널', where '터미널' is selected. At the bottom, the terminal window displays the command 'git pull origin master' being run. The output shows the command being executed from a MacBook Pro at the path '~/Desktop/git-gitlab' on branch 'master'. It fetches changes from the remote 'origin' and performs a fast-forward merge, updating the file 'README.md' with one insertion and one deletion. The command concludes successfully.

```
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab * master git pull origin master
From https://kdt-gitlab.elice.io/constP/git-gitlab
 * branch            master      -> FETCH_HEAD
Updating 8939299..4153203
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
o parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab [master]
```

원격 저장소의 변경 사항을 로컬 환경으로 가져오기 위해 `git pull origin master` 명령어를 입력하겠습니다. `origin`은 `git remote -v` 명령어를 입력했을 때 원격 저장소로 설정한 이름이며, `master`는 원격 저장소에서 받아오고자 하는 Branch의 이름입니다. `Pull` 명령어를 입력함으로써 테스트2로 변경된 것을 볼 수 있습니다.

04

실무에서 필수! Git & GitLab 사용법

04 실무에서 필수! Git & GitLab 사용법

⑤ 다른 개발자들과 어떻게 함께 일할까?

- 지금까지 홀로 Git과 GitLab을 활용해서 버전 관리하는 방법에 대해 배웠습니다.
- 하지만 실무에서는 혼자가 아닌, 다른 개발자들과 함께 일하곤 하는데요,
다른 개발자들과 함께 코드의 버전을 어떻게 관리하곤 할까요?
- 협업에서 버전 관리하는 방식에 대해 알아보겠습니다.
- **.gitignore, Git Flow, MR, commit** 컨벤션 순서로 협업 방식에 대해 알아보겠습니다.

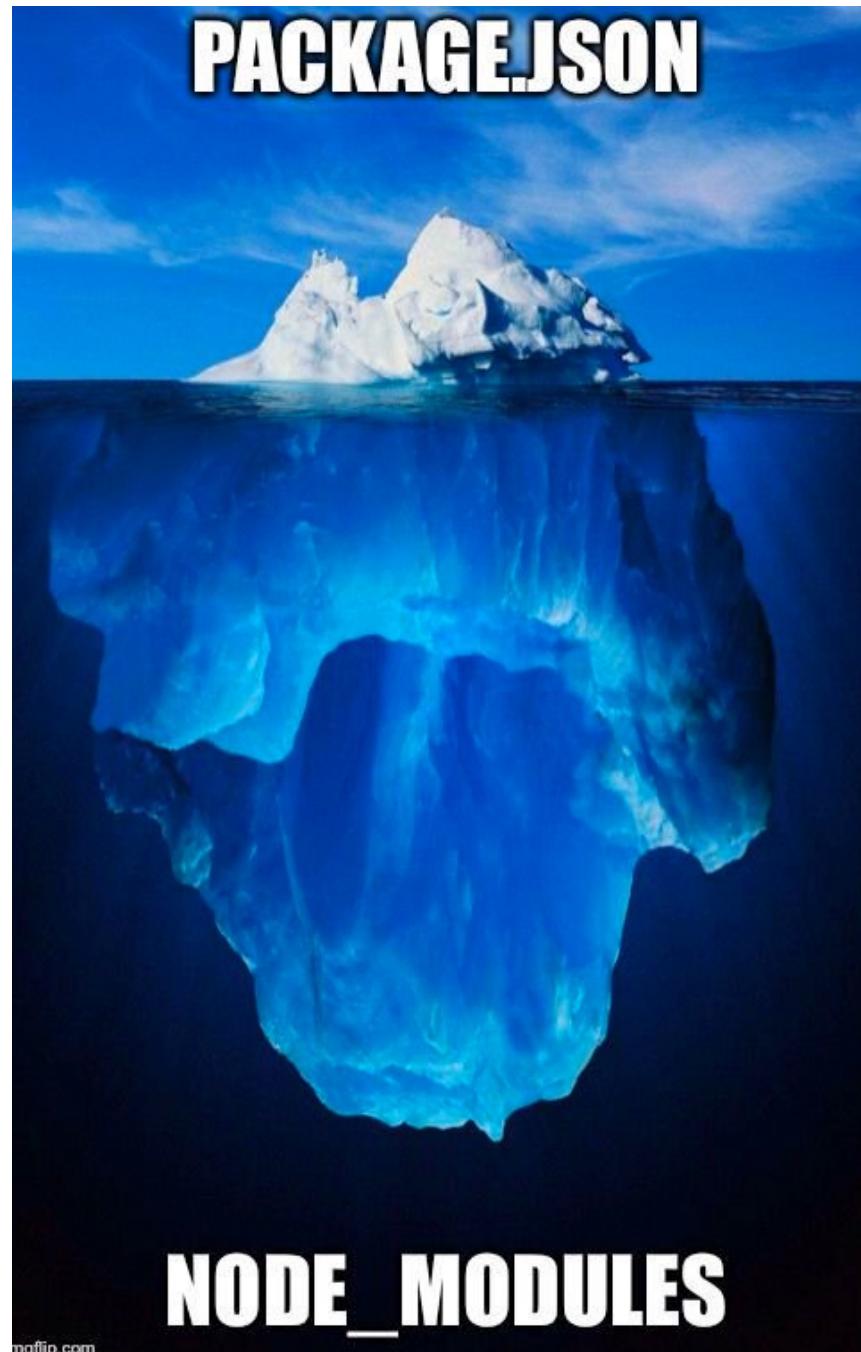
04 실무에서 필수! Git & GitLab 사용법

⑤ .gitignore를 활용하기

- gitignore란 Git 버전 관리 시스템에서 특정 파일이나 디렉터리를 버전 관리 대상에서 제외하는 기능을 제공합니다.
gitignore를 활용하여 Git 저장소에서 추적하지 않아도 되는 파일들을 지정할 수 있습니다.
- 일반적으로 개발 환경 설정 파일, 로그 파일, 빌드 결과물 등을 Git 추적 대상에서 제외합니다.
- 해당 파일들은 로컬에서만 활용하고, 다른 개발자들과 공유할 필요가 없습니다.
- gitignore 파일은 Git 저장소의 commit log에 기록되지 않으며, gitignore 파일을 추가하거나 수정해도 이미 Git 저장소에 올라간 파일들은 추척되기 때문에 주의해야 합니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ .gitignore를 활용하기



- node.js 환경에서 npm, yarn을 이용하여 개발한다면 package.json에 있는 의존성을 설치하다 보면 node_modules 폴더가 생성됩니다.
- node_modules 폴더는 상당한 용량을 차지하는데요. 이런 폴더는 로컬 저장소에만 저장하고, 원격 저장소에는 올릴 필요가 전혀 없습니다. 용량을 많이 차지하는 파일의 경우에도 원격 저장소에 올리지 말아야 합니다.
- 또한 환경 변수도 원격 저장소에 올릴 필요가 전혀 없습니다. 환경 변수에는 프로젝트 운영에 필요한 아주 민감한 정보들이 저장되어 있는데, 정보들이 원격 저장소에 업로드 된다면 어떤 문제가 발생할 수 있을까요?

04 실무에서 필수! Git & GitLab 사용법

⑤ .gitignore를 활용하기

```
| Email      | text    | YES   |      | NULL   |      |
+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> select * from WARNING
->;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'WARNING' at line 1
mysql> select * from WARNING
->;
+-----+
| id | warning
+-----+
| 1 | To recover your lost Database and avoid leaking it: Send us 0.06 Bitcoin (BTC) to our Bitcoin address 1BLYhUDmmnVPVjcTwgc6gFT6DCYwbVieUD and contact us by Email with your Server IP or
Domain name and a Proof of Payment. If you are unsure if we have your data, contact us and we will send you a proof. Your Database is downloaded and backed up on our servers. Backups that w
e have right now: wefish . If we dont receive your payment in the next 10 Days, we will make your database public or use them otherwise. | 1BLYhUDmmnVPVjcTwgc6gFT6DCYwbVieUD | contact@sql
db.to |
+-----+
```

만약 DB가 담겨 있는 환경 변수를 원격 저장소에 올린다면,
해커들이 여러분들이 올려둔 환경 변수를 활용해서 DB를 손쉽게 해킹할 수 있습니다.
비트코인을 주면 해킹한 정보를 살려주겠다는 아주 무시무시한 상황을 접할 수 있으니,
반드시 환경 변수는 gitignore를 설정해줘야 합니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ .gitignore를 활용하기



```
.gitignore
1 # compiled output
2 /dist
3 /node_modules
4
5 # env
6 *.env
7
```

실습을 진행하던 프로젝트에서
.gitignore 파일을 생성하고 코드를 작성하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ .gitignore를 활용하기

- 아래 있는 코드를 .gitignore 파일에 복사 붙여넣기 해줍니다.

/dist

/node_modules

*.env

04 실무에서 필수! Git & GitLab 사용법

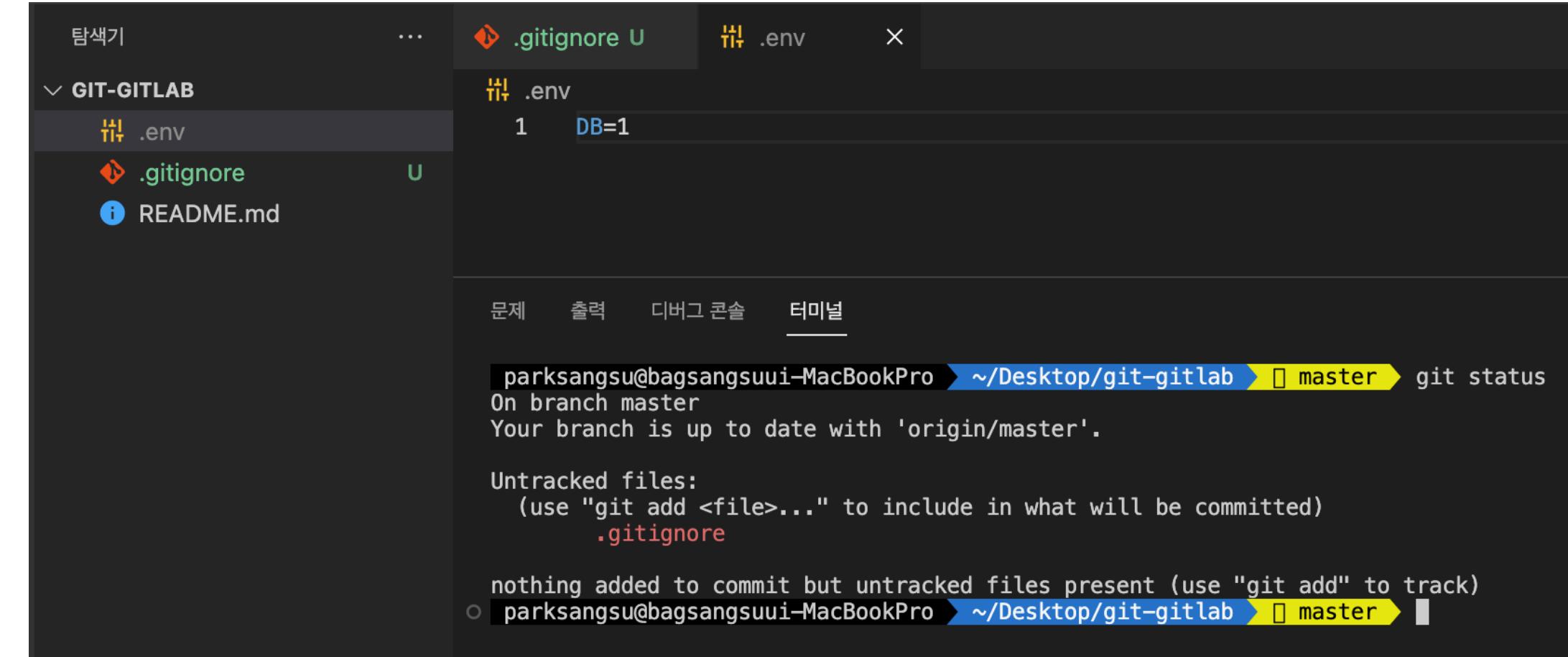
⑤ .gitignore를 활용하기



그 후 .env라는 환경 변수 파일을 생성하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ .gitignore를 활용하기



The screenshot shows a terminal window with a dark theme. On the left, there's a file tree with a folder named 'GIT-GITLAB' containing '.env', '.gitignore', and 'README.md'. The '.gitignore' file is currently selected. In the main pane, the contents of '.gitignore' are displayed: 'DB=1'. Below the code editor, there are tabs for '문제', '출력', '디버그 콘솔', and '터미널'. The '터미널' tab is active, showing the output of the 'git status' command:

```
parksangsu@bagsangsuui-MacBookPro ~/Desktop/git-gitlab master git status
On branch master
Your branch is up to date with 'origin/master'.

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

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

gitignore를 활용하여 특정 파일이 git 추적 대상에서 제외됐는지 확인하기 위해 git status 명령어를 입력해보겠습니다.

입력해보면, .env 파일은 추적 대상에서 제외됐고, .gitignore 파일만 추적 대상으로 포함된 것을 확인할 수 있습니다.

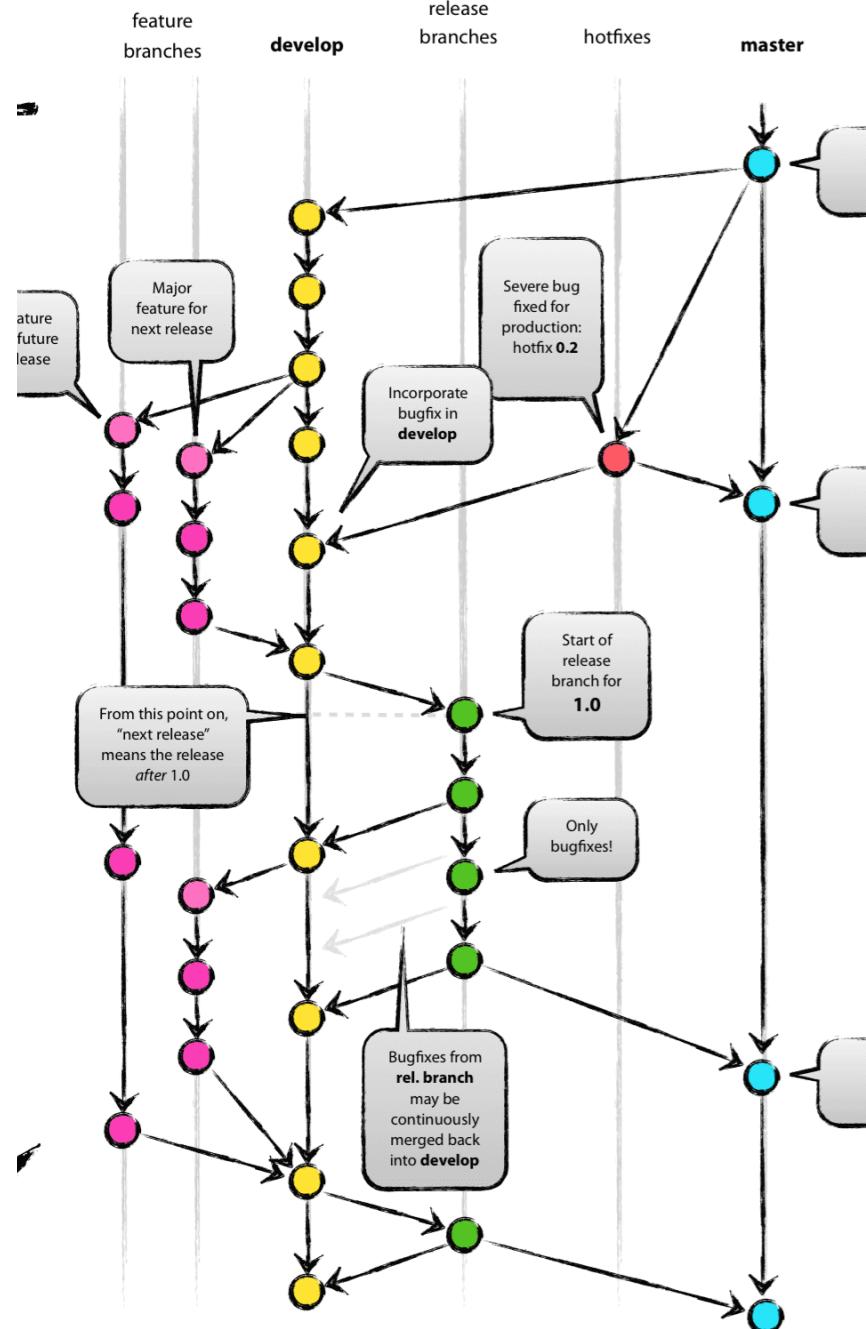
04 실무에서 필수! Git & GitLab 사용법

⑤ Git Flow

- 다양한 개발자와 협업할 때, 몇 가지 규칙을 정해서 버전 관리를 하곤 합니다.
- 이번 시간에는 Git Branch 전략을 활용하여 버전 관리하는 방법에 대해 알아보겠습니다.
- Git Branch 전략에는 Git-Flow, GitLab-Flow, Github-Flow 등이 사용되는데,
그 중 많은 기업에서 사용하는 Git Branch 전략인 Git-Flow에 대해 알아보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

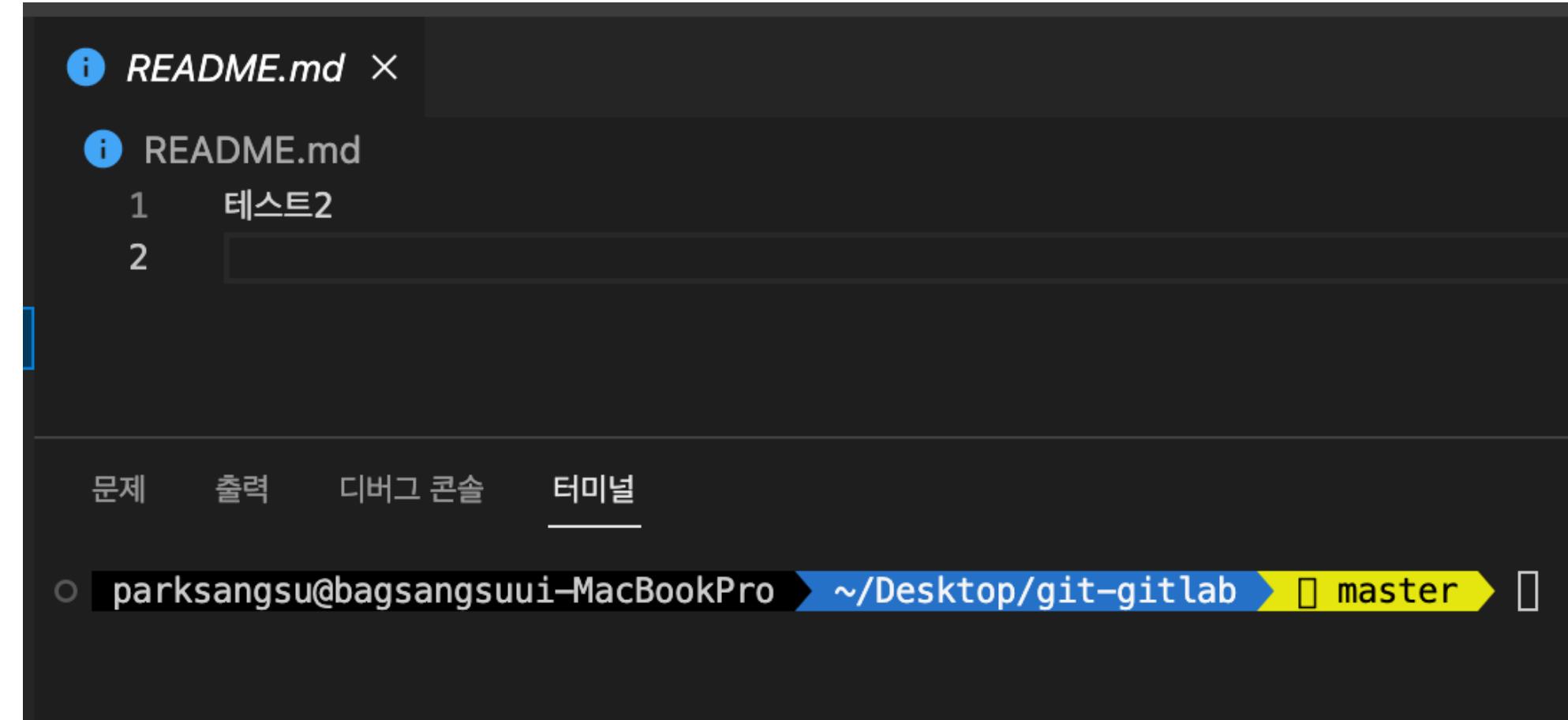
Git Flow



- Git-Flow는 Branch를 크게 3~4가지로 나눠 개발하는 버전 관리 전략입니다.
- Git-Flow를 활용하면 Branch 별로 역할과 책임을 나눠서 버전 관리를 할 수 있습니다.
- 방금 전까지 사용한 branch는 무엇이었는지 알아보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Git Flow



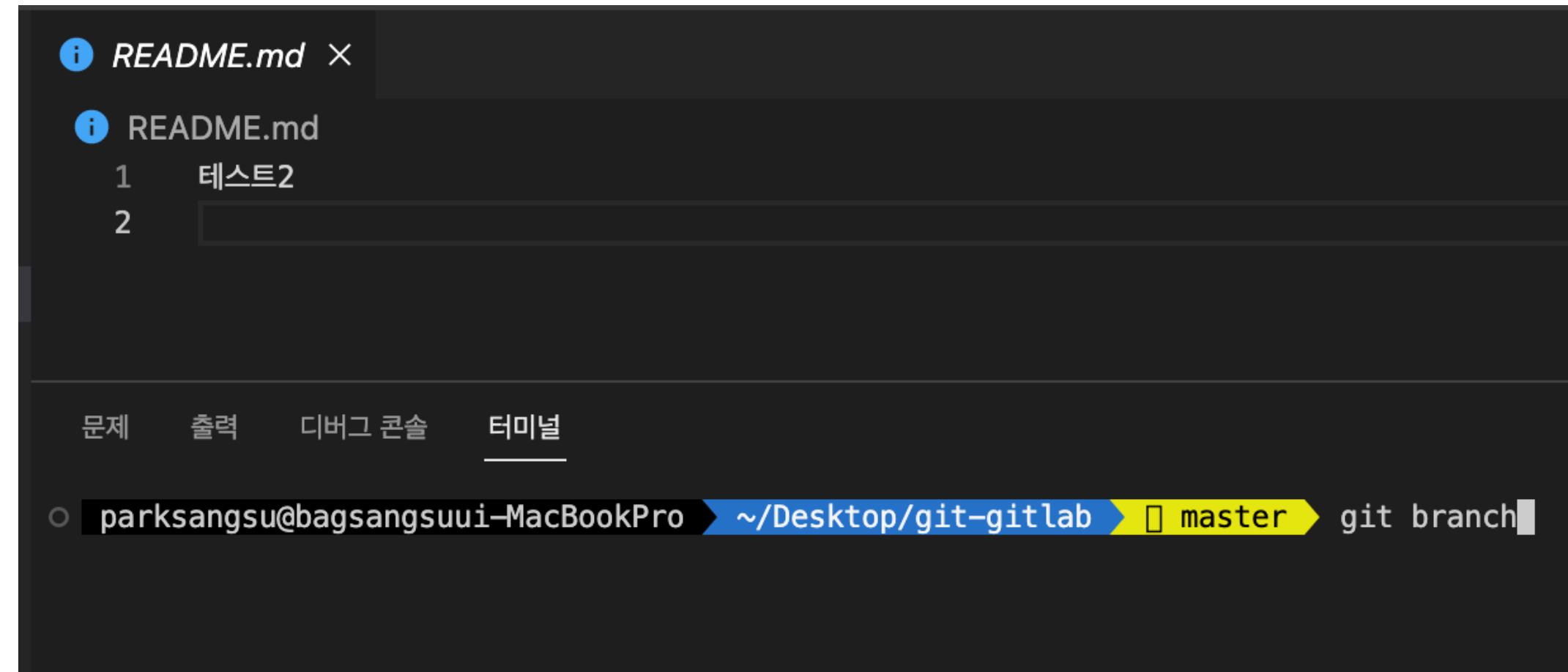
```
i README.md X
i README.md
1 테스트2
2

문제    출력    디버그 콘솔    터미널
○ parksangs@bagsangsuui-MacBookPro ~/Desktop/git-gitlab master ▶
```

현재까지는 master Branch 하나로 모든 작업을 해왔습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Git Flow

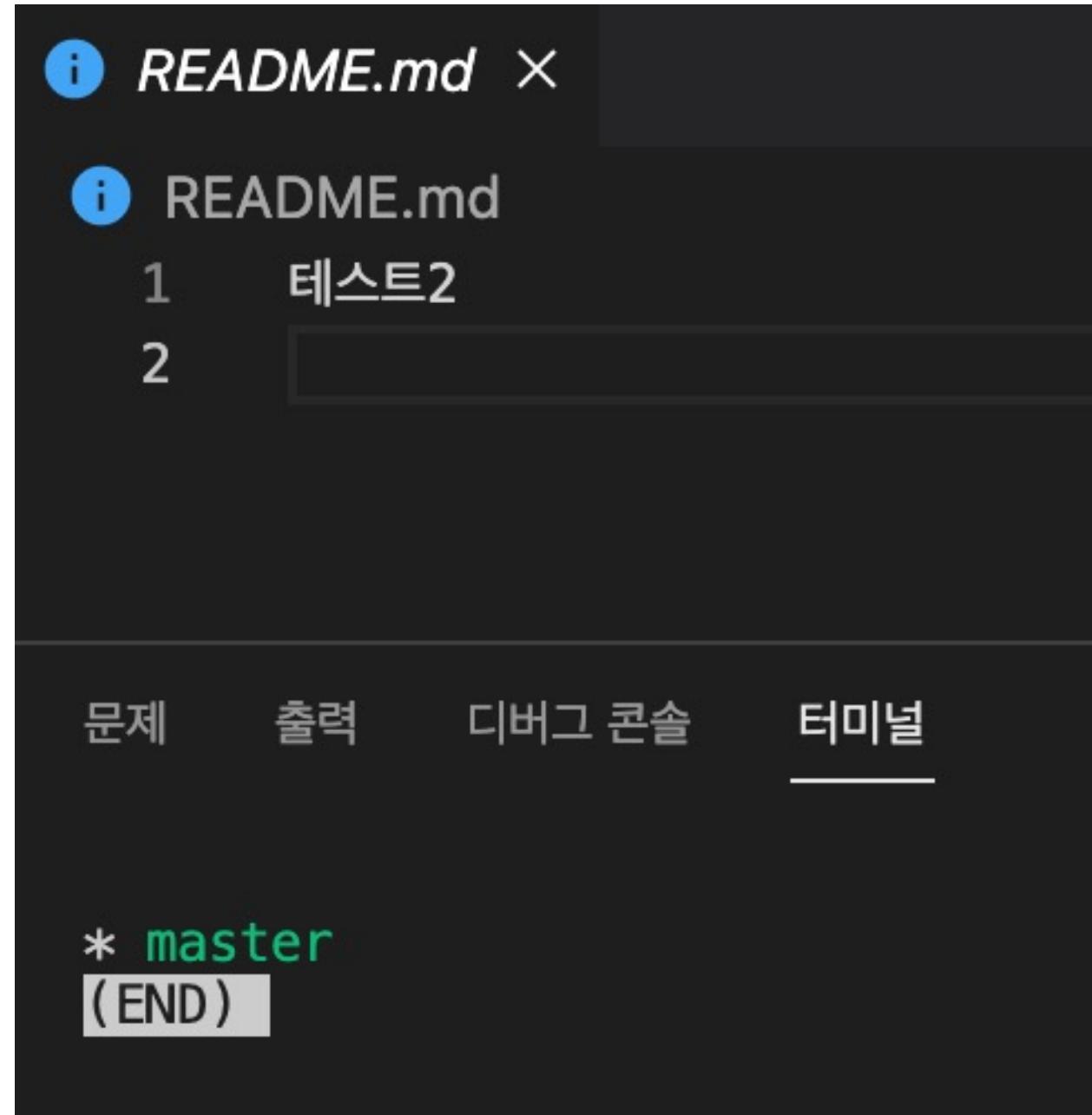


A screenshot of a terminal window with a dark background. At the top, there are two tabs: 'README.md' (active) and 'README.md'. The main area shows the content of the README.md file, which contains two lines of Korean text: '1 테스트2' and '2'. Below the file content, there is a horizontal navigation bar with four items: '문제', '출력', '디버그 콘솔', and '터미널', with '터미널' being underlined. At the bottom, there is a command-line interface (CLI) showing the user's session: 'parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab master git branch'.

현재 로컬 환경에서 어떤 branch가 있는지 확인하기 위해 git branch 명령어를 입력해보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Git Flow



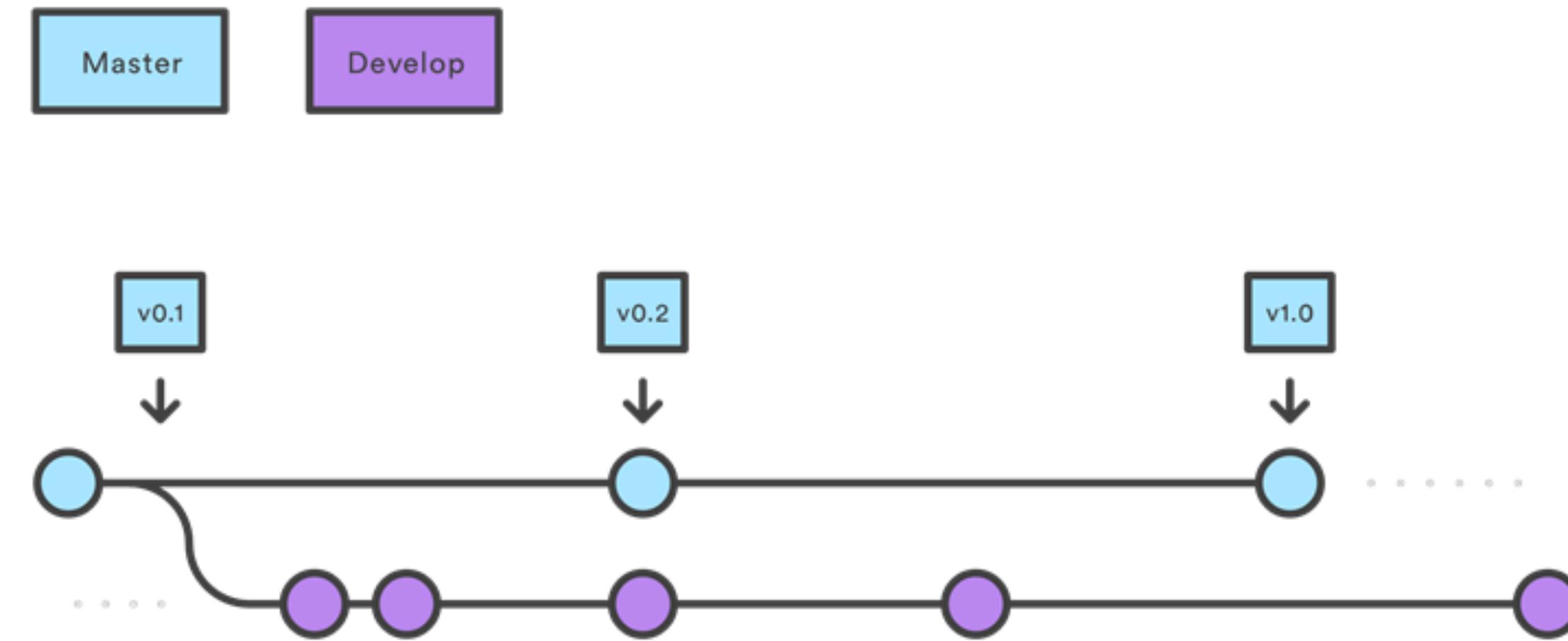
현재는 master branch만 존재하는 것을 볼 수 있습니다.

Git-Flow에서는 master branch 뿐 아니라 develop branch가 존재하고, feature, hot fix, release라는 필요에 따라 생성하는 branch가 있습니다.

그럼 이제 각각의 branch에 대해 알아보고
어떻게 활용할 수 있는지 알아보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ 메인 Branch



Git-Flow에서는 메인 Branch를 구성해야 합니다.

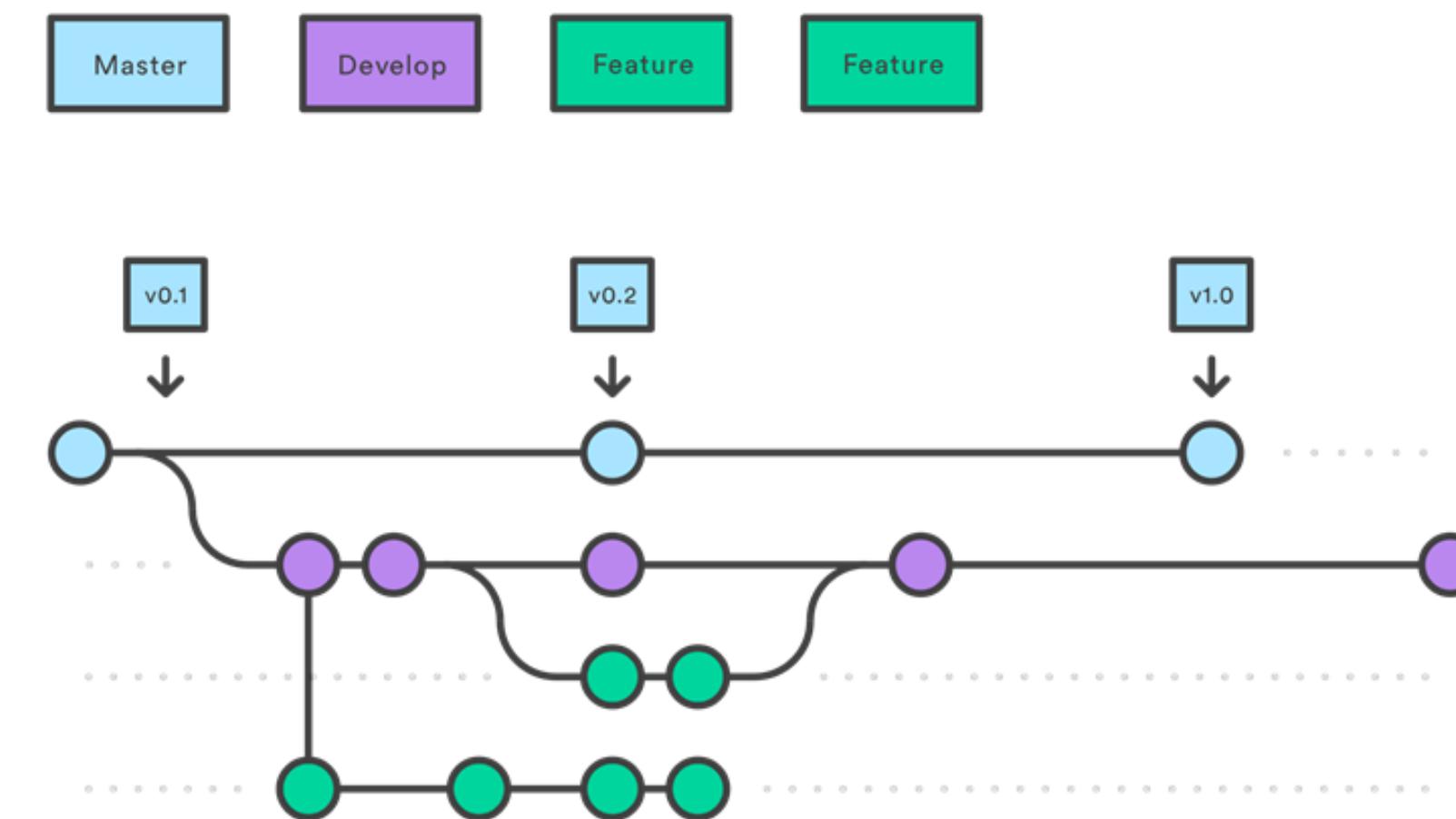
메인 Branch는 master Branch와 develop Branch 두 종류를 말합니다.

Master Branch는 배포 가능한 상태만을 관리하는 Branch입니다.

Develop Branch는 다음 배포할 것을 개발하는 Branch입니다.

04 실무에서 필수! Git & GitLab 사용법

✓ 보조 Branch



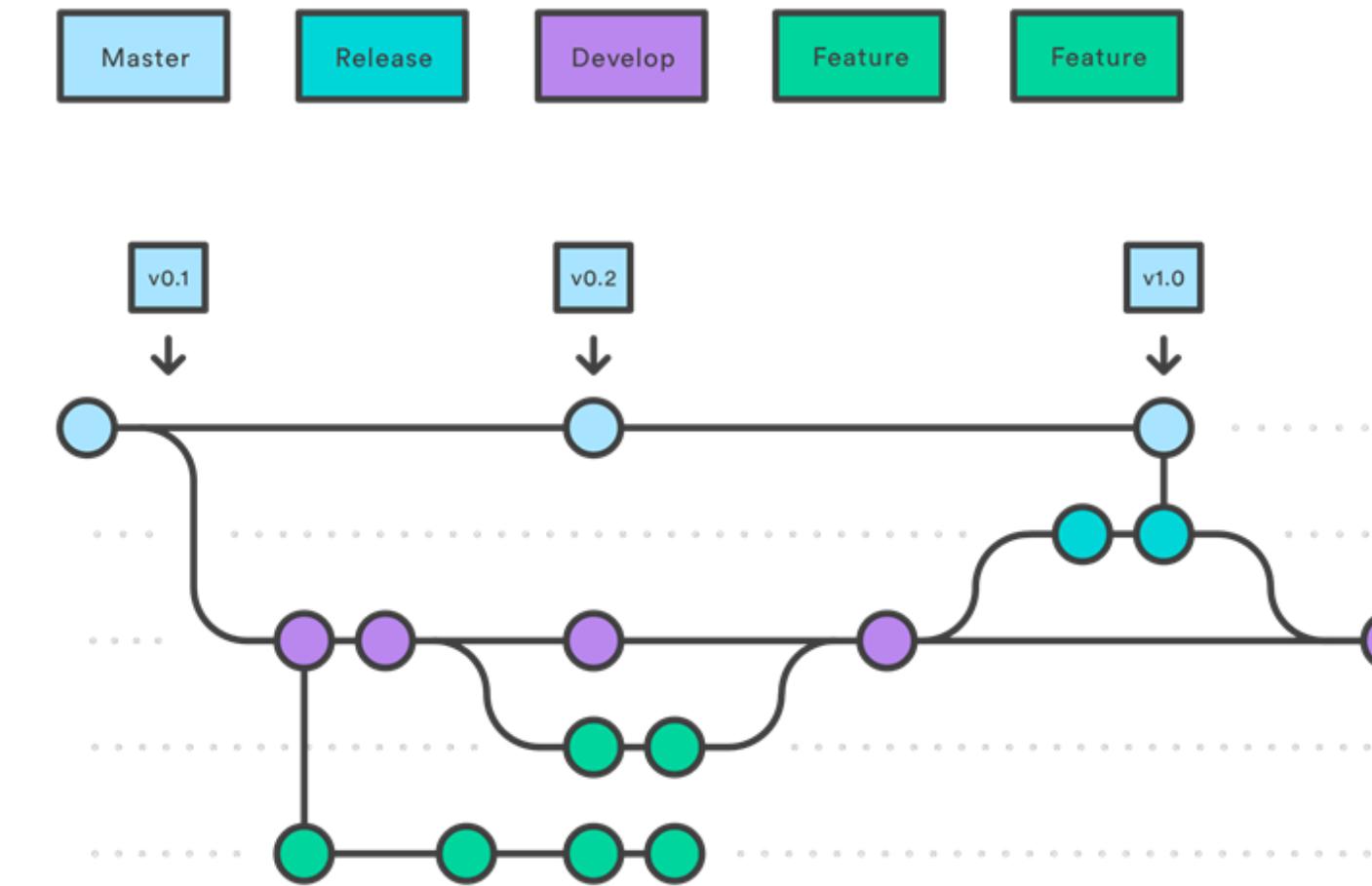
보조 Branch는 feature Branch를 말합니다. Master Branch에서 develop Branch를 만들었고,

develop Branch에서 다시 feature Branch를 나눠 작업합니다. **Feature Branch는 기능을 개발하는 Branch입니다.**

기능을 다 완성할 때까지 유지하고, 완성되면 develop Branch로 merge합니다.

04 실무에서 필수! Git & GitLab 사용법

✓ 릴리즈 Branch



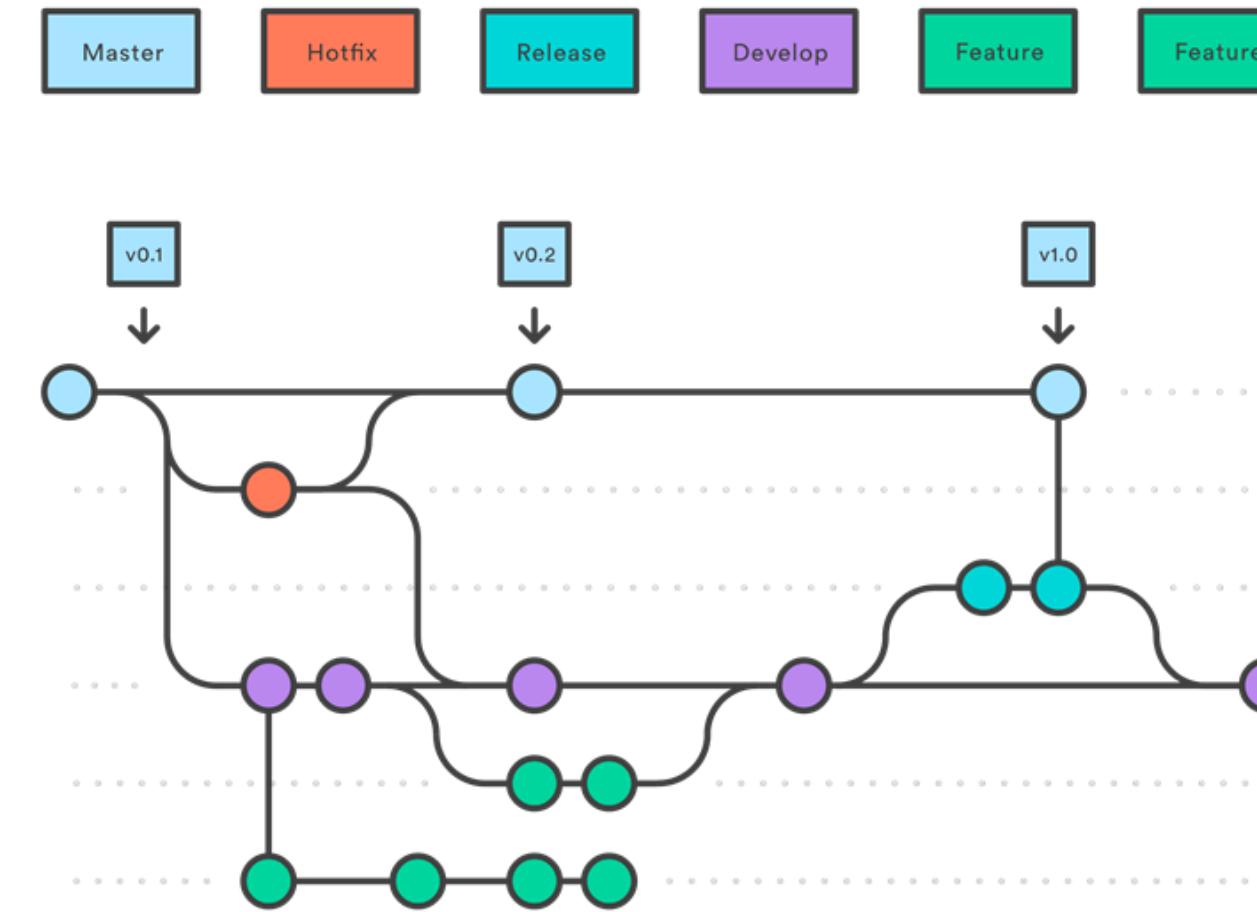
릴리즈 Branch는 배포를 위한 최종적인 버그 수정 등의 개발을 수행하는 Branch를 말합니다.

Develop Branch에 버전에 포함되는 기능이 Merge 되었다면

QA를 위해 Develop Branch에서부터 Release Branch를 생성합니다.

04 실무에서 필수! Git & GitLab 사용법

✓ 핫픽스 Branch



핫픽스 Branch는 배포한 버전에서 긴급하게 수정할 필요가 있을 때 master Branch에서 분리하는 Branch입니다.

핫픽스 Branch에서 변경 사항은 develop Branch에도 merge하여 문제가 되는 부분을 함께 반영해줘야 합니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Branch 생성 및 활용 방법

```
git branch 브랜치이름 (해당 브랜치가 존재하지 않는다면 브랜치를 새로 만듭니다.)  
ex) git branch feature/#1
```

```
git checkout 브랜치이름 (존재하는 브랜치가 있다면 그 브랜치로 이동합니다.)  
ex) git checkout feature/#1
```

```
git checkout -b 브랜치 이름 (해당 브랜치가 존재하지 않는다면 브랜치를 새로 만들면서 바로 그 브랜치로 이동합니다.)  
ex) git checkout -b feature/#1
```

Git Flow에 맞게 Branch를 나누고, 나눈 Branch로 이동하는 방법을 알아보겠습니다.

먼저 branch 생성을 위해 git branch 명령어를 입력해줍니다. feature/#1이라는 branch가 생성됩니다.

그 후 git checkout feature/#1을 입력하면, 현재의 버전은 feature/#1이 됩니다. 만약 Branch를 생성하고, 해당 Branch로 이동하는 것을 한 줄로 하고 싶다면 git checkout -b feature/#1 과 같이 입력하면 됩니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ MR을 활용하여 다른 branch에 내 코드를 반영해보자.

- 지금까지 Git-Flow에 대해 알아봤습니다.
- 그럼 Git-Flow를 이용하여 Branch를 분리하여 원격 저장소에 Branch를 push해보겠습니다.
- 그리고 feature Branch와 develop Branch의 코드를 MR를 이용하여 병합하는 과정을 실습을 통해 알아보도록 하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Develop Branch 생성

The screenshot shows a terminal window with a dark theme. On the left, there's a file explorer sidebar titled '탐색기' (Explorer) showing a folder named 'GIT-GITLAB' containing files '.env', '.gitignore', and 'README.md'. The 'README.md' file is selected. The main area is a terminal window with the following content:

```
문제 출력 디버그 콘솔 터미널
● parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab master git checkout -b develop
Switched to a new branch 'develop'
○ parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab develop
```

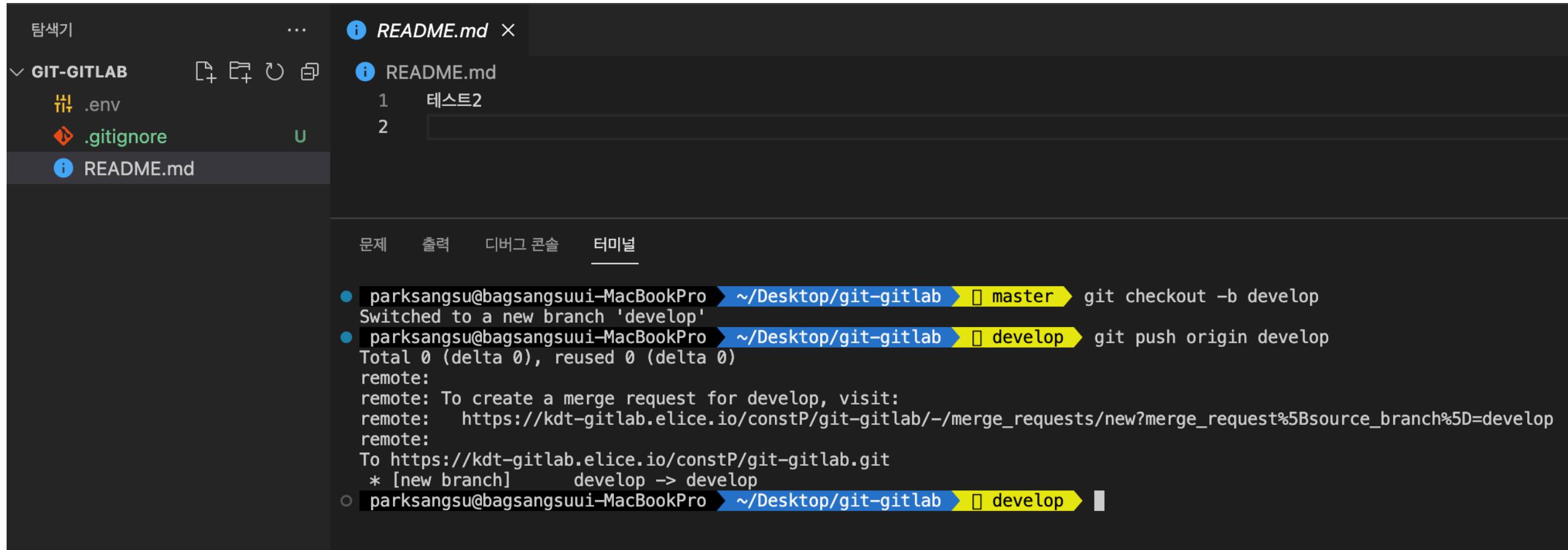
The terminal shows the command 'git checkout -b develop' being run, switching the branch to 'develop'.

현재 master Branch에 위치하고 있습니다.

develop Branch를 생성하고, develop Branch로 이동해보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ MR을 활용하여 다른 branch에 내 코드를 반영해보자.



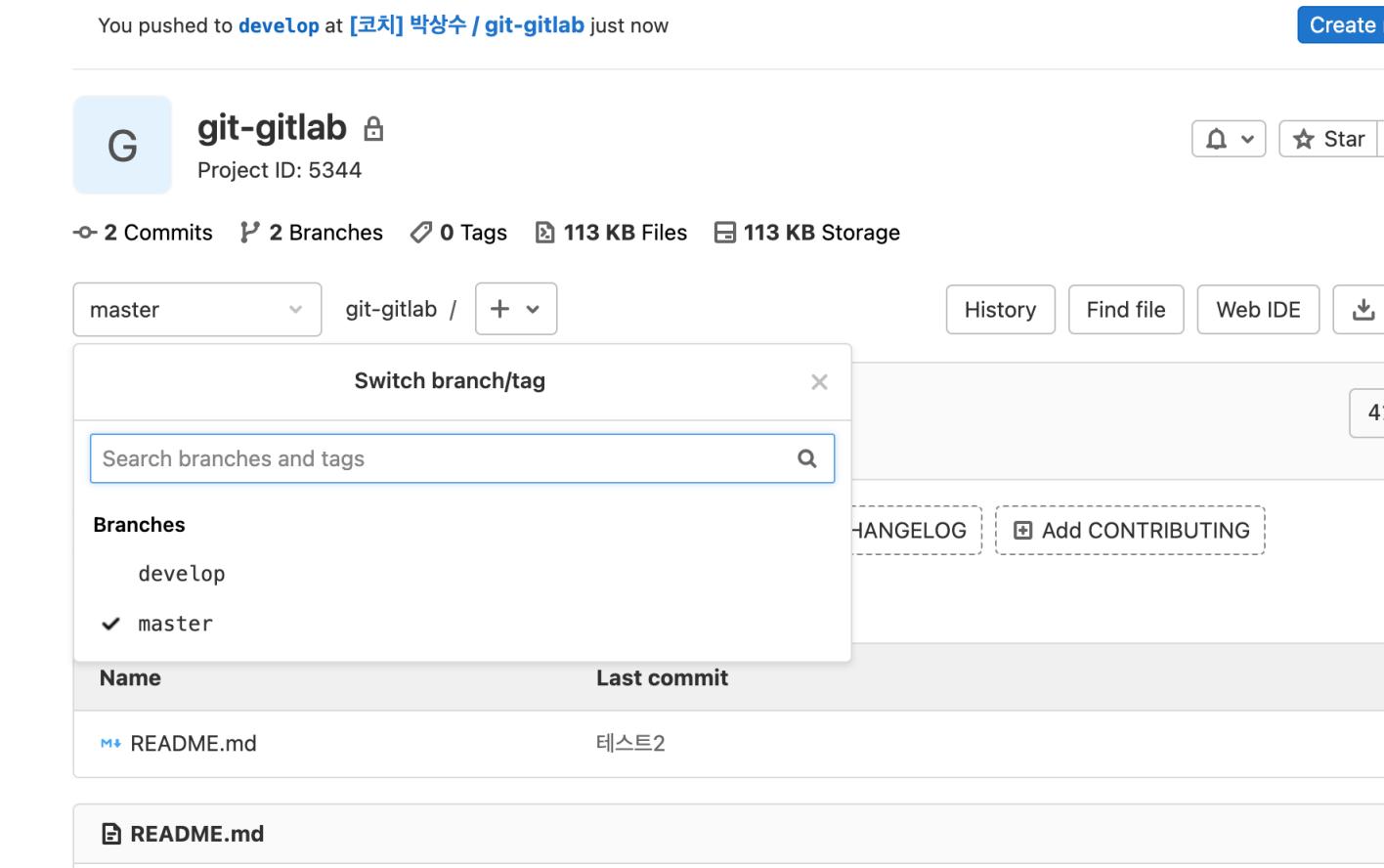
The screenshot shows a terminal window with the following command history:

- parksangsu@bagsangsuumi-MacBookPro ~/Desktop/git-gitlab master git checkout -b develop
- Switched to a new branch 'develop'
- parksangsu@bagsangsuumi-MacBookPro ~/Desktop/git-gitlab develop git push origin develop
- Total 0 (delta 0), reused 0 (delta 0)
- remote:
- remote: To create a merge request for develop, visit:
- remote: https://kdt-gitlab.elice.io/constP/git-gitlab/-/merge_requests/new?merge_request%5Bsource_branch%5D=develop
- remote:
- To https://kdt-gitlab.elice.io/constP/git-gitlab.git
- * [new branch] develop -> develop

develop Branch를 만들었다면, develop Branch를 원격 저장소에 push 해보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

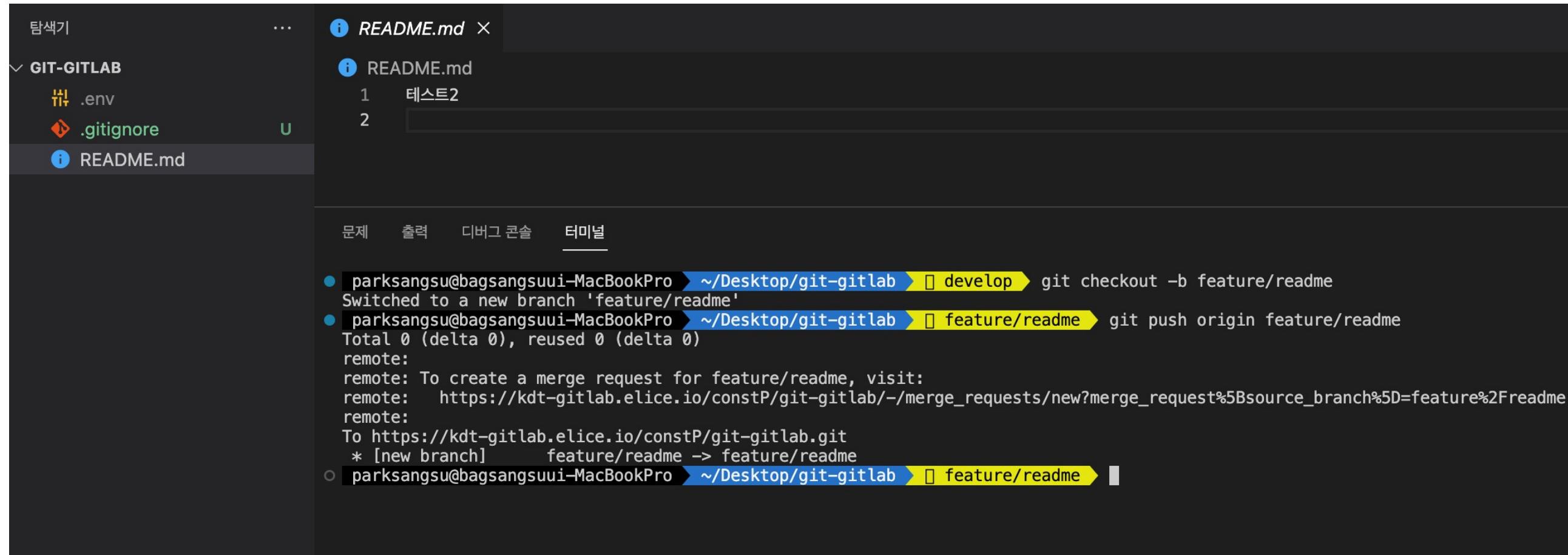
✓ Develop Branch 생성



왼쪽 화면처럼 Branch 명이 노출됐다면,
성공적으로 Branch를 생성하고, 원격 저장소에 반영했습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Feature Branch 생성

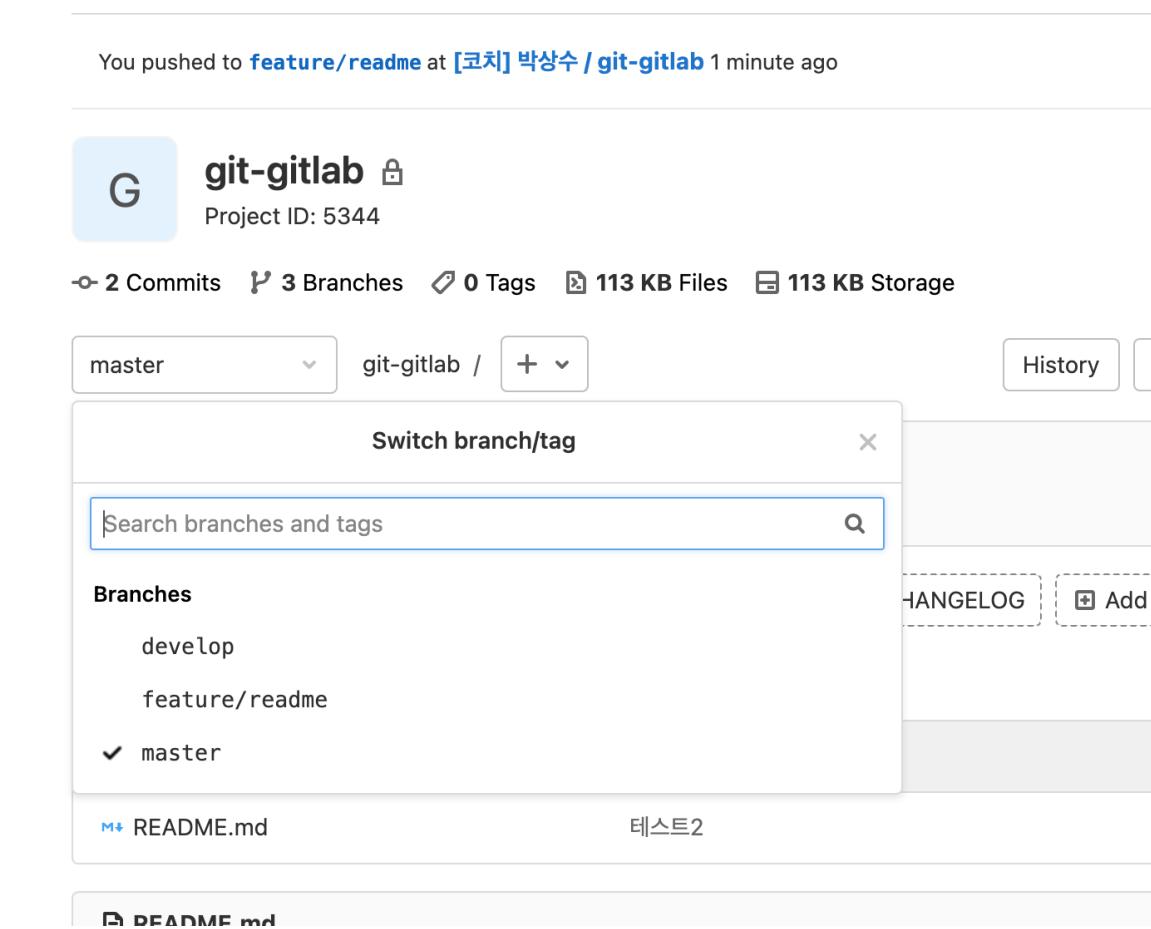


```
parksangsu@bagsangsuui-MacBookPro ~/Desktop/git-gitlab develop git checkout -b feature/readme
Switched to a new branch 'feature/readme'
parksangsu@bagsangsuui-MacBookPro ~/Desktop/git-gitlab feature/readme git push origin feature/readme
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for feature/readme, visit:
remote: https://kdt-gitlab.elice.io/constP/git-gitlab/-/merge_requests/new?merge_request%5Bsource_branch%5D=feature%2Freadme
remote:
To https://kdt-gitlab.elice.io/constP/git-gitlab.git
 * [new branch]      feature/readme -> feature/readme
parksangsu@bagsangsuui-MacBookPro ~/Desktop/git-gitlab feature/readme
```

develop Branch 생성과 동일하게
feature Branch도 생성한 후, 원격 저장소에 push 해보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Feature Branch 확인

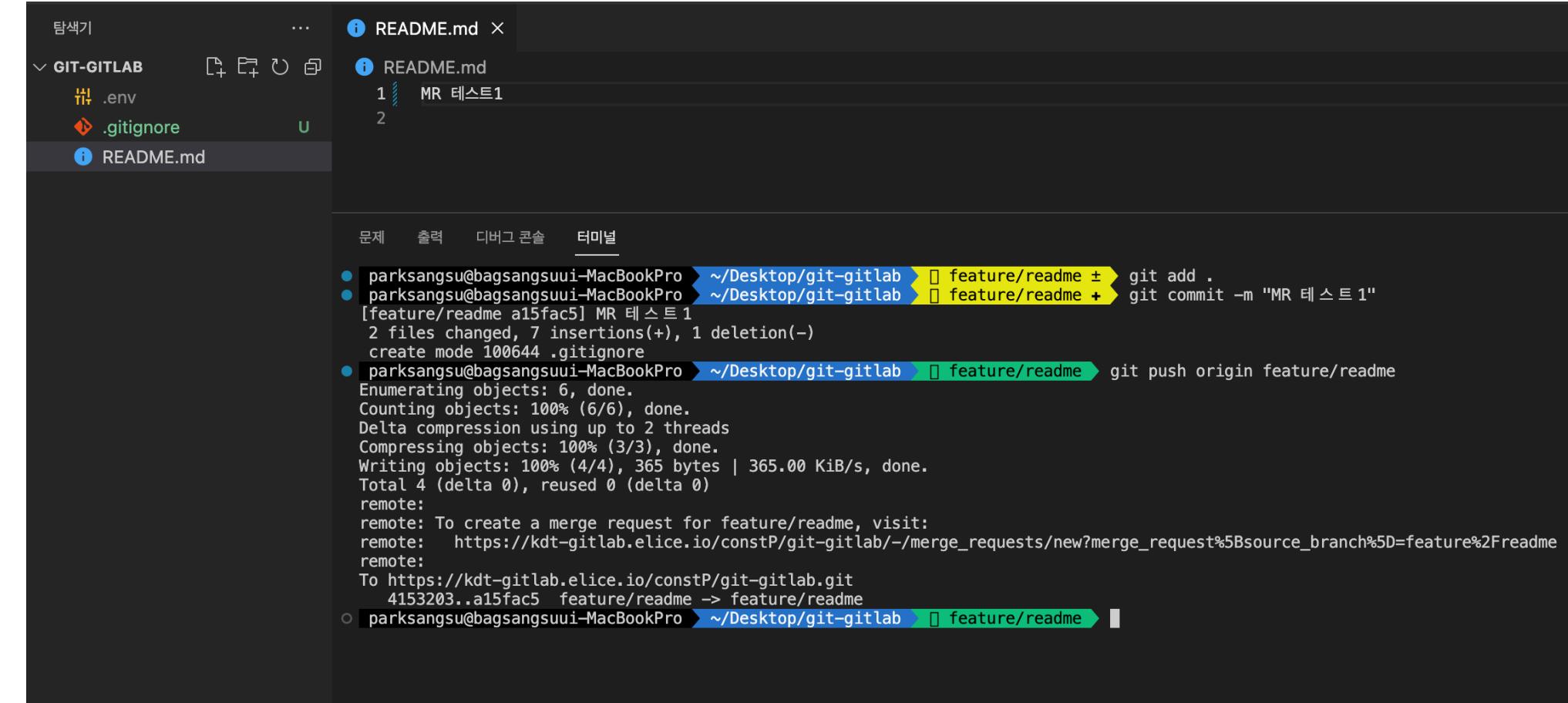


왼쪽 화면처럼 Branch 명이 노출됐다면 성공적으로 Branch를 생성하고, 원격 저장소에 반영했습니다.

그럼 이제 feature/readme Branch에서 특정 파일을 변경해보고, 변경 사항을 develop Branch에 머지해보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Feature/readme Branch 작업



The screenshot shows a terminal window with the following session:

```
parksangs@bagsangsuui-MacBookPro ~/Desktop/git-gitlab feature/readme ± git add .
[parksangs@bagsangsuui-MacBookPro ~/Desktop/git-gitlab feature/readme +] MR 테스트1
2 files changed, 7 insertions(+), 1 deletion(-)
create mode 100644 .gitignore
[parksangs@bagsangsuui-MacBookPro ~/Desktop/git-gitlab feature/readme] git push origin feature/readme
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 365 bytes | 365.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for feature/readme, visit:
remote: https://kdt-gitlab.elice.io/constP/git-gitlab/-/merge_requests/new?merge_request%5Bsource_branch%5D=feature%2Freadme
remote:
To https://kdt-gitlab.elice.io/constP/git-gitlab.git
 4153203..a15fac5  feature/readme -> feature/readme
[parksangs@bagsangsuui-MacBookPro ~/Desktop/git-gitlab feature/readme]
```

feature/readme Branch에 commit을 생성하고, 생성한 commit을 원격 저장소에 push 하겠습니다.
commit은 master, develop Branch가 아닌 feature/readme Branch에 올려줍니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ 원격 저장소 확인

The screenshot shows a GitLab repository page for a project named 'git-gitlab'. The current branch is 'feature/readme'. A recent commit was pushed just now by 'MR 테스트1'. The commit message is '[코치] 박상수 authored 45 seconds ago'. The commit hash is 'a15fac55'. Below the commit, there is a table showing file details:

Name	Last commit	Last update
.gitignore	MR 테스트1	45 seconds ago
README.md	MR 테스트1	45 seconds ago

Under the README.md file, there is a preview pane showing the content 'MR 테스트1'.

원격 저장소의 feature/readme Branch에서 README.md 파일을 확인했을 때
“MR 테스트1”이라고 작성됐다면 해당 Branch에 commit이 제대로 추가되었습니다.
그럼 이제 GitLab에서 MR을 활용하여 Branch를 병합해보겠습니다.

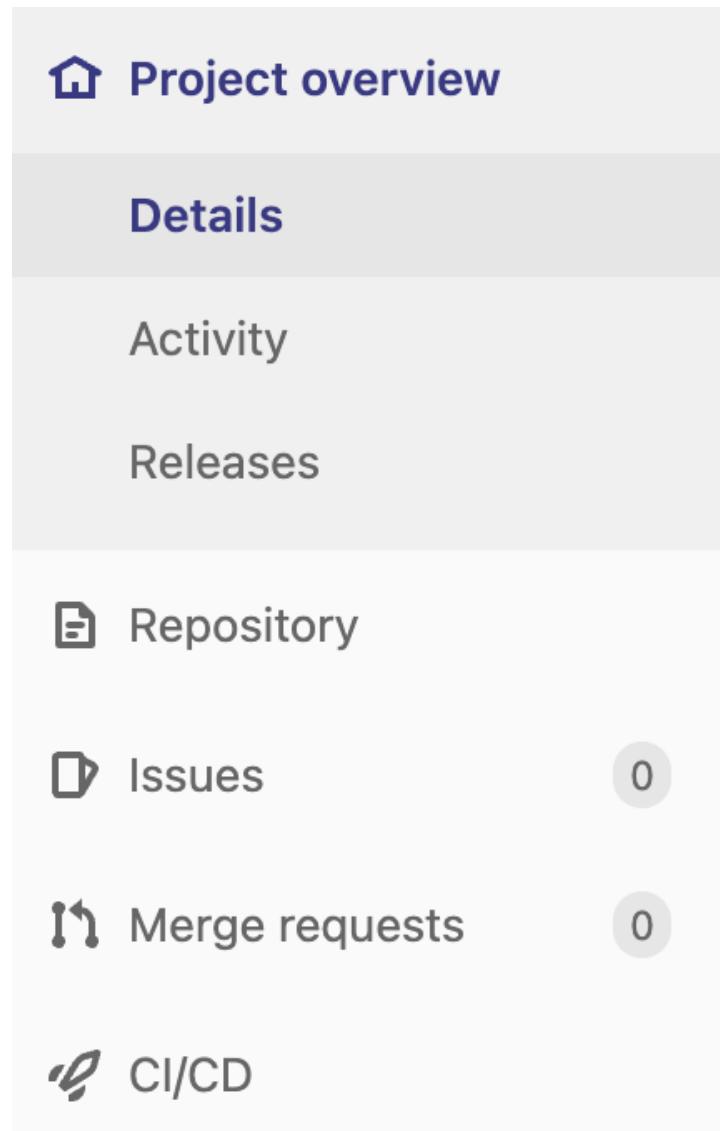
04 실무에서 필수! Git & GitLab 사용법

⑤ MR을 활용하여 다른 branch에 내 코드를 반영해보자.

- MR은 Merge Requests의 줄임말이며, 뜻 그대로 merge를 요청한다는 뜻입니다.
- 예를 들어 develop Branch에 내가 방금 작성한 코드의 변경 사항을 합쳐도 되겠니? 라고 다른 개발자의 동의를 구하기 위해 요청을 하는 것입니다.
- develop Branch를 관리하는 다른 개발자들은 요청 사항을 살펴보면서, 코드 리뷰를 통해 코드를 점검하고, 점검이 완료되면 develop Branch에 코드를 합쳐도 된다고 승인을 합니다.
- 승인이 완료됐다면, 비로소 코드를 합치게 됩니다. 이를 통해 개발자 간의 협업을 할 수 있습니다.
- 그럼 지금부터 GitLab에서 MR 하는 방법에 대해 알아보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

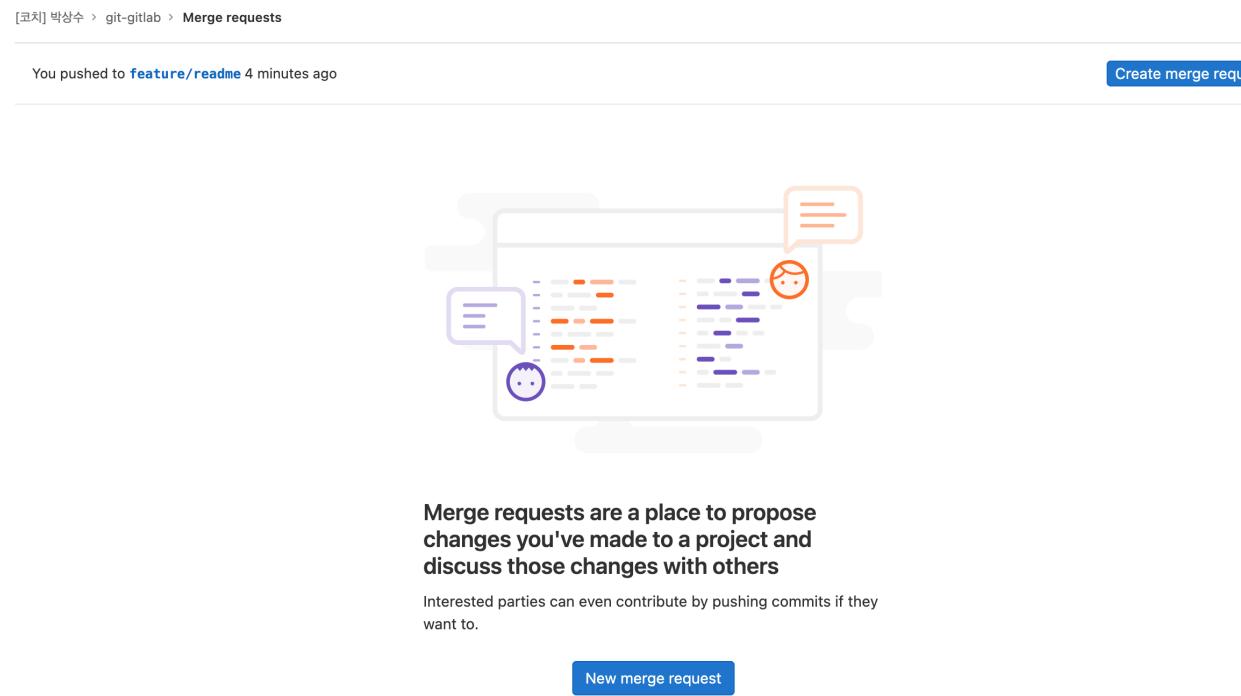
⑤ Merge Requests



- MR을 하기 위해 GitLab의 왼쪽을 보면 Merge requests를 클릭해줍니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Merge Requests



New merge request 버튼을 눌러줍니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Merge Requests

[코치] 박상수 > git-gitlab > Merge requests > New

New merge request

Source branch	Target branch
constP/git-gitlab feature/readme	constP/git-gitlab develop
 MR 테스트1 [코치] 박상수 authored 5 minutes ago a15fac55	 테스트2 [코치] 박상수 authored 1 hour ago 41532036

[Compare branches and continue](#)

Source branch는 현재 합치려고 하는 branch입니다.

Target branch는 소스 코드가 합쳐지는 branch입니다.

feature/readme branch를 develop branch로 합치기 때문에 왼쪽과 같이 설정해줍니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Merge Requests

The screenshot shows the 'New merge request' page in GitLab. At the top, there's a breadcrumb navigation: [코치] 박상수 > git-gitlab > Merge requests > New. Below it, the title 'New merge request' is displayed. A note says 'From feature/readme into develop' with a 'Change branches' link. The 'Title' field contains 'MR 테스트1'. A note below it says 'Start the title with Draft: to prevent a merge request that is a work in progress.' and 'Add description templates to help your contributors communicate effectively'. The 'Description' section has tabs 'Write' (which is selected) and 'Preview'. The 'Write' tab shows the text 'MR 테스트1'. A note at the bottom says 'Markdown and quick actions are supported'.

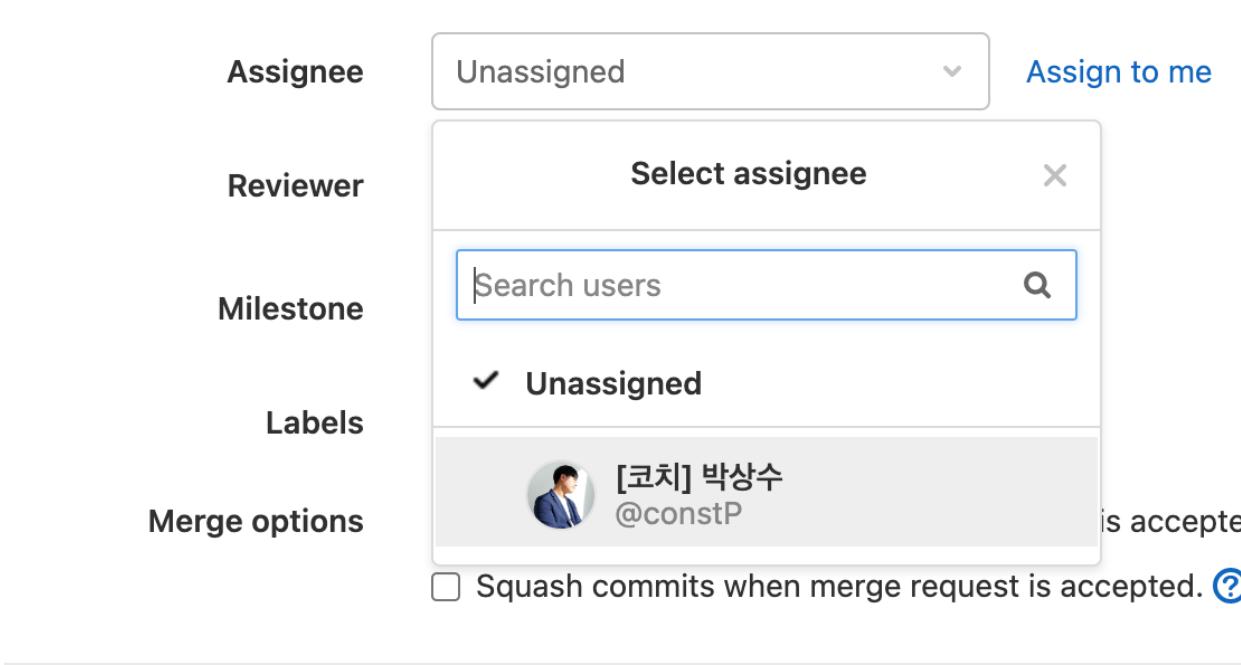
Compare branches and continue 버튼을 눌렀다면, 왼쪽과 같이 MR을 생성하는 화면이 나옵니다.

Title은 MR에 대한 제목을 작성하고, Description은 MR에 대한 내용을 작성해줍니다.

지금은 왼쪽과 같이 간단하게 작성하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Merge Requests

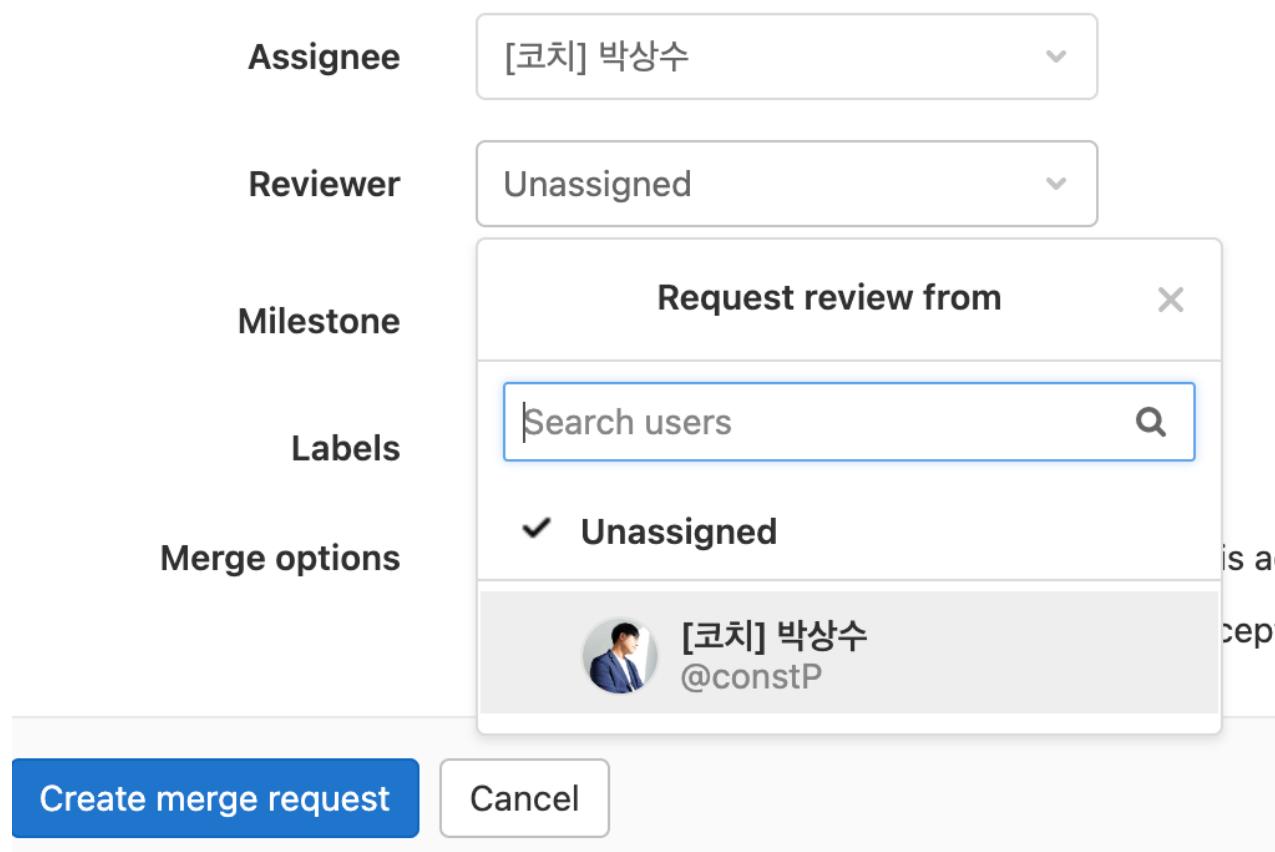


Title, description을 작성했다면, 아래 Assignee를 선택해줍니다.

MR을 올린 사람을 선택해줍니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Merge Requests



Reviewer는 내가 작성한 코드를 develop Branch에 병합할 때,
코드를 검수하는 사람을 선택해줍니다.

현재는 개인 Repository를 활용하고 있기 때문에, 자기 자신만 나올 것입니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Merge Requests

The screenshot shows the 'Create merge request' dialog box. It includes fields for Assignee (selected as [코치] 박상수), Reviewer (selected as [코치] 박상수), Milestone (selected as Milestone), Labels (selected as Labels), and Merge options (checkboxes for 'Delete source branch when merge request is accepted.' checked and 'Squash commits when merge request is accepted.' unchecked). At the bottom are 'Create merge request' and 'Cancel' buttons. Below the dialog, a commit history is visible: '11 Mar, 2023 1 commit' by 'MR 테스트1 [코치] 박상수 authored 6 minutes ago'.

Assignee와 Reviewer를 모두 선택했다면 바로 Create merge request 버튼을 클릭하는 것이 아닌, 아래 commit 내역과 어떤 코드들이 변화될 수 있는지를 살펴보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Merge Requests

The screenshot shows a 'Create merge request' dialog box with 'Create merge request' and 'Cancel' buttons. Below it, a navigation bar has 'Commits 1' and 'Changes 2' selected. A message says 'Showing 2 changed files ▾ with 7 additions and 1 deletion'. The changes are listed under two files:

- .gitignore**:
1 + # compiled output
2 + /dist
3 + /node_modules
4 +
5 + # env
6 + *.env
- README.md**:
1 - 테스트2
1 + MR 테스트1

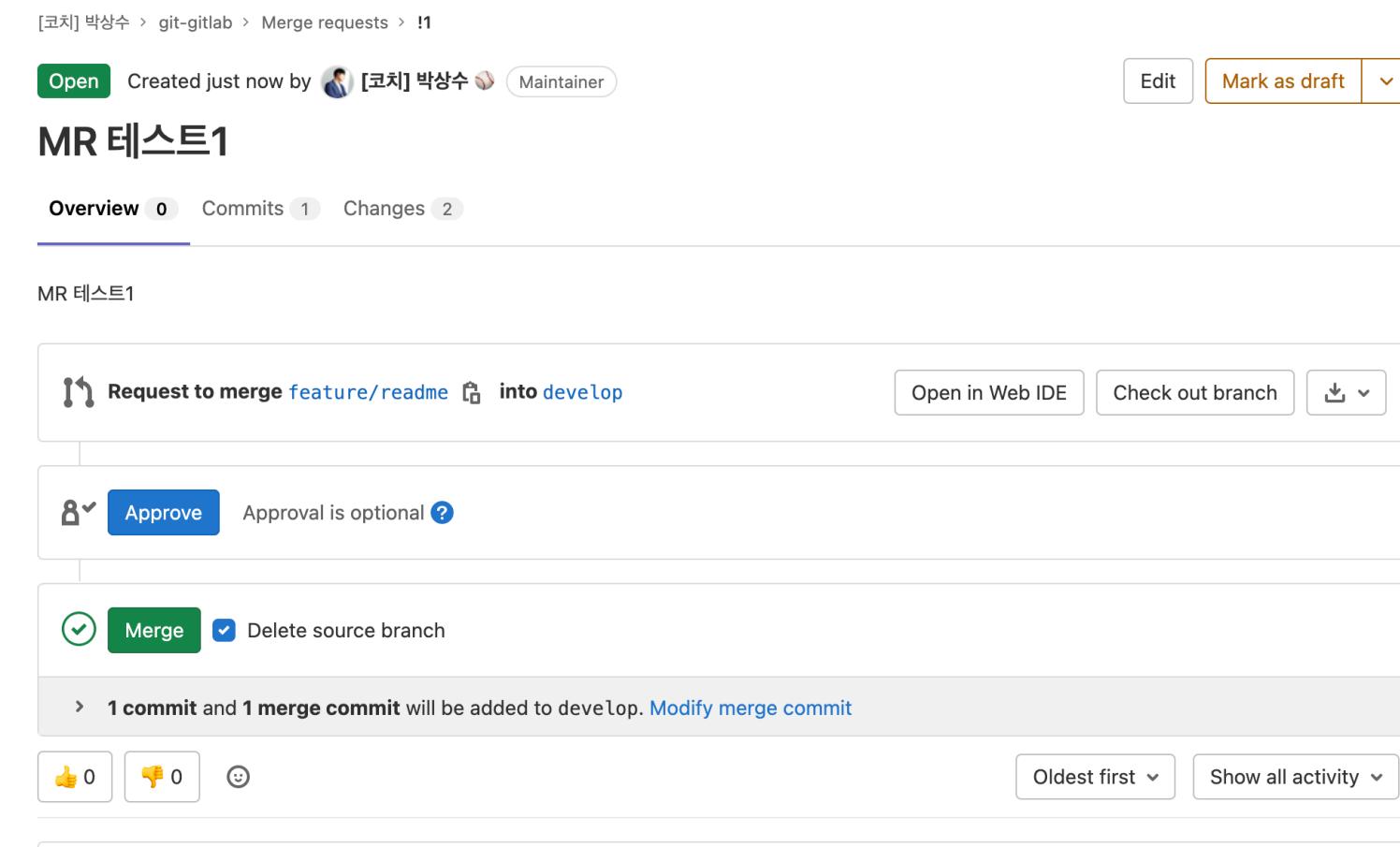
Changes를 클릭해서, 어떤 코드들이 변화되는지 살펴볼 수 있습니다.

commit 내역과 Changes를 통해 변경 사항을 모두 확인했다면

Create merge request 버튼을 클릭하여 MR을 생성합니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Merge Requests

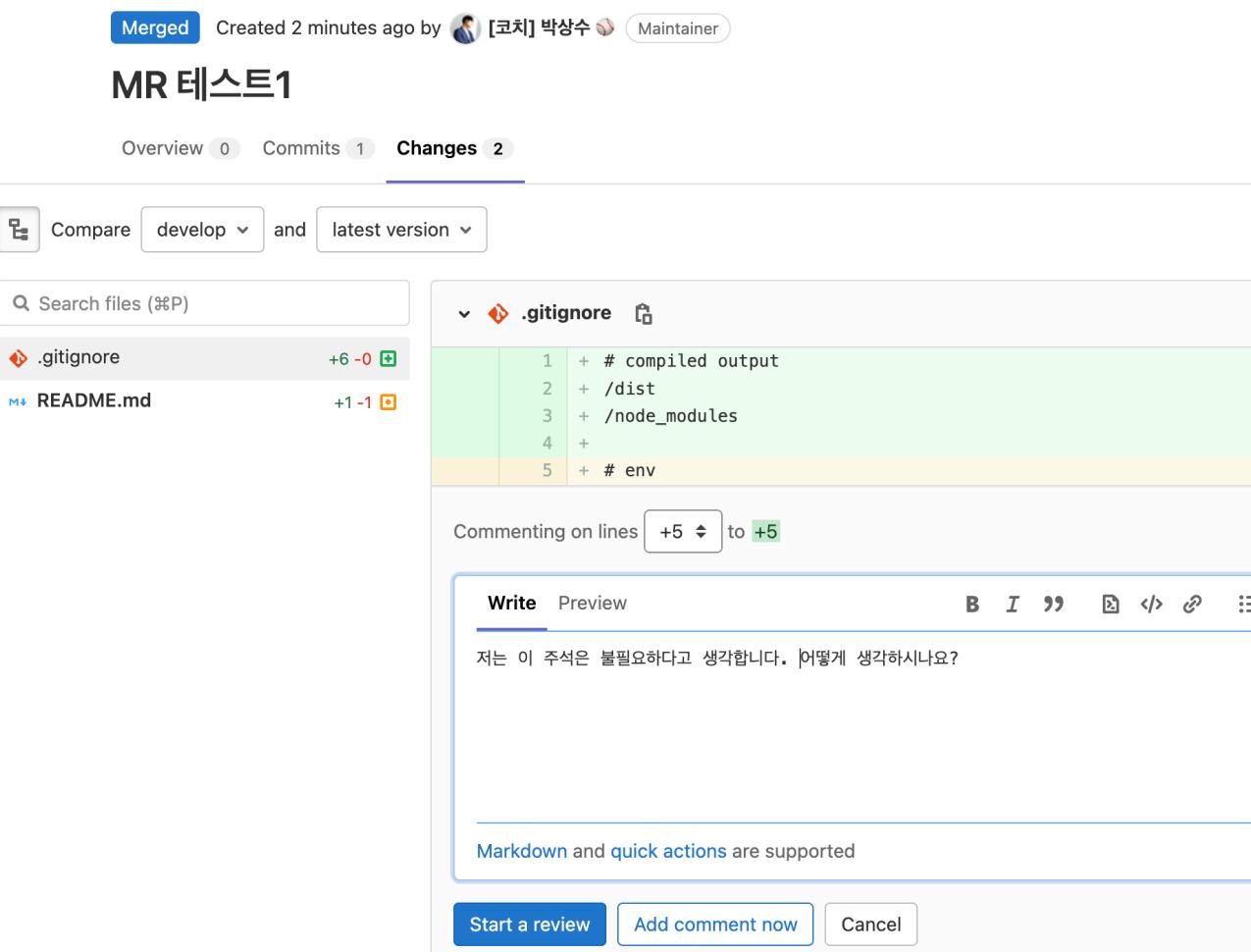


MR을 생성하면 왼쪽과 같은 화면이 나옵니다.

작성한 Title, description이 나옵니다. 만약 리뷰어가 Changes에 나온 코드를 살펴보고, 문제가 없다면 Approve 버튼을 클릭하여 develop Branch에 코드가 병합되어도 괜찮다고 표시해줍니다.

04 실무에서 필수! Git & GitLab 사용법

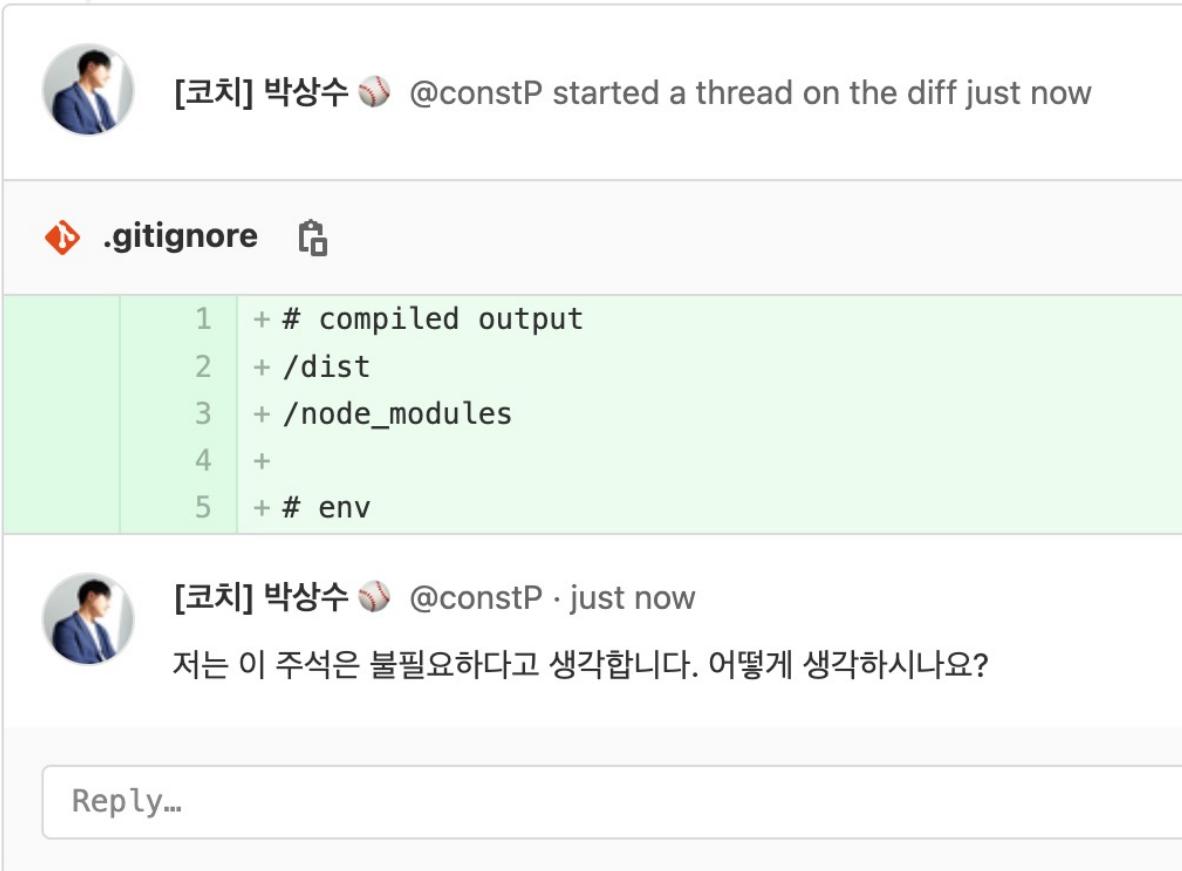
✓ Merge Requests



changes에서 코드의 변경이 필요하다면,
리뷰어는 변경이 필요한 코드를 클릭하여, 자신의 의견을 작성합니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Merge Requests



[코치] 박상수 🍃 @constP started a thread on the diff just now

🔴 .gitignore 📁

```
1 + # compiled output
2 + /dist
3 + /node_modules
4 +
5 + # env
```

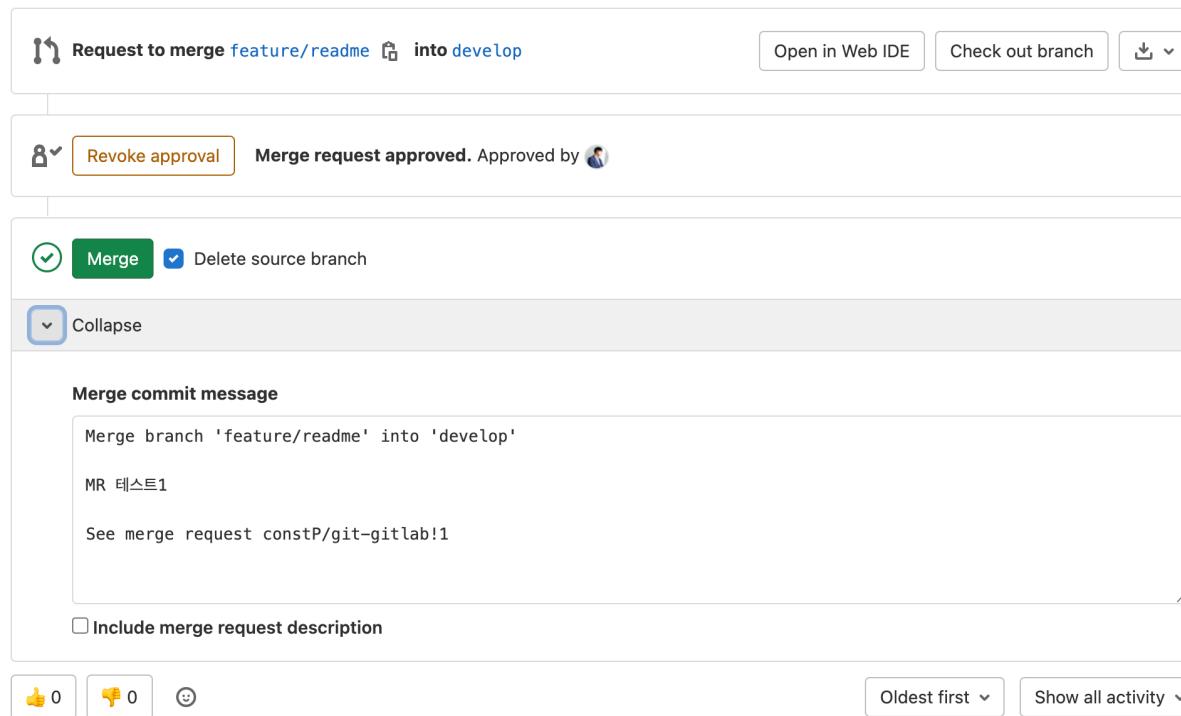
[코치] 박상수 🍃 @constP · just now
저는 이 주석은 불필요하다고 생각합니다. 어떻게 생각하시나요?

Reply...

리뷰가 모두 작성됐다면 왼쪽과 같이
코드 아래 부분에 해당 코드에 대한 리뷰가 남습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Merge Requests



코드의 문제가 없고, Approve까지 받았다면 비로소 코드를 병합해줍니다.

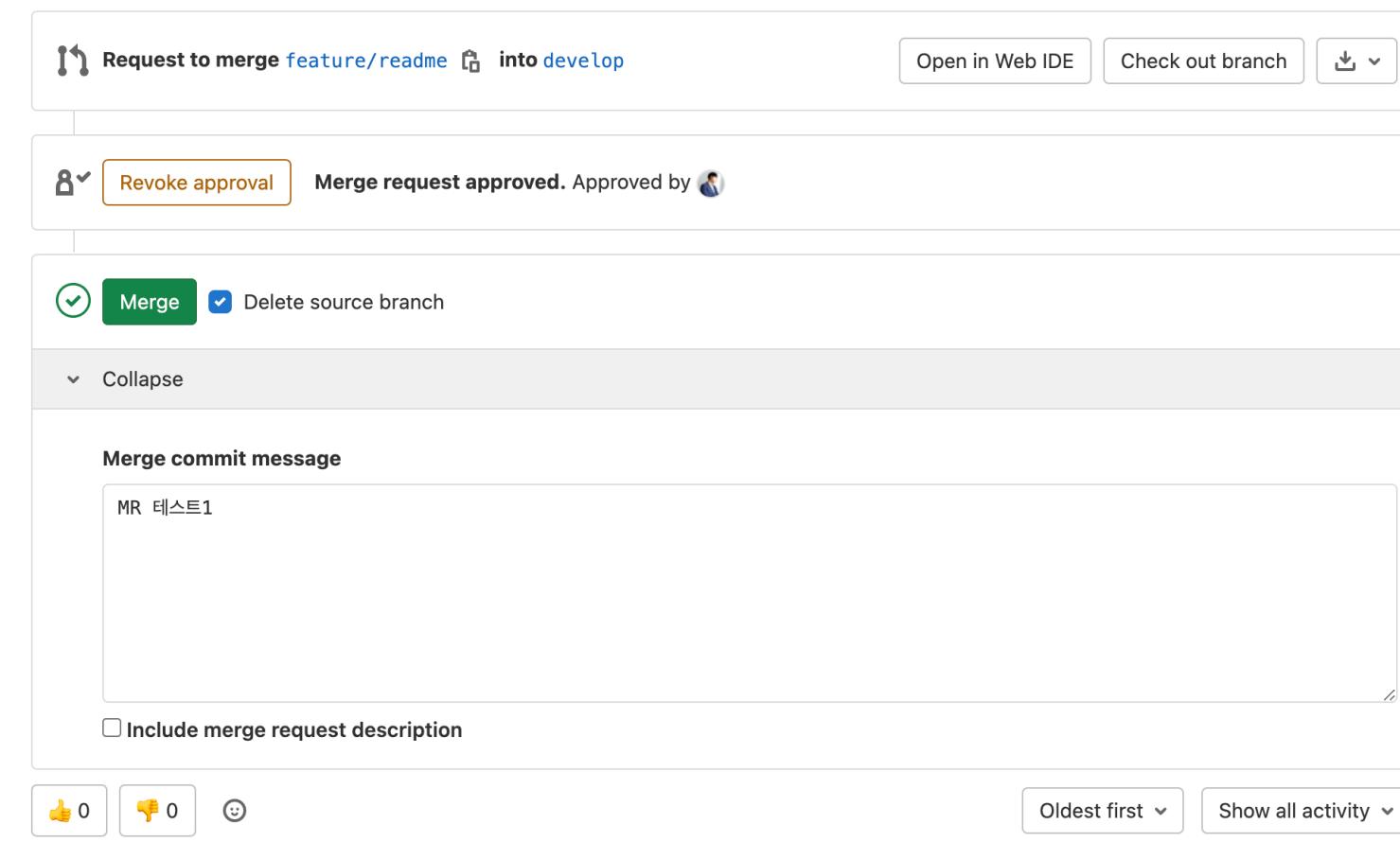
Merge를 바로 하기 전, 아래 화살표를 클릭하면,

Merge commit message를 별도로 구성할 수 있습니다.

현재는 Git이 자동으로 생성해준 commit message가 작성되어 있습니다.

04 실무에서 필수! Git & GitLab 사용법

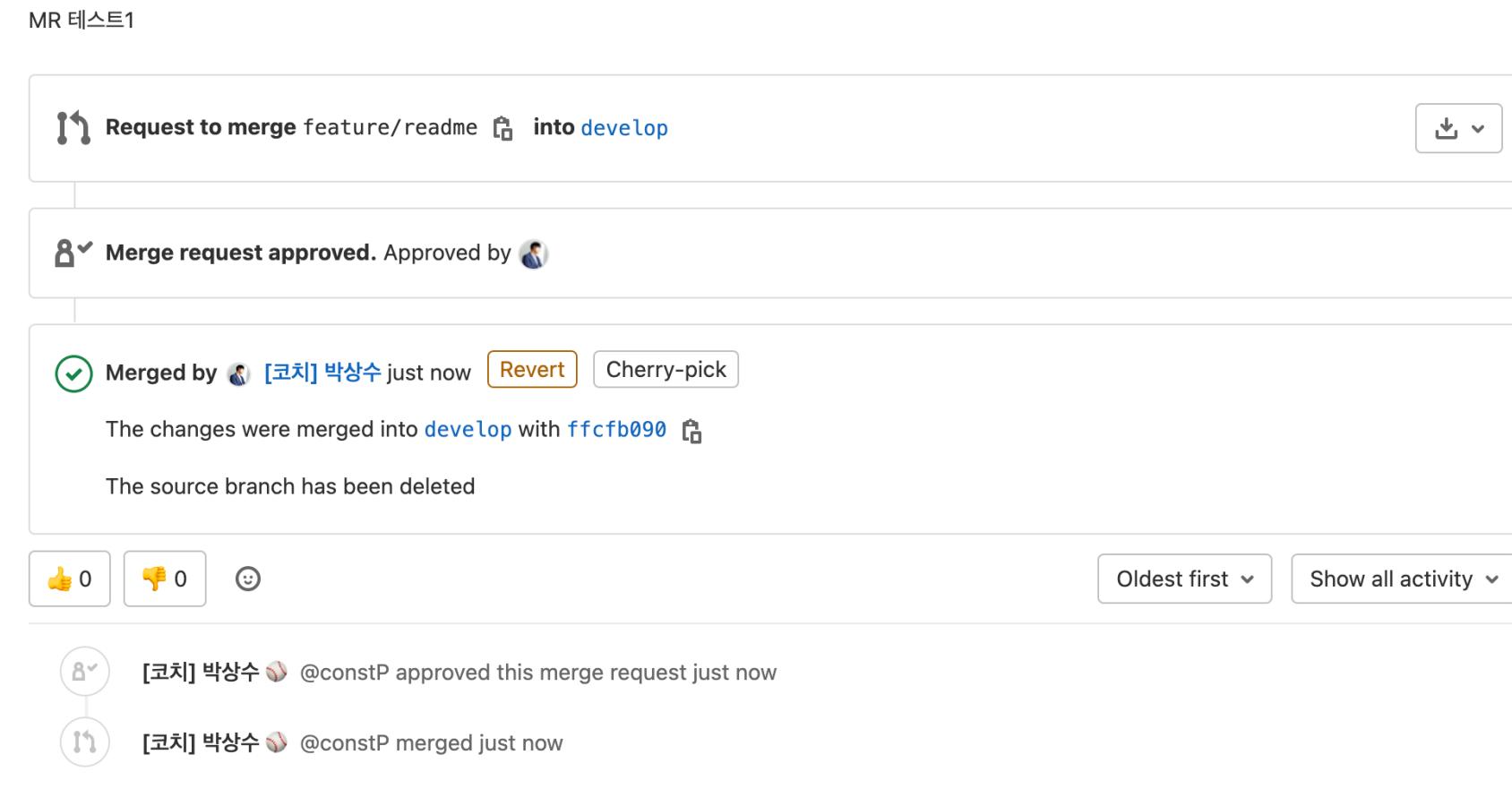
✓ Merge Requests



Merge commit message를 왼쪽과 같이 작성해보도록 하겠습니다.
작성이 완료됐다면, 비로소 Merge 버튼을 클릭하여 코드를 병합해줍니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Merge Requests



화면처럼 Merge가 완료됐다는 표시가 출력됐다면, 성공적으로 MR을 완료했습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Merge Requests

The screenshot shows a GitLab repository page for a project named 'git-gitlab'. The 'develop' branch is selected. A merge request titled 'MR 테스트1' was authored by 박상수 15 seconds ago. The commit history shows two commits: one for '.gitignore' and one for 'README.md', both authored by MR 테스트1 11 minutes ago. The 'README.md' file is expanded, showing its content: 'MR 테스트1'.

Name	Last commit	Last update
.gitignore	MR 테스트1	11 minutes ago
README.md	MR 테스트1	11 minutes ago

develop branch에 commit이 생성됐는지 확인해보고,
위와 같이 commit이 생성됐다면 MR을 완료했습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Commit 컨벤션을 지켜보자.

[코치] 박상수 > git-gitlab > Commits

develop git-gitlab

11 Mar, 2023 4 commits



MR 테스트1

[코치] 박상수 authored 7 minutes ago



MR 테스트1

[코치] 박상수 authored 18 minutes ago



테스트2

[코치] 박상수 authored 1 hour ago



테스트1

[코치] 박상수 authored 1 hour ago

- 지금까지 commit 내역을 살펴보면 어떤 코드가 어떻게 변경됐는지 message만 보면 전혀 알 수 없습니다.
- 처음 문제로 봤던 최종1, 최최종, 리얼 최종과 크게 다를 바 없는 버전 관리 방식입니다.
- 협업에서는 코드의 변경 사항을 다른 개발자가 더 잘 이해할 수 있도록 commit 컨벤션이라는 것을 지켜가면서 개발합니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Commit 컨벤션을 지켜보자.

```
type(옵션): Subject // -> 제목  
(한 줄을 띄워 분리합니다.)  
body(옵션) // -> 본문  
(한 줄을 띄워 분리합니다.)  
footer(옵션) // -> 꼬리말
```

Commit 컨벤션을 지키기 위해 크게 제목, 본문, 꼬리말 세 가지 파트로 나눠서 message를 작성합니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Commit 컨벤션을 지켜보자.

태그 이름	설명
Feat	새로운 기능을 추가할 경우
Fix	버그를 고친 경우
Design	CSS 등 사용자 UI 디자인 변경
!BREAKING CHANGE	커다란 API 변경의 경우
!HOTFIX	급하게 치명적인 버그를 고쳐야하는 경우
Style	코드 포맷 변경, 세미 콜론 누락, 코드 수정이 없는 경우
Refactor	프로덕션 코드 리팩토링
Comment	필요한 주석 추가 및 변경
Docs	문서를 수정한 경우
Test	테스트 추가, 테스트 리팩토링(프로덕션 코드 변경 X)
Chore	빌드 태스크 업데이트, 패키지 매니저를 설정하는 경우(프로덕션 코드 변경 X)
Rename	파일 혹은 폴더명을 수정하거나 옮기는 작업만인 경우
Remove	파일을 삭제하는 작업만 수행한 경우

- 제목 부분에는 type과 Subject를 작성해줘야 합니다.
- '태그:제목'의 형태로 작성합니다.
- 예를 들어 회원 가입 API를 개발했다면,
“Feat: 회원가입 API 작성”처럼
commit message를 작성할 수 있습니다.



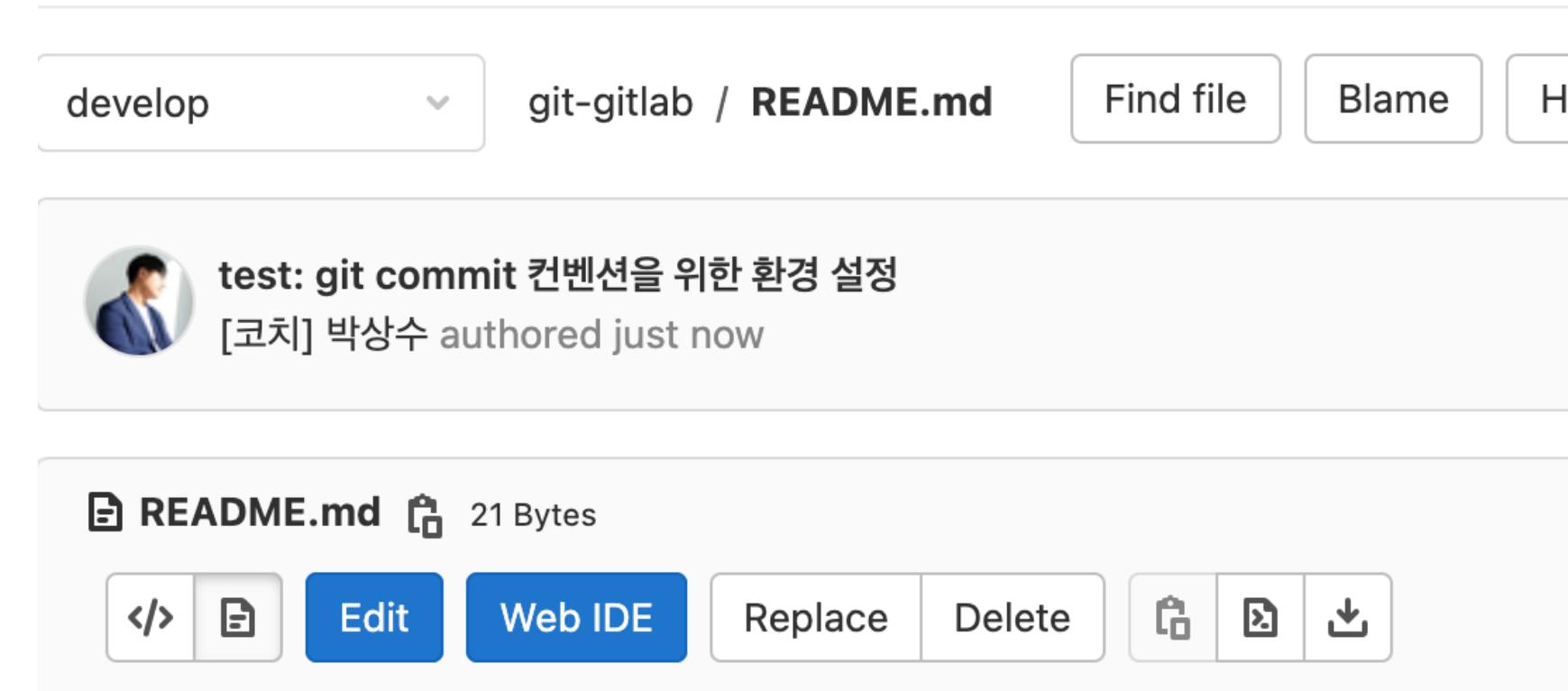
Commit 컨벤션

Commit 컨벤션에 대해 더 자세히 알고 싶으신 분들은 아래 링크를 참고해주세요.

<https://overcome-the-limits.tistory.com/6>

04 실무에서 필수! Git & GitLab 사용법

⑤ Commit 컨벤션을 지켜보자.



Commit 컨벤션을 잘 지킨 commit message를 작성해보도록 하겠습니다.

GitLab에 들어가서, README.md 파일을 git commit 컨벤션 이라는 내용으로 수정한 후 commit message를 'test: git commit 컨벤션을 위한 환경 설정'이라는 이름으로 지정해보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Commit 컨벤션을 지켜보자.

The screenshot shows a list of five commits in a Git repository. The repository path is 'develop' → 'git-gitlab'. The date is '11 Mar, 2023' and there are '5 commits'. Each commit is authored by '박상수' (Park Sang-soo) with the label '[코치]' (coach). The commit messages are:

- test: git commit 컨벤션을 위한 환경 설정
- MR 테스트1
- MR 테스트1
- 테스트2
- 테스트1

commit 컨벤션을 지켜 가며 commit message를 작성한 결과 전에 작성한 commit message 보다 보다 깔끔하고, 더 명확한 message를 전달할 수 있습니다

04 실무에서 필수! Git & GitLab 사용법

⑤ Git 충돌 상황이 발생한다면?

- 지금까지 MR을 활용해서 원격 저장소에 다른 개발자들이 함께 버전 관리하는 Branch에 내 코드를 반영하는 방법에 대해 배웠습니다.
- 협업하다보면, 만약 원격 저장소에 변경이 일어났는데, 내 로컬 환경에서의 코드에도 변경이 일어난다면 어떻게 될까요?
- 즉 로컬에서 새로운 commit을 만드는 사이에 원격 저장소에 다른 버전이 업로드 됐다면, 원격 환경과 로컬 환경 간의 '충돌'이 일어나게 됩니다.
- 충돌 상황을 만들어보고, 충돌 상황을 해결해보도록 하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Git 충돌 상황이 발생한다면?

You pushed to **develop** just now

develop git-gitlab / README.md Find file Blame Hi

test: git commit 커밋을 위한 환경 설정
[코치] 박상수 authored just now

README.md 21 Bytes

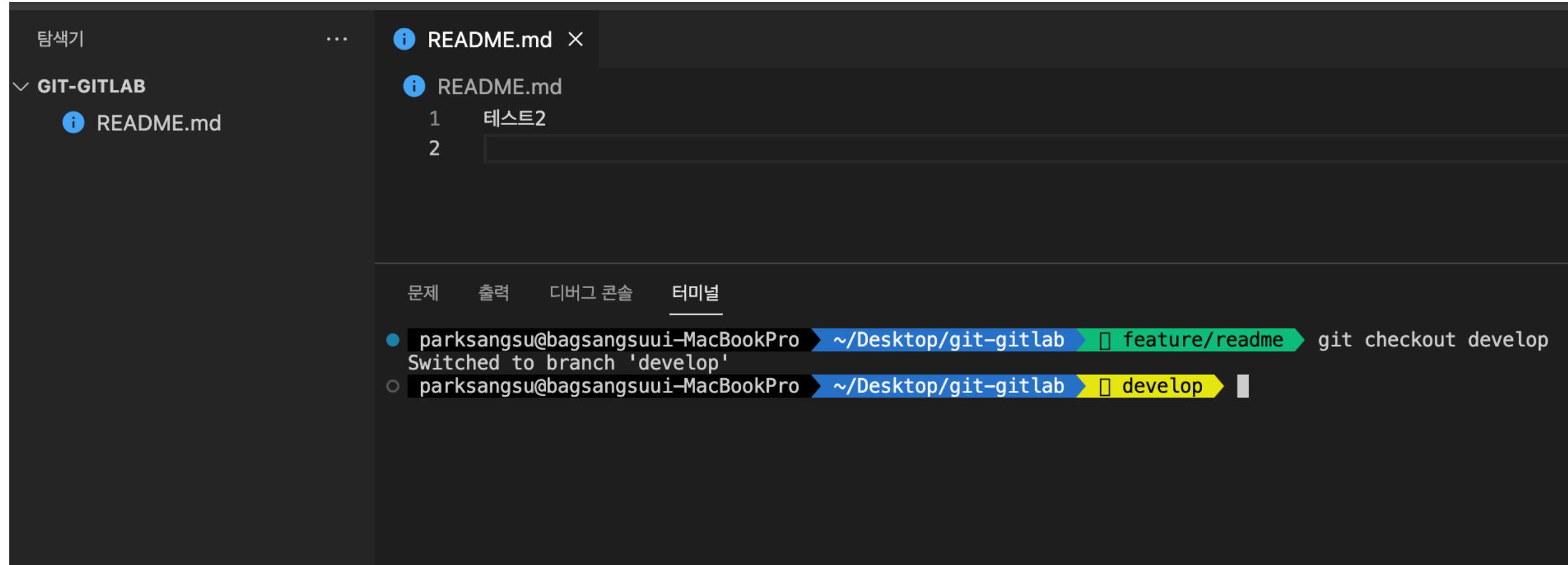
</> Edit Web IDE Replace Delete ⌂ ⌂ ⌂

git commit 커밋

방금 전 Git Commit 컨벤션을 알아보기 위해
commit 하나를 원격 저장소에 생성했습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Git 충돌 상황이 발생한다면?



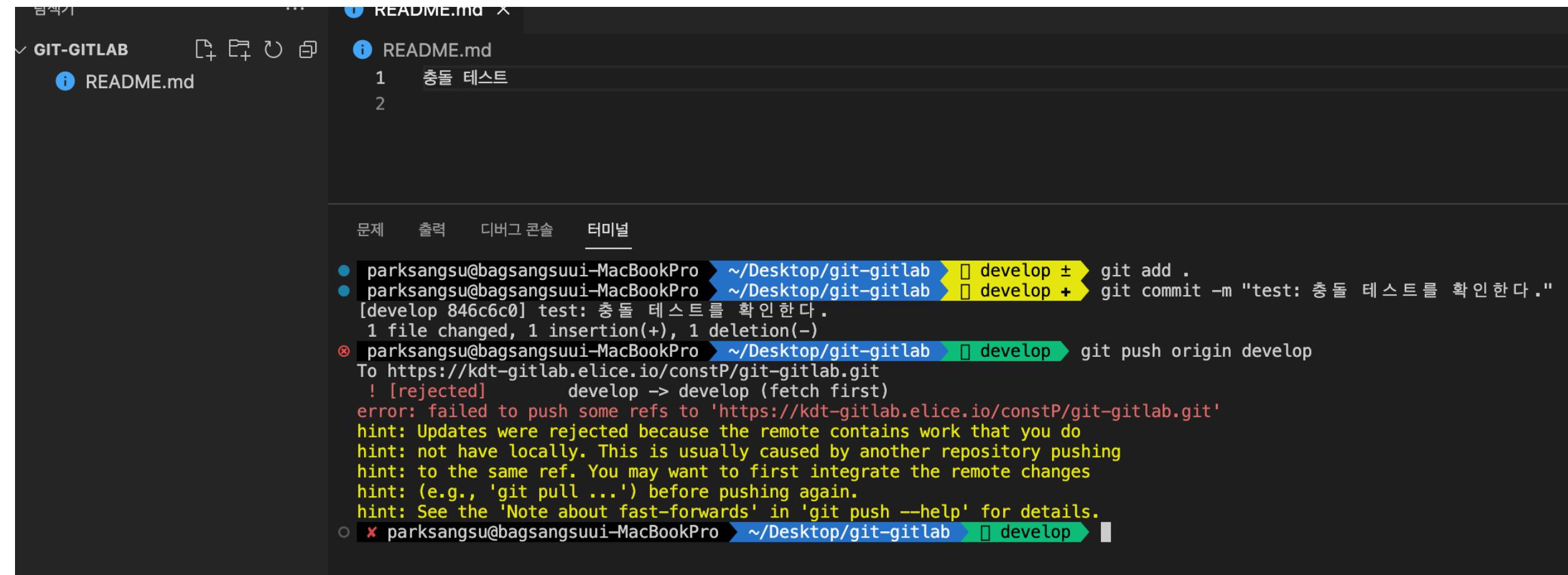
A screenshot of a terminal window within a dark-themed code editor interface. The terminal shows the following command and its execution:

```
parksangsu@bagsangsuui-MacBookPro ~/Desktop/git-gitlab feature/readme git checkout develop
Switched to branch 'develop'
parksangsu@bagsangsuui-MacBookPro ~/Desktop/git-gitlab develop
```

로컬에서는 현재 feature/readme Branch 환경에 있을텐데,
develop Branch로 이동하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Git 충돌 상황이 발생한다면?



The screenshot shows a terminal window with a dark theme. At the top, there's a file browser interface showing a folder named 'GIT-GITLAB' containing a file 'README.md'. The content of 'README.md' is shown as:

```
1 충돌 테스트  
2
```

Below the file browser is a terminal window with tabs for '문제', '출력', '디버그 콘솔', and '터미널'. The '터미널' tab is active, displaying the following command history and error message:

```
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab develop ± git add .  
[develop 846c6c0] test: 충돌 테스트를 확인 한다.  
 1 file changed, 1 insertion(+), 1 deletion(-)  
④ parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab develop ➜ git push origin develop  
To https://kdt-gitlab.elice.io/constP/git-gitlab.git  
 ! [rejected]      develop -> develop (fetch first)  
error: failed to push some refs to 'https://kdt-gitlab.elice.io/constP/git-gitlab.git'  
hint: Updates were rejected because the remote contains work that you do  
hint: not have locally. This is usually caused by another repository pushing  
hint: to the same ref. You may want to first integrate the remote changes  
hint: (e.g., 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.  
○ ✘ parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab develop ➜
```

그 후 README.md 파일에 '충돌 테스트'라는 이름으로 파일을 변경하겠습니다.

이렇게 로컬에서 버전을 만들어준 후, 원격 저장소의 develop Branch에 로컬 버전을 push 하려고 하니,
rejected라는 문구가 노출되며 경고 문구가 출력되고 있습니다.

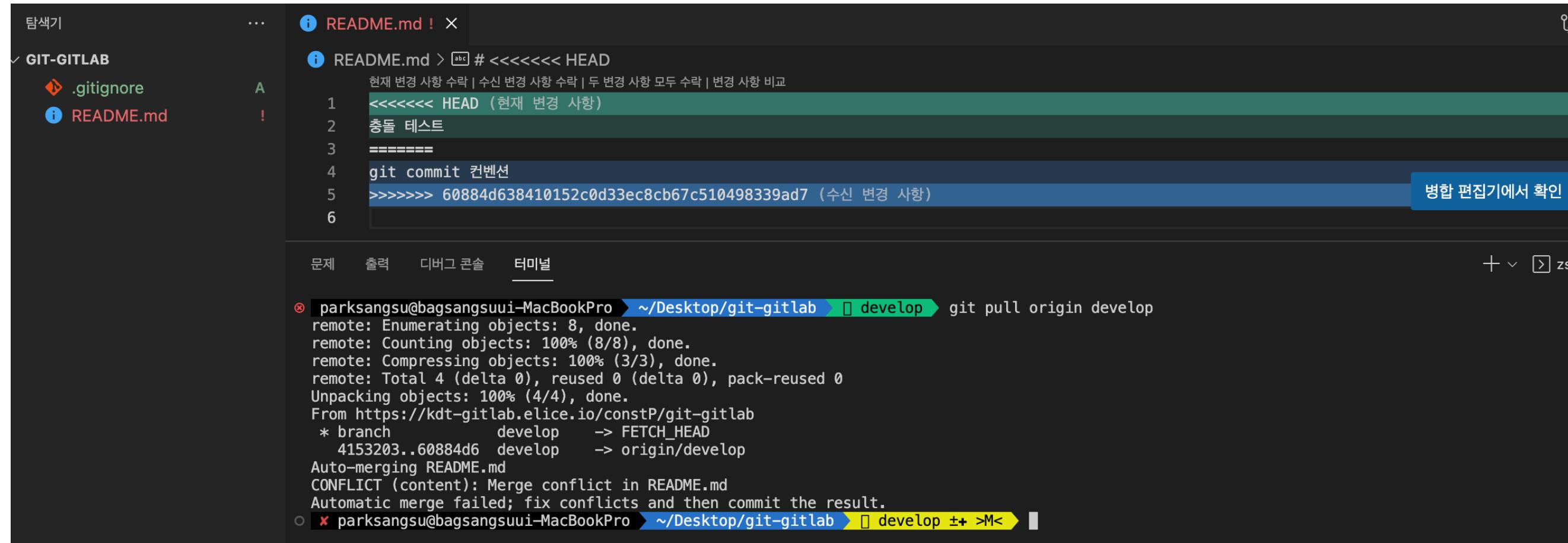
04 실무에서 필수! Git & GitLab 사용법

⑤ 왜 충돌이 발생했을까?

- 원격 저장소에서도 README.md 파일이 변경됐고, 로컬 저장소에서도 README.md 파일이 변경됐습니다.
- Git 입장에서는 원격 저장소의 변경 사항과 로컬 저장소의 변경 사항 중 어떤 변경 사항을 반영해야 하는지 혼란스러운 상태입니다.
- Git 입장에서는 어떤 변경 사항을 반영해줘야 하는지 버전을 관리하는 사람에게 “직접 문제를 해결하라고 경고를 출력시켜주는 것입니다.
- 충돌 문제를 해결하려면 어떻게 해야 할까요? 이에 대해 알아보겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Git 충돌 상황을 해결해보자



The screenshot shows a terminal window with a dark theme. On the left, there's a file tree for a repository named 'GIT-GITLAB' containing '.gitignore' and 'README.md'. The 'README.md' file is open in the main pane, showing its content. A blue bar at the bottom right of the file pane says '병합 편집기에서 확인' (Check in merge editor). The bottom half of the window is a terminal session:

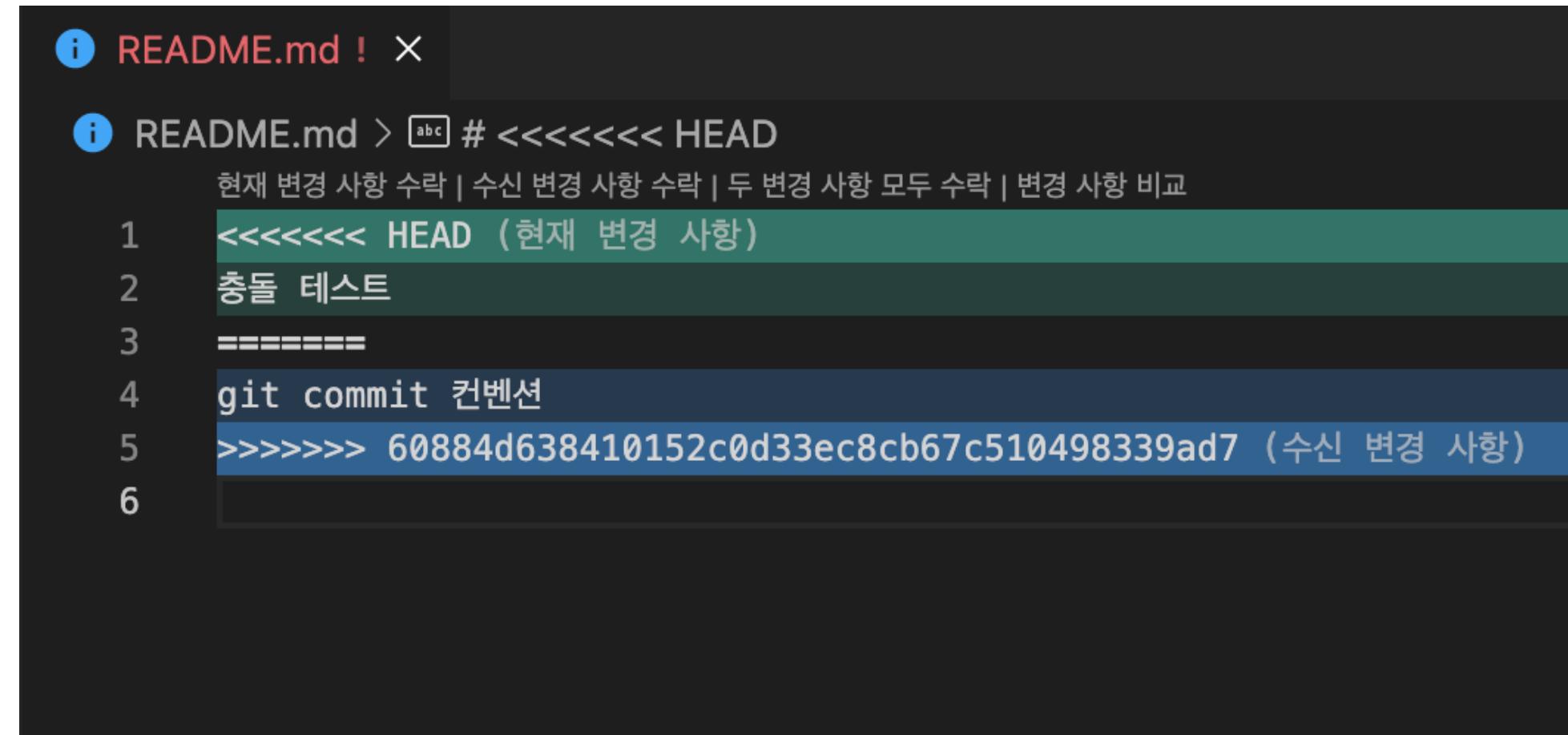
```
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab develop git pull origin develop
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://kdt-gitlab.elice.io/constP/git-gitlab
 * branch            develop    -> FETCH_HEAD
   4153203..60884d6  develop    -> origin/develop
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab develop ±+ >M<
```

Git 충돌 상황을 해결하기 위해 먼저 `git pull origin develop`이라는 명령어를 입력하겠습니다.

즉 원격 저장소의 코드 변경 사항을 로컬 저장소로 가져옴으로써, 어떤 충돌 상황이 발생했는지 살펴봐야 합니다.

04 실무에서 필수! Git & GitLab 사용법

✓ Git 충돌 상황을 해결해보자



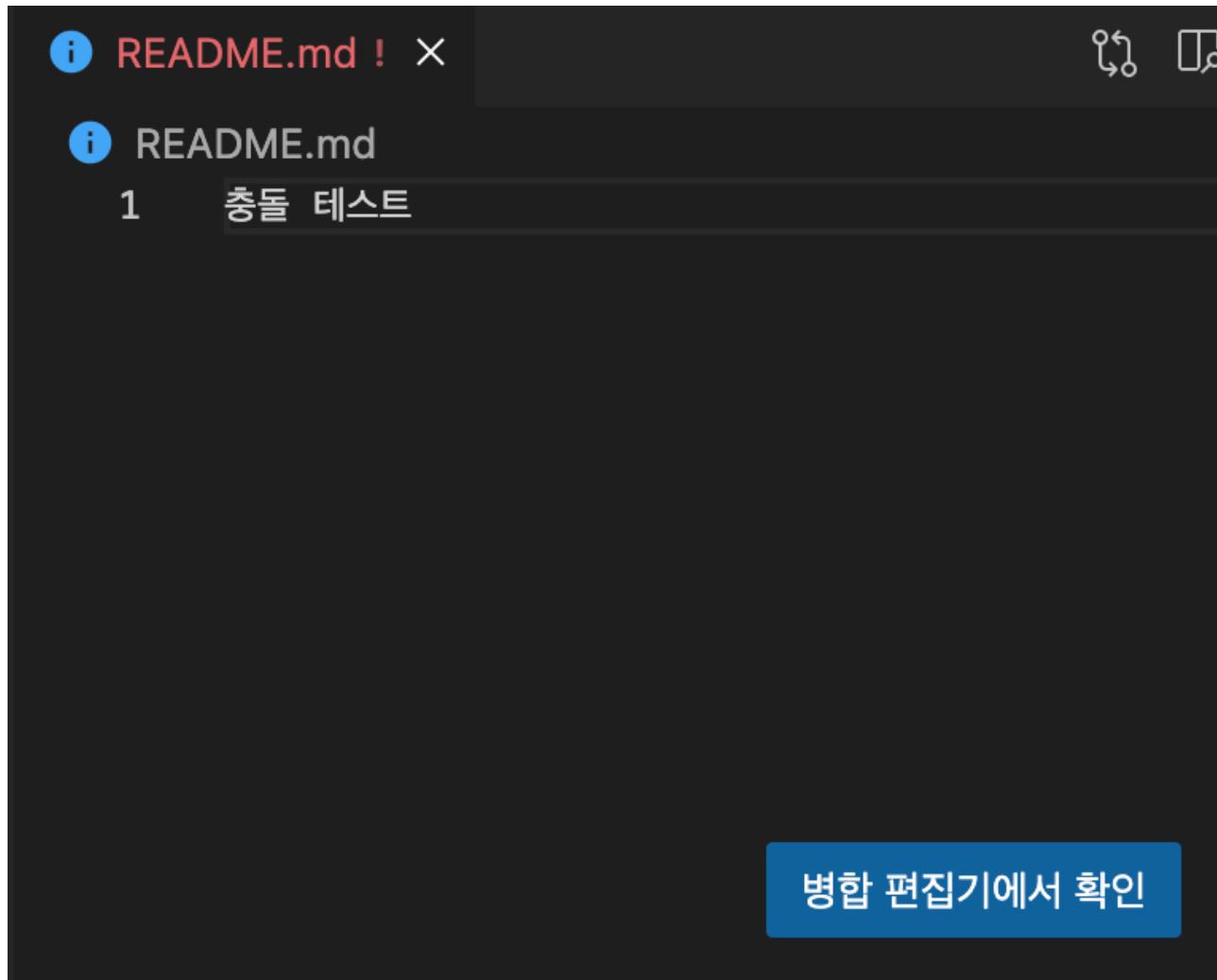
```
i README.md ! ×  
i README.md > abc # <<<<< HEAD  
  현재 변경 사항 수락 | 수신 변경 사항 수락 | 두 변경 사항 모두 수락 | 변경 사항 비교  
1 <<<<< HEAD (현재 변경 사항)  
2 충돌 테스트  
3 =====  
4 git commit 컨벤션  
5 >>>>> 60884d638410152c0d33ec8cb67c510498339ad7 (수신 변경 사항)  
6
```

충돌 상황을 보면, 로컬에서 수정한 내역이 HEAD라는 부분으로 표시되어 있고,
원격에서 수정한 내역이 수신 변경 사항이라는 부분으로 표시되어 있습니다.

로컬에서 반영한 코드를 버전으로 생성하고 싶기 때문에, “충돌 테스트”라는 문구를 제외한 다른 문구들은 모두 제거하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

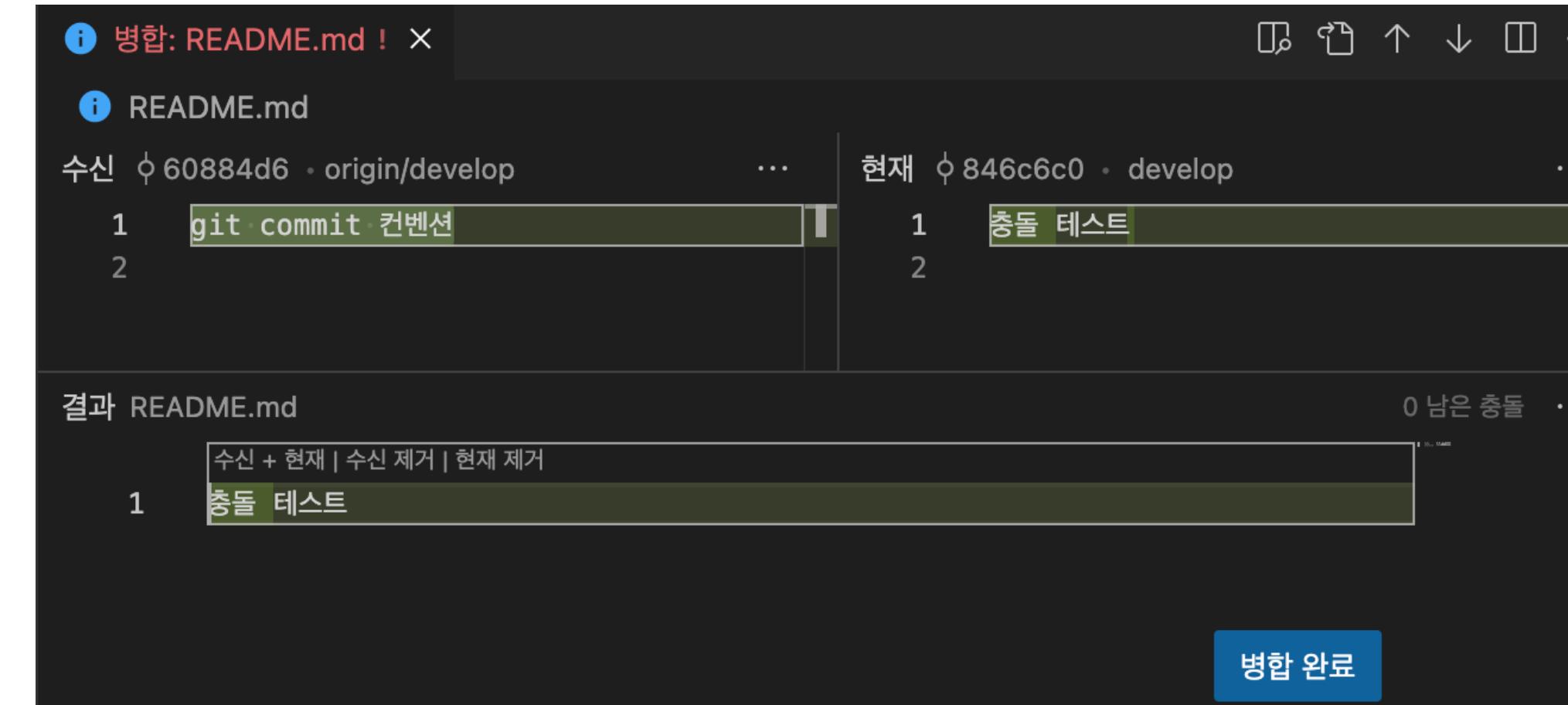
⑤ Git 충돌 상황을 해결해보자



“충돌 테스트” 문구만 남기고, 모두 지운 후,
‘병합 편집기에서 확인’이라는 버튼을 클릭해줍니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Git 충돌 상황을 해결해보자



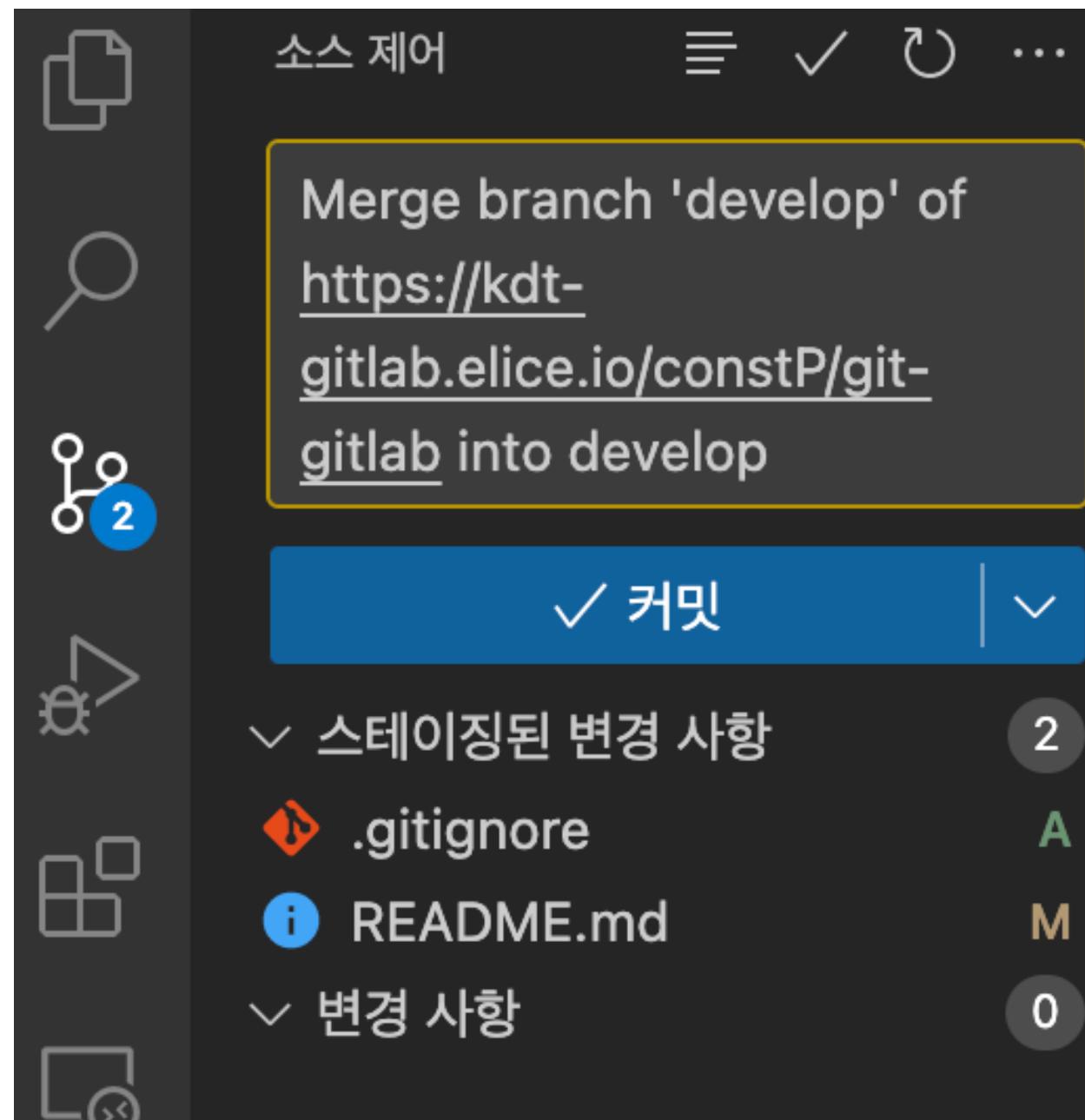
최종적으로 어떤 내역을 병합할 것인지 마지막으로 보여주는 화면입니다.

충돌 테스트라는 문구로 병합할 것이기 때문에 결과 창에는 “충돌 테스트” 문구만 남아있는 것을 볼 수 있습니다.

코드를 모두 반영했다면 “병합 완료” 버튼을 눌러줍니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Git 충돌 상황을 해결해보자



충돌 상황을 모두 해결했다면, 원격 저장소에 저장된 코드를 로컬 환경에 합치는 작업이 필요합니다.

이를 위해 원격 저장소에서 pull 받아온 코드를 로컬 환경에 commit하겠습니다.

Commit message를 보면, commit 컨벤션에 맞지 않기 때문에, commit 컨벤션에 맞게 새롭게 commit message를 작성하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Git 충돌 상황을 해결해보자

```
Automatic merge failed; fix conflicts and then commit the result.
● ✘ parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab ⏎ develop + >M< git commit -m "fix: 코드 병합 확인"
[develop 6fb9a80] fix: 코드 병합 확인
● parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab ⏎ develop ⏎ git push origin develop
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 630 bytes | 630.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for develop, visit:
remote: https://kdt-gitlab.elice.io/constP/git-gitlab/-/merge_requests/new?merge_request%5Bsource_branch%5D=develop
remote:
To https://kdt-gitlab.elice.io/constP/git-gitlab.git
  60884d6..6fb9a80  develop -> develop
○ parksangsu@bagsangsuum-MacBookPro ~/Desktop/git-gitlab ⏎ develop ⏎
```

fix: 코드 병합 확인이라는 commit message를 예시로 작성하여 버전을 생성하고,
로컬에서 생성한 버전을 원격 저장소로 push 하겠습니다.

04 실무에서 필수! Git & GitLab 사용법

⑤ Git 충돌 상황을 해결해보자

The screenshot shows a GitLab repository interface for a project named 'git-gitlab'. The top navigation bar includes 'develop' (branch dropdown), 'git-gitlab /' (repository dropdown), and a '+' button. On the right, there are links for 'History', 'Find file', 'Web IDE', and a dropdown menu with 'Copy SSH clone URL'. Below the navigation is a commit list:

- A recent commit by 'fix: 코드 병합 확인' (authored 21 seconds ago) is shown with a blue 'Copy' icon.

Below the commit list is a table showing the status of files:

Name	Last commit	Last update
.gitignore	MR 테스트1	33 minutes ago
README.md	fix: 코드 병합 확인	21 seconds ago

Under the table, there's a preview of the 'README.md' file content:

```
GitLab 테스트
```

원격 저장소를 확인하면 로컬에서 생성한 commit이 원격 저장소에 잘 반영된 것을 살펴볼 수 있습니다.
실습을 통해 충돌이 일어나는 경우엔 개발자가 직접 충돌을 해결해줘야 한다는 것을 배웠습니다.

05

GitLab을 더 잘 활용해보자.

05 GitLab을 더 잘 활용해보자

⑤ 다른 개발자들과 똑똑하게 협업해보자

- 지금까지 GitLab에 코드를 업로드하고, MR을 하는 방법까지 살펴봤습니다.
- 지금부터는 다른 개발자들과 GitLab을 활용하여 똑똑하게 협업하는 방법에 대해 배워보겠습니다.
- 라벨, 마일스톤, 이슈, Wiki를 생성하여 다른 개발자와 협업하는 방법을 익혀보겠습니다.

05 GitLab을 더 잘 활용해보자

⑤ 라벨 만들기

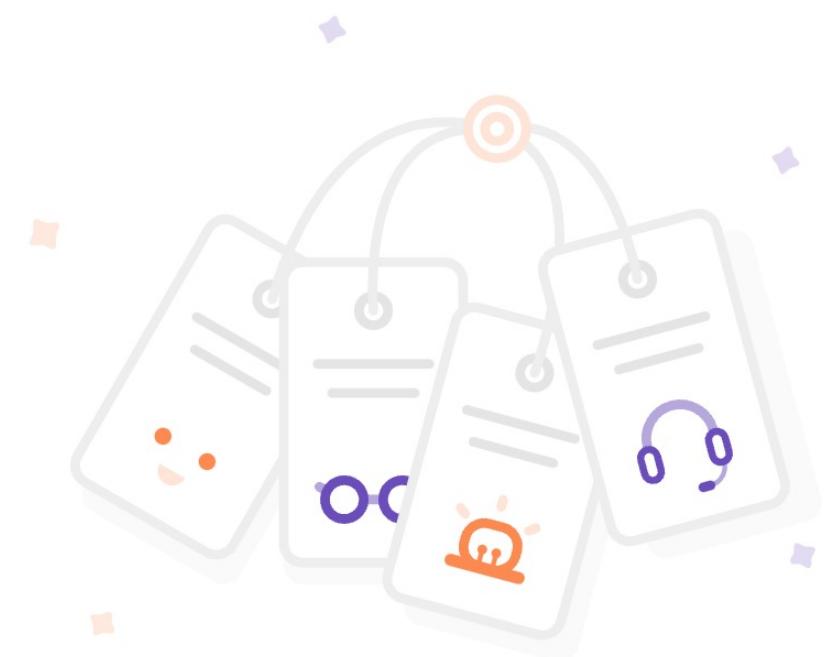
The screenshot shows the GitLab interface for the project 'git-gitlab'. The top navigation bar includes 'Projects', 'Groups', and 'More'. The sidebar on the left has options: 'Project overview', 'Issues' (0), 'List', 'Boards', 'Labels' (which is highlighted with a purple box), 'Service Desk', and 'Milestones'. The main content area shows the path '[코치] 박상수 > git-gitlab > Issues'.

라벨은 이슈를 꾸며주는 역할을 하는 도구입니다.

라벨은 메뉴바에서 Issues를 클릭하면,
Issues 목록에 Labels를 클릭할 수 있습니다.

05 GitLab을 더 잘 활용해보자

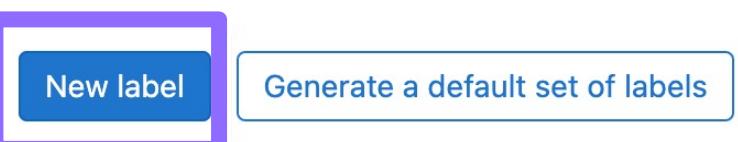
⑤ 라벨 만들기



라벨 만들기를 진행하기 위해 New label 버튼을 클릭합니다.

Labels can be applied to issues and merge requests to categorize them.

You can also star a label to make it a priority label.



05 GitLab을 더 잘 활용해보자

⑤ 라벨 만들기

New Label

Title 버그

Description 이 라벨이 붙은 이슈는 버그 관련된 이슈입니다.

Background color #ff0000

Choose any color.
Or you can choose one of the suggested colors below



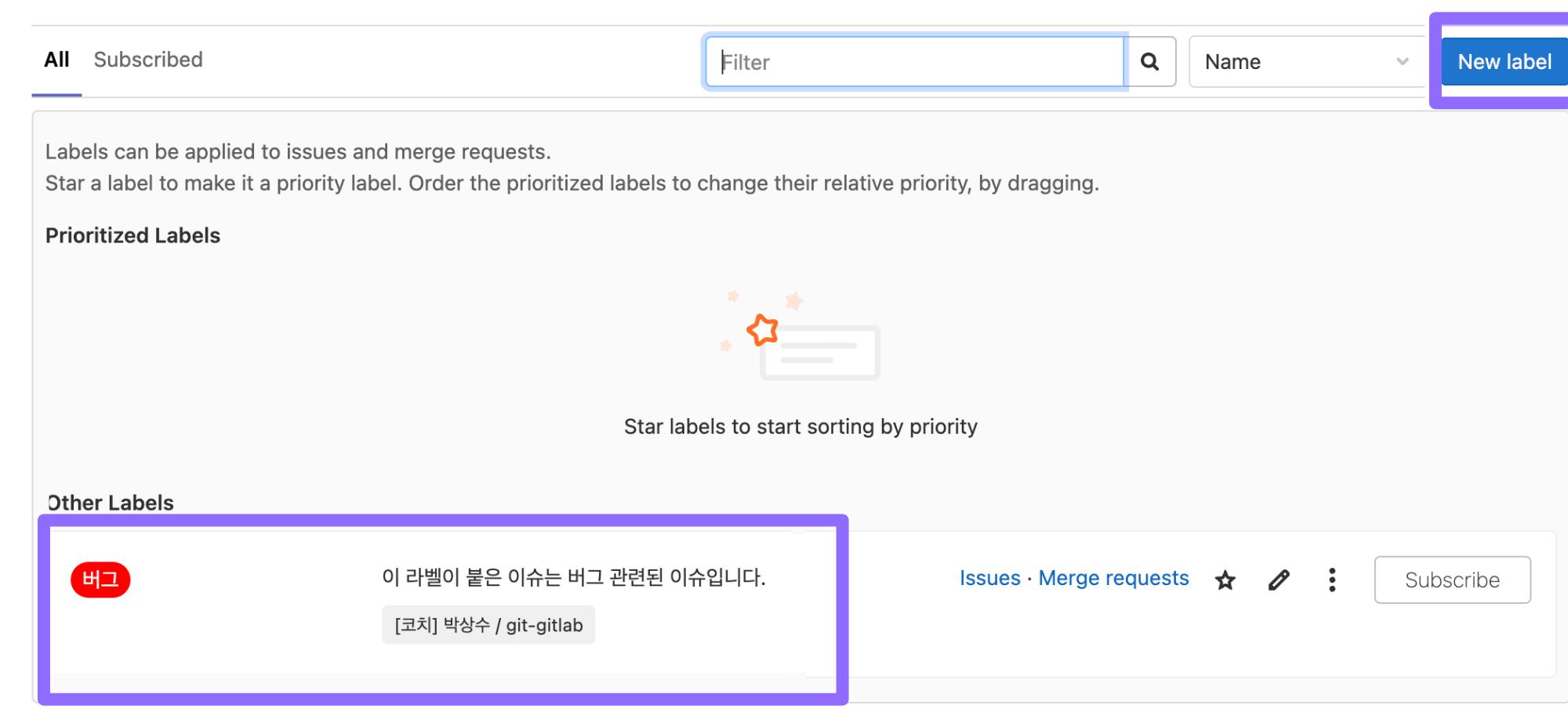
Create label Cancel

Title에는 라벨의 이름을 작성하고, Description에는 라벨에 대한 설명을 작성합니다.

그 후 라벨의 색을 지정하고, Create label을 클릭하여 라벨을 생성합니다.

05 GitLab을 더 잘 활용해보자

⑤ 라벨 만들기



라벨을 성공적으로 생성했다면, 왼쪽 그림과 같이 라벨의 정보가 노출되어야 합니다.

만약 다른 라벨을 활용하고 싶다면, New label 버튼을 클릭하여 라벨을 추가합니다.

05 GitLab을 더 잘 활용해보자

⑤ 마일스톤 만들기

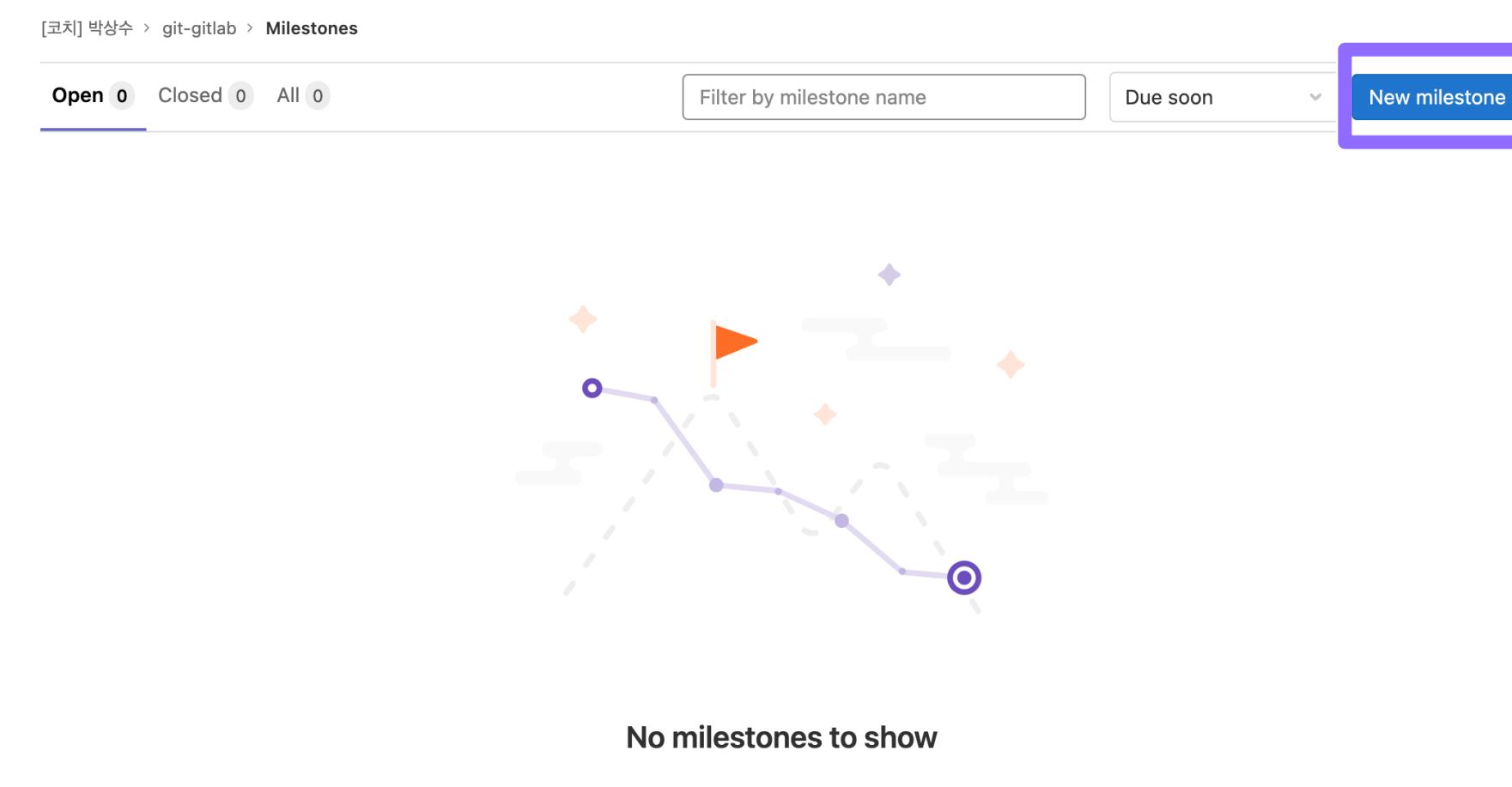
The screenshot shows the GitLab interface for the project 'git-gitlab'. The top navigation bar includes 'Projects', 'Groups', and 'More'. The left sidebar has links for 'Project overview', 'Issues' (0), 'List', 'Boards', 'Labels', 'Service Desk', and 'Milestones'. The 'Milestones' link is highlighted with a purple rectangle. The main content area shows the path '[코치] 박상수 > git-gitlab > Milestones'. Below this are buttons for 'Open 0', 'Closed 0', and 'All 0'.

마일스톤은 이슈를 모아서 관리하는 역할을 하며 기한을 설정하는 기능이 존재하여 마일스톤을 잘 활용한다면 이슈 진행 상황 및 다른 개발자의 업무 진행 현황을 파악하는데 도움이 될 수 있습니다.

마일스톤은 메뉴바에서 Issues를 클릭하면, Issues 목록에서 Milestones를 찾아볼 수 있습니다.

05 GitLab을 더 잘 활용해보자

⑤ 마일스톤 만들기



현재는 마일스톤을 생성하지 않았기 때문에 왼쪽 화면처럼 아무런 정보도 노출되지 않을 겁니다.

마일스톤 생성을 위해 New milestone 버튼을 클릭합니다.

05 GitLab을 더 잘 활용해보자

⑤ 마일스톤 만들기

New Milestone

Title

Description
Write Preview
B I , </>

Start Date Clear start date

Due Date Clear due date

Cancel

Title에서는 목표를 작성합니다. Description에는 목표에 대한 상세 목표를 작성합니다.

Start Date에는 목표 시작 날짜를 작성합니다.

Due Date는 목표를 마무리하는 기한을 정합니다. 모두 작성했다면 Create milestone 버튼을 클릭합니다.

05 GitLab을 더 잘 활용해보자

⑤ 마일스톤 만들기

The screenshot shows the 'Milestones' section of a GitLab project. A new milestone titled '(백엔드) 회원가입 API' has been created. The milestone details are as follows:

- Start date:** Mar 12, 2023 (with an 'Edit' button)
- Due date:** Mar 18, 2023 (Past due) (with an 'Edit' button)
- Issues:** 0 (New issue button)
- Participants:** 0
- Labels:** 0
- Time tracking:** No estimate or time spent
- Merge requests:** 0 (Open: 0, Closed: 0, Merged: 0)
- Releases:** None
- Reference:** ... (with a copy icon)

The main content area displays the milestone title and a note: "Assign some issues to this milestone." Below this, there are three status boxes: "Unstarted Issues (open and unassigned)" (0), "Ongoing Issues (open and assigned)" (0), and "Completed Issues (closed)" (0).

마일스톤 생성이 완료됐다면 왼쪽 화면처럼 출력될 것입니다.

제목, 설명, 시작 날짜와 기한 날짜가 제대로 작성됐다면, 마일스톤을 성공적으로 생성했습니다.

05 GitLab을 더 잘 활용해보자

⑤ 마일스톤 만들기

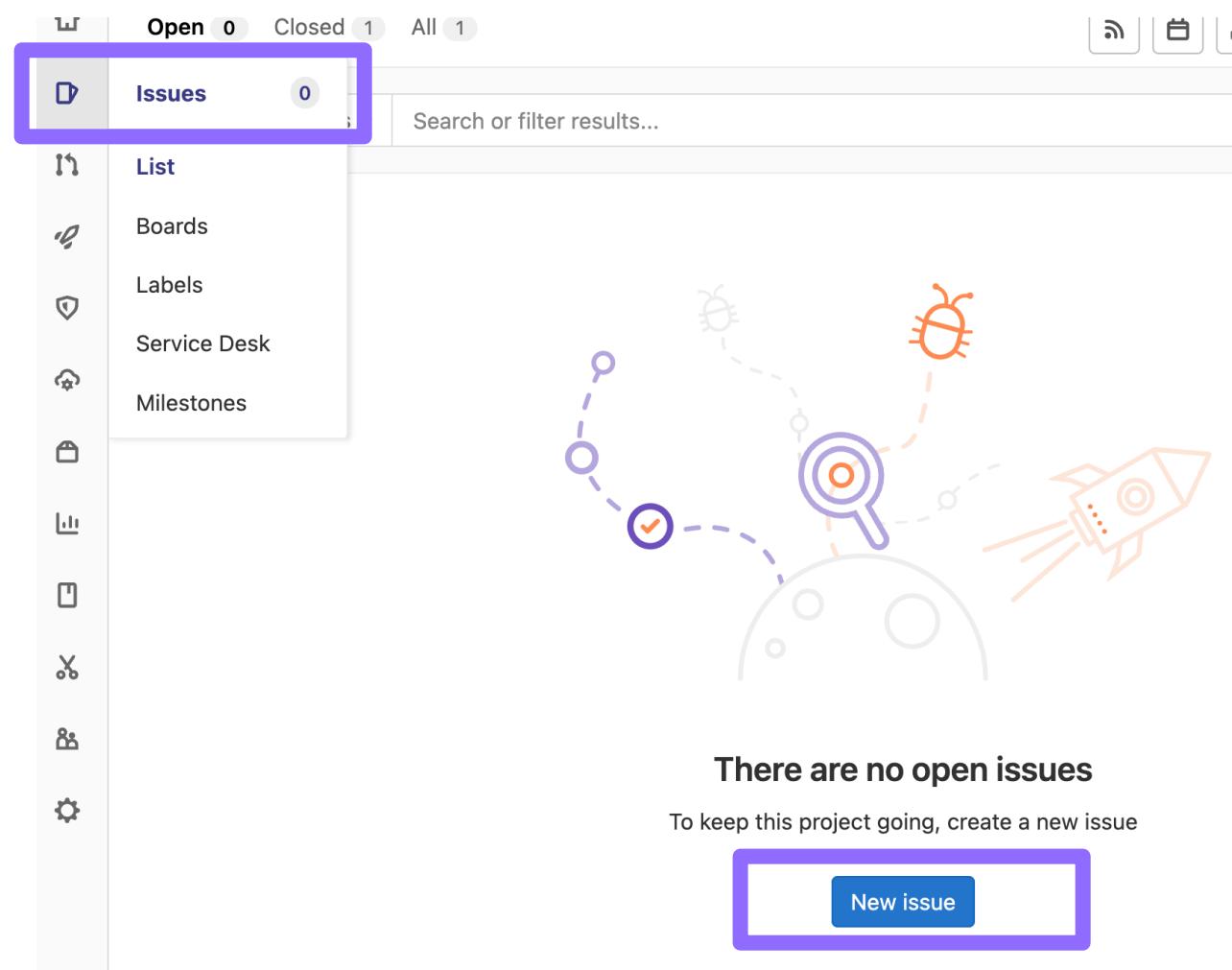
The screenshot shows the 'Milestones' page for a project named 'git-gitlab'. On the left sidebar, the 'Milestones' button is highlighted with a purple rectangle. The main area displays a single milestone card for '(백엔드) 회원가입 API'. The card shows the following details: 'Mar 12, 2023–Mar 18, 2023', 'Expired', and '[코치] 박상수 / git-gitlab'. To the right of the card, there is a summary box containing '0 Issues · 0 Merge requests' and '0% complete'. A blue rectangular box highlights this summary area. At the top of the page, there are filters for 'Open 1', 'Closed 0', 'All 1', 'Filter by milestone name', 'Due soon', and a 'New milestone' button.

마일스톤 목록을 보고 싶다면 왼쪽의 Milestones 버튼을 클릭하면 마일스톤의 목록이 출력됩니다.

마일스톤에 지정된 목표가 얼마나 달성됐는지 수치로도 파악할 수 있습니다.

05 GitLab을 더 잘 활용해보자

⑤ 이슈 만들기



GitLab에서 이슈는 “문제”라는 의미보다 해야 할 일, 해결해야 할 일을 의미합니다.

마일스톤이라는 그릇에 담길, 작은 목표들이 이슈입니다.

그럼 이슈를 만들어보겠습니다. Issues를 클릭한 후, New issue 버튼을 클릭합니다.

05 GitLab을 더 잘 활용해보자

⑤ 이슈 만들기

회원가입 API를 만들어야 한다면, 제목과 설명에는 해야 할 일에 대해 내용을 작성합니다.

The screenshot shows the GitLab issue creation interface. The 'Title' field contains '(백엔드) 회원가입 API 만들기'. The 'Description' field contains '1. 회원가입 API 로직 생성하기'. The 'Assignee' dropdown is set to '박상수'. The 'Milestone' dropdown is set to '(백엔드) 회원가입 API'. The 'Labels' dropdown is set to '버그'. The 'Due date' field is set to '2023-03-12'. A purple box highlights the 'Title' and 'Description' fields. Another purple box highlights the 'Assignee', 'Milestone', and 'Labels' fields. A third purple box highlights the 'Create issue' button at the bottom left.

Assignee는 누가 이 일을 할 것인지 선택합니다.

Milestone은 방금 생성했던 마일스톤을 지정합니다.

라벨은 방금 전 생성했던 버그라는 라벨을 임시로 생성했는데,
여러분들이 원하는 라벨을 생성해서 지정합니다.

Due date는 이 일을 언제까지 할 것인지
날짜를 지정해서 선택한 후 Create issue 버튼을 클릭합니다.

05 GitLab을 더 잘 활용해보자

⑤ 이슈 만들기

The screenshot shows a GitLab interface for creating a new issue. At the top left, there's a breadcrumb navigation: [코치] 박상수 > git-gitlab > Issues > #2. Below it, a green button says "Open" and "Created just now by [코치] 박상수". A purple box highlights the title "백엔드 회원가입 API 만들기" and the sub-instruction "1. 회원가입 API 로직 생성하기". To the right, the main issue creation form is visible, featuring sections for "To Do", "Assignee" (set to "[코치] 박상수 @constP"), "Milestone" (set to "백엔드 회원가입 API (Past due)"), "Time tracking" (empty), "Due date" (set to "Mar 12, 2023 - remove due date"), "Labels" (one labeled "버그"), "Confidentiality" (set to "Not confidential"), "Lock issue" (set to "Unlocked"), and "1 participant" (the user who created the issue). The bottom section has a "Write" button and a text input field for comments or file uploads.

Create issue를 클릭하면, 왼쪽 화면과 같이 출력됩니다.

05 GitLab을 더 잘 활용해보자

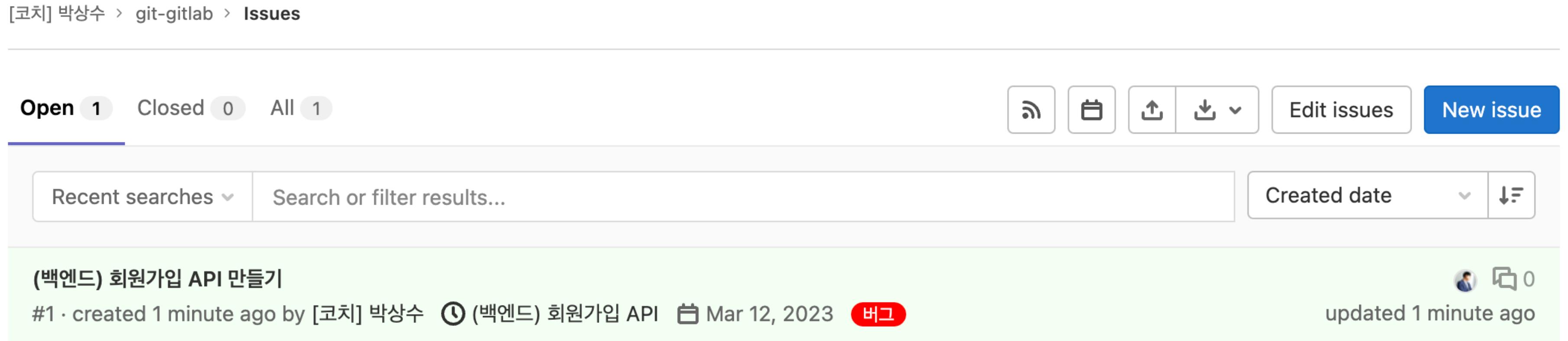
⑤ 이슈 만들기

[코치] 박상수 > git-gitlab > Issues

Open 1 Closed 0 All 1

Recent searches Search or filter results... Created date

(백엔드) 회원가입 API 만들기 0
#1 · created 1 minute ago by [코치] 박상수 🕒 (백엔드) 회원가입 API Mar 12, 2023 버그 updated 1 minute ago



이슈 내용이 정상적으로 저장됐다면 이슈 목록들을 확인해보겠습니다.

목록들을 확인해보면, 해야 할 일들을 한 눈에 살펴볼 수 있습니다.

05 GitLab을 더 잘 활용해보자

⑤ 이슈 완료 처리

[코치] 박상수 > git-gitlab > Issues

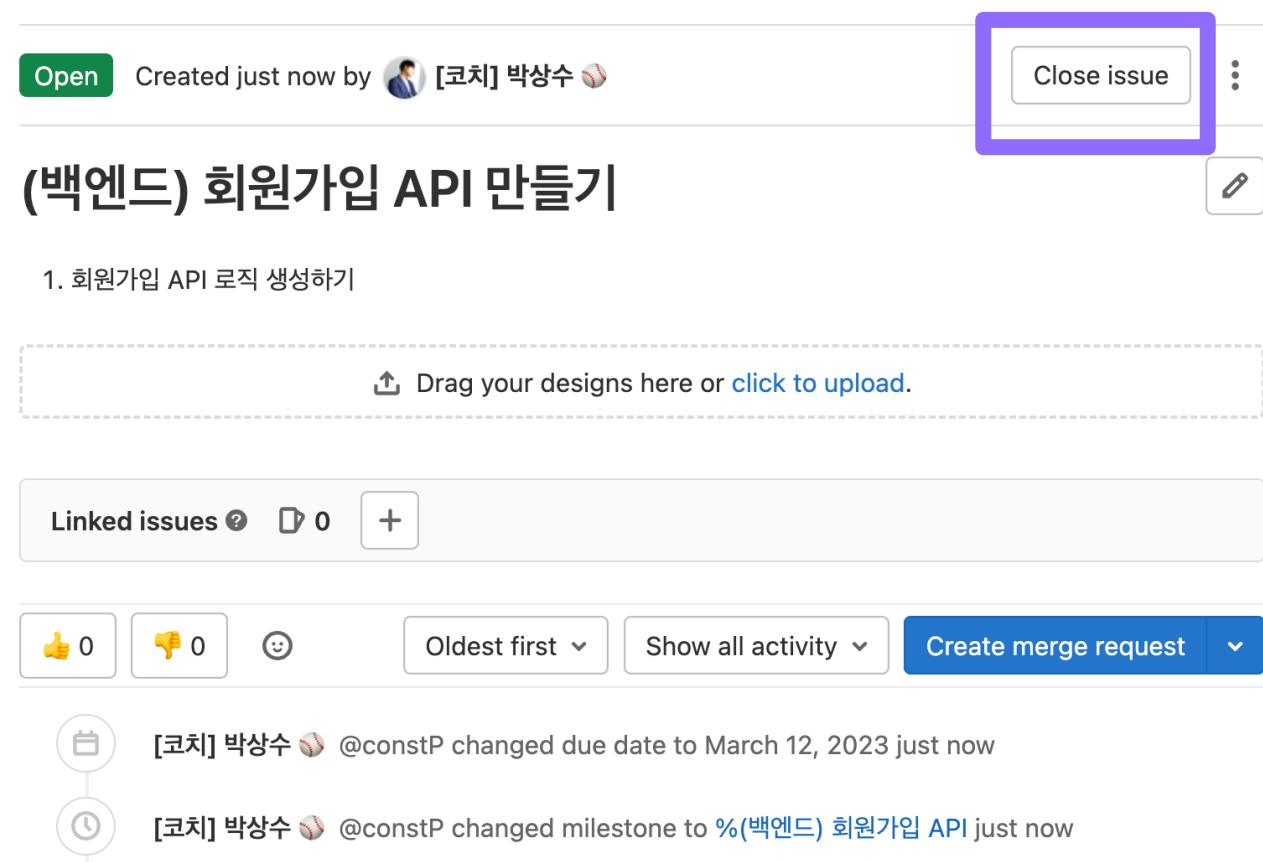
The screenshot shows the 'Issues' page of a GitLab repository. At the top, there are filters for 'Open' (1), 'Closed' (0), and 'All' (1). To the right are buttons for 'Edit issues' and 'New issue'. Below the filters is a search bar with 'Recent searches' and a 'Search or filter results...' field. To the right of the search bar are dropdowns for sorting by 'Created date' and 'Updated date'. A single issue is listed: '#1 · created 1 minute ago by [코치] 박상수'. The issue title is '(백엔드) 회원가입 API 만들기'. It has a status icon (bug), a creation date of 'Mar 12, 2023', and an update message 'updated 1 minute ago'. The issue is marked as a 'bug' and was created by '박상수' on 'Mar 12, 2023'. The last update was 'updated 1 minute ago'.

해야할 일을 모두 처리했다면, 이슈는 완료 처리해야 합니다.

이슈를 완료 처리를 위해 방금 생성한 이슈를 클릭합니다.

05 GitLab을 더 잘 활용해보자

⑤ 이슈 완료 처리



이슈를 클릭했다면, Close issue 버튼을 클릭하여 해당 이슈를 종료합니다.

05 GitLab을 더 잘 활용해보자

⑤ 이슈 완료 처리

The screenshot shows the GitLab milestones interface. At the top, there are filters for 'Open' (1), 'Closed' (0), and 'All' (1) issues. Below these are search fields for 'Filter by milestone name' and 'Due soon', and a blue button for 'New milestone'. A specific milestone card is highlighted with a purple border. The card has the title '(백엔드) 회원가입 API', the date range 'Mar 12, 2023–Mar 18, 2023', and status indicators 'Expired' and '[코치] 박상수 / git-gitlab'. To the right of the card is a progress bar showing '1 Issue · 0 Merge requests' and '100% complete'. A 'Close Milestone' button is also visible.

이슈 생성을 할 때 마일스톤을 연결했었습니다.

이슈 완료 처리를 한 후 마일스톤을 확인해보면, 마일스톤의 프로그레스 바가 변경된 것을 볼 수 있습니다.

05 GitLab을 더 잘 활용해보자

⑤ 이슈 완료 처리

(백엔드) 회원가입 API

1. 회원가입 API로직 구성

All issues for this milestone are closed. You may close this milestone now.

Issues 1 Merge requests 0 Participants 1 Labels 1

Unstarted Issues 0
(open and unassigned)

Ongoing Issues (open 0
and assigned)

Completed Issues 1
(closed)

(백엔드) 회원가입 API 만들기

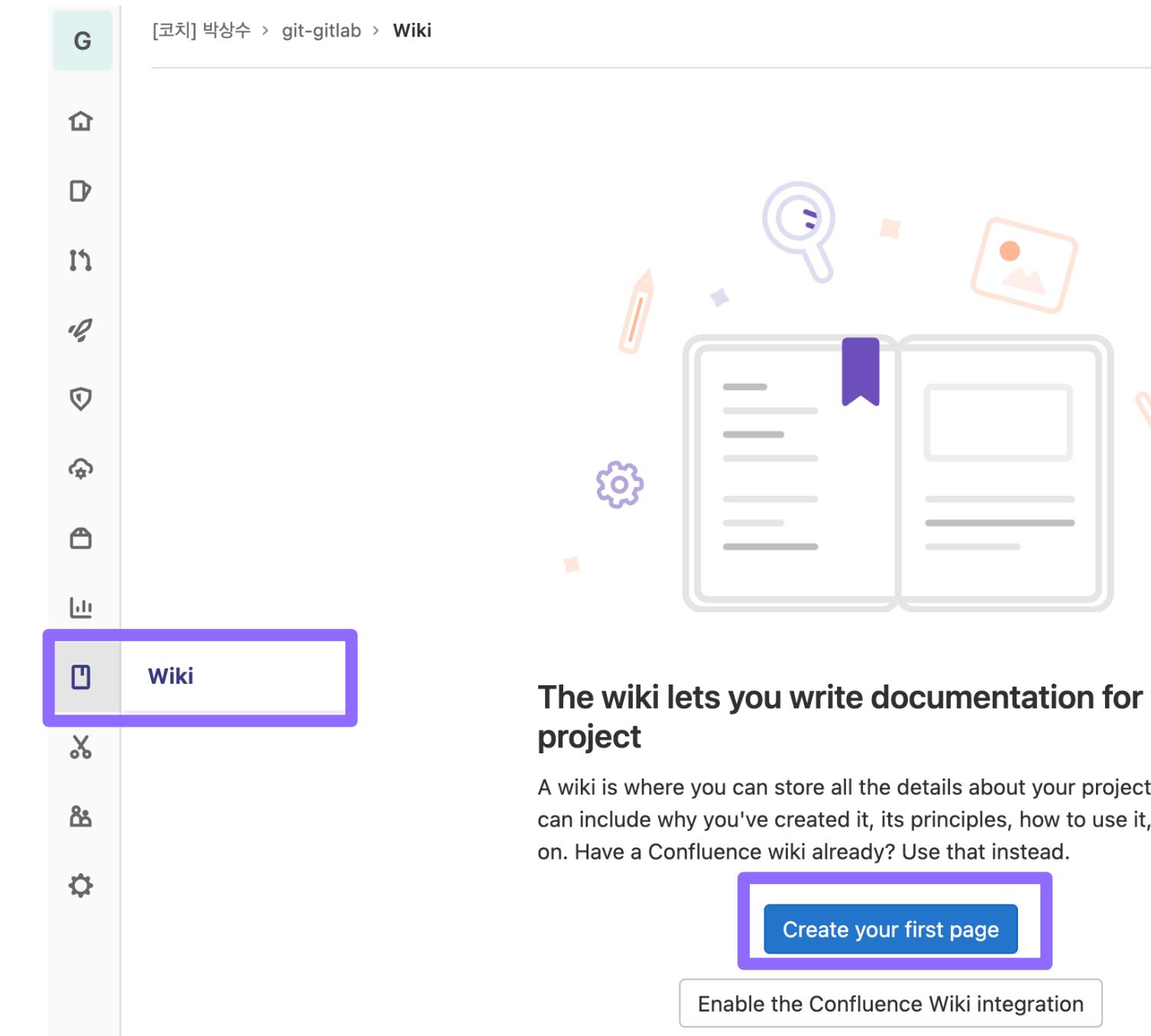
#2 버그

마일스톤을 클릭해서 들어가면, Completed Issues 보드에 생성했던 이슈가 포함되어 있는 것을 볼 수 있습니다.

이슈와 마일스톤을 잘 활용한다면, 다른 개발자와 소통할 때 큰 도움이 될 수 있습니다.

05 GitLab을 더 잘 활용해보자

⑤ Wiki 작성



Wiki는 회의록, 기획안 등 프로젝트 관련 문서들을 두는 공간입니다.

Wiki을 새로 만들어보겠습니다. 왼쪽 탭에서 Wiki을 클릭한 후 Create your first page 버튼을 클릭합니다.

✓ Wiki 템플릿

Wiki 작성은 위해 아래의 스크럼 템플릿을 복사하겠습니다.

2022-NN-NN Scrum

Scrum Rule

- 어제 한 일, 오늘 할 일, 이슈를 간단히 공유해주세요.
- 이슈란에는 아래 내용을 적어주세요.
 - 계획했던 일이 진행되지 못한 이유
 - 막히는 것이 있거나 고민이 있는 사항 등

참석자

@엘리스, @캐터필러, @체셔, @도도새

어제 한 일

- [] Task1 @엘리스, @체셔
- [] Task2 @캐터필러
- [] Task3 @도도새

오늘 할 일

- [] Action1 (Due date: 2022-NN-NN) @캐터필러, @도도새
- [] Action2 (Due date: 2022-NN-NN) @엘리스
- [] Action3 (Due date: 2022-NN-NN) @체셔

이슈 (막히는 것 & 고민 사항)

- Issue1
- Issue2



Wiki 작성

wiki는 기본적으로 markdown 문법

을 활용하여 작성해야 하기 때문에
markdown 문법을 활용하여 문서를
작성합니다.

05 GitLab을 더 잘 활용해보자

⑤ Wiki 작성

Create New Page

Title Tip: You can specify the full path for the new file. We will automatically create any missing directories. [More Information](#).

Format

Content

Write Preview

```
## 2023-NN-NN Scrum

### Scrum Rule
- 어제 한 일, 오늘 할 일, 이슈를 간단히 공유해주세요.
- 이슈란에는 아래 내용을 적어주세요.
  - 계획했던 일이 진행되지 못한 이유
  - 막히는 것이 있거나 고민이 있는 사항 등

---
## 참석자
@엘리스, @캐터필러, @체서, @도도새

## 어제 한 일
- [ ] Task1 @엘리스, @체서
- [ ] Task2 @캐터필러
- [ ] Task3 @도도새

## 오늘 할 일
- [ ] Action1 (Due date: 2022-NN-NN) @캐터필러, @도도새
- [ ] Action2 (Due date: 2022-NN-NN) @엘리스
- [ ] Action3 (Due date: 2022-NN-NN) @체서

## 이슈 (막히는 것 & 고민 사항)
- Issue1
- Issue2
```

Markdown is supported

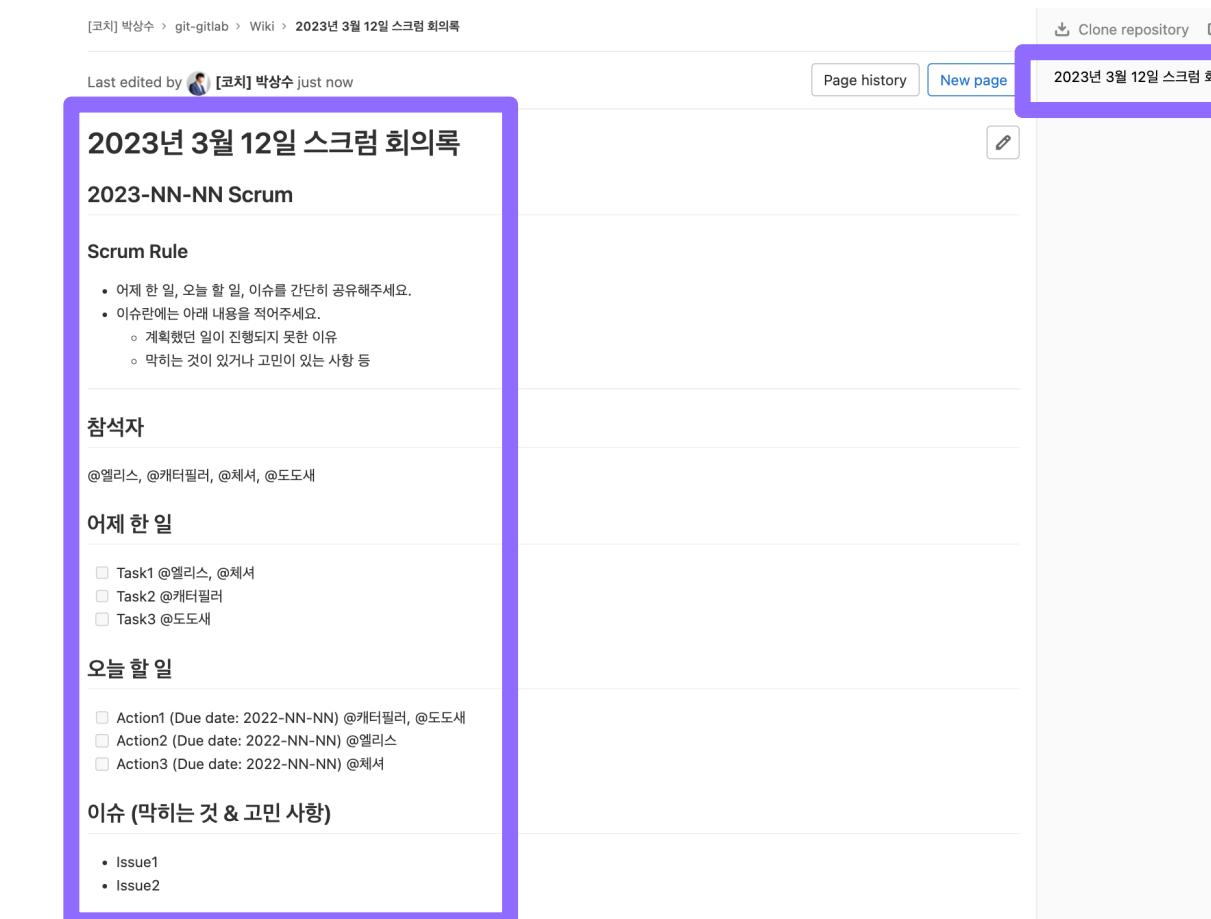
To link to a (new) page, simply type `[Link Title](page-slug)`. More examples are in the [documentation](#).

Commit message

- Wiki 문서의 제목을 작성한 후, Content에는 방금 복사한 스크럼 템플릿을 붙여넣기 해줍니다.
- 문서를 모두 생성했다면 Create page 버튼을 클릭하여 문서를 생성하겠습니다.

05 GitLab을 더 잘 활용해보자

⑤ Wiki 작성



Create page 버튼을 클릭하면 생성된 스크럼과 Wiki 목록을 확인할 수 있습니다.

회의록 내용과 우측에 Wiki 목록이 확인된다면 성공적으로 Wiki 작성은 마무리했습니다.