

# Git

# INDEX

---

- Why Git & Github
- CLI & Markdown
- Git 기본기

# Why Git & Github

# Git

## | Git이란

Git이 뭘까요?

## | Git이란



분산 버전 관리 프로  
그램

## | Git이란



분산 버전 관리 프로  
그램

## | Git이란



**버전** : 컴퓨터 소프트웨어의 특정 상태

**관리** : 어떤 일의 사무, 시설이나 물건의 유지·개량

**프로그램** : 컴퓨터에서 실행될 때 특정 작업을 수행하는 일련의 명령어들의 모음



## | Git이란

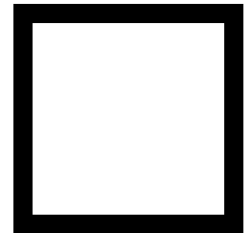


버전 관리 = 컴퓨터 소프트웨어의 특정 상태들을 관리하는 것?

## | Git이란



버전 관리 = 컴퓨터 소프트웨어의 특정 상태들을 관리하는 것?



## 버전관리



우리는 이미 버전 관리를 알고 있다!

## 버전관리



마케팅관리\_레포트\_최종.docx  
마케팅관리\_레포트\_진짜최종.docx  
마케팅관리\_레포트\_리얼최종.docx  
마케팅관리\_레포트\_정말최종.docx  
마케팅관리\_레포트\_진짜진짜최  
종.docx

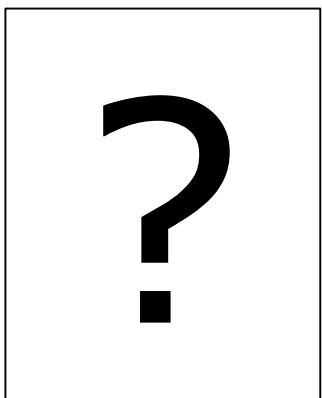
## 버전관리



마케팅관리\_레포트\_최종.docx  
마케팅관리\_레포트\_진짜최종.docx  
마케팅관리\_레포트\_리얼최종.docx  
마케팅관리\_레포트\_정말최종.docx  
마케팅관리\_레포트\_진짜진짜최  
종.docx

어떻게 진짜 최종이지???

## 버전관리

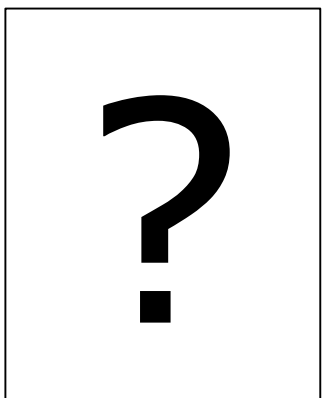


마케팅관리\_레포트\_최종.docx  
마케팅관리\_레포트\_진짜최종.docx  
마케팅관리\_레포트\_리얼최종.docx  
마케팅관리\_레포트\_정말최종.docx  
마케팅관리\_레포트\_진짜진짜최  
종.docx

## 버전관리



파일에 날짜와 시간을 적어봐!



마케팅관리\_레포트\_최종.docx

마케팅관리\_레포트\_진짜최종.docx

마케팅관리\_레포트\_리얼최종.docx

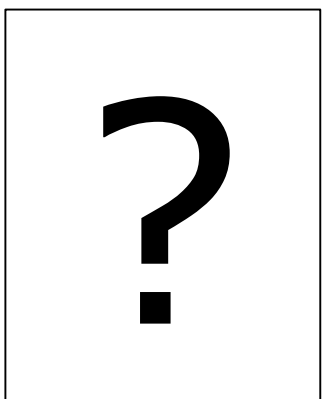
마케팅관리\_레포트\_정말최종.docx

마케팅관리\_레포트\_진짜진짜최  
종.docx

## 버전관리



파일에 날짜와 시간을 적어봐!



마케팅관리\_레포트\_211201\_1604.docx

마케팅관리\_레포트\_211202\_1215.docx

마케팅관리\_레포트\_211203\_0530.docx

마케팅관리\_레포트\_211204\_0921.docx

마케팅관리\_레포트\_211205\_1305.docx



## 버전관리



마케팅관리\_레포트\_211201\_1604.docx  
마케팅관리\_레포트\_211202\_1215.docx  
마케팅관리\_레포트\_211203\_0530.docx  
마케팅관리\_레포트\_211204\_0921.docx  
마케팅관리\_레포트\_211205\_1305.docx

**가정1** : 만약 레포트 1개 당 1000장이라면?

## 버전관리

2000장을 언제 다 보냐...

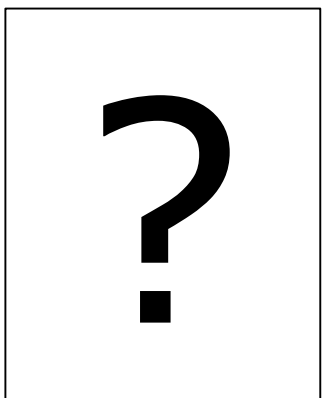


마케팅관리\_레포트\_211201\_1604.docx  
마케팅관리\_레포트\_211202\_1215.docx  
마케팅관리\_레포트\_211203\_0530.docx  
마케팅관리\_레포트\_211204\_0921.docx  
마케팅관리\_레포트\_211205\_1305.docx

뭐를  
수정  
했더라?

**가정1** : 만약 레포트 1개 당 1000장이라면?

## 버전관리



2000장을 언제 다 보냐...



마케팅관리\_레포트\_211201\_1604.docx  
마케팅관리\_레포트\_211202\_1215.docx  
마케팅관리\_레포트\_211203\_0530.docx  
마케팅관리\_레포트\_211204\_0921.docx  
마케팅관리\_레포트\_211205\_1305.docx

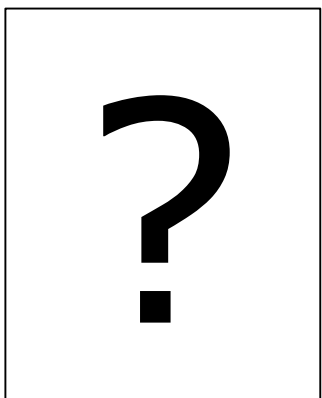
뭐를  
수정  
했더라?

**가정1** : 만약 레포트 1개 당 1000장이라면?

## 버전관리



변경사항을 기록하는 파일을 만들어봐!



마케팅관리\_레포트\_211201\_1604.docx

마케팅관리\_레포트\_211202\_1215.docx

마케팅관리\_레포트\_211203\_0530.docx

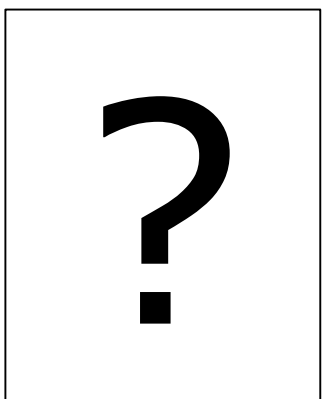
마케팅관리\_레포트\_211204\_0921.docx

마케팅관리\_레포트\_211205\_1305.docx

**가정1** : 만약 레포트 1개 당 1000장이라면?

## 버전 관리

변경사항을 기록하는 파일을 만들어봐!



**가정1** : 만약 레포트 1개 당 1000장이라면?

마케팅관리\_레포트\_211201\_1604.docx

마케팅관리\_레포트\_211202\_1215.docx

마케팅관리\_변경사항\_ 211202\_1215.docx

마케팅관리\_레포트\_211203\_0530.docx

마케팅관리\_변경사항\_ 211203\_0530.docx

마케팅관리\_레포트\_211204\_0921.docx

마케팅관리\_변경사항\_ 211204\_0921.docx

마케팅관리\_레포트\_211205\_1305.docx

마케팅관리\_변경사항\_ 211205\_1305.docx

## 버전 관리



**가정1** : 만약 레포트 1개 당 1000장이라면?

**가정2** : 만약 레포트 1개당 10억장이라면?

마케팅관리\_레포트\_211201\_1604.docx

마케팅관리\_레포트\_211202\_1215.docx

마케팅관리\_변경사항\_ 211202\_1215.docx

마케팅관리\_레포트\_211203\_0530.docx

마케팅관리\_변경사항\_ 211203\_0530.docx

마케팅관리\_레포트\_211204\_0921.docx

마케팅관리\_변경사항\_ 211204\_0921.docx

마케팅관리\_레포트\_211205\_1305.docx

마케팅관리\_변경사항\_ 211205\_1305.docx

## 버전 관리

용량 너무 크다...



**가정1** : 만약 레포트 1개 당 1000장이라면?

**가정2** : 만약 레포트 1개당 10억장이라면?

마케팅관리\_레포트\_211201\_1604.docx

마케팅관리\_레포트\_211202\_1215.docx

마케팅관리\_변경사항\_ 211202\_1215.docx

마케팅관리\_레포트\_211203\_0530.docx

마케팅관리\_변경사항\_ 211203\_0530.docx

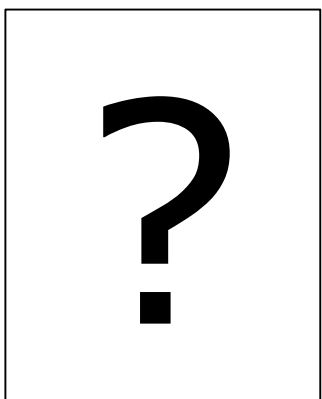
마케팅관리\_레포트\_211204\_0921.docx

마케팅관리\_변경사항\_ 211204\_0921.docx

마케팅관리\_레포트\_211205\_1305.docx

마케팅관리\_변경사항\_ 211205\_1305.docx

## 버전 관리



용량 너무 크다...



**가정1** : 만약 레포트 1개 당 1000장이라면?

**가정2** : 만약 레포트 1개당 10억장이라면?

마케팅관리\_레포트\_211201\_1604.docx

마케팅관리\_레포트\_211202\_1215.docx

마케팅관리\_변경사항\_ 211202\_1215.docx

마케팅관리\_레포트\_211203\_0530.docx

마케팅관리\_변경사항\_ 211203\_0530.docx

마케팅관리\_레포트\_211204\_0921.docx

마케팅관리\_변경사항\_ 211204\_0921.docx

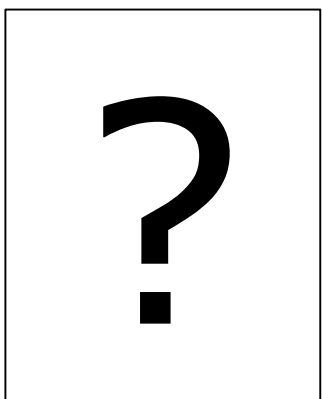
마케팅관리\_레포트\_211205\_1305.docx

마케팅관리\_변경사항\_ 211205\_1305.docx



## 버전 관리

맨 나중 파일과, 이전 변경사항만 남겨



**가정1** : 만약 레포트 1개 당 1000장이라면?

**가정2** : 만약 레포트 1개당 10억장이라면?

마케팅관리\_레포트\_211201\_1604.docx

마케팅관리\_레포트\_211202\_1215.docx

마케팅관리\_변경사항\_ 211202\_1215.docx

마케팅관리\_레포트\_211203\_0530.docx

마케팅관리\_변경사항\_ 211203\_0530.docx

마케팅관리\_레포트\_211204\_0921.docx

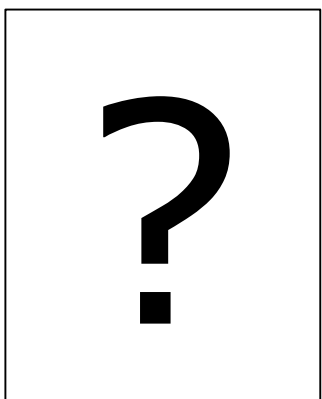
마케팅관리\_변경사항\_ 211204\_0921.docx

마케팅관리\_레포트\_211205\_1305.docx

마케팅관리\_변경사항\_ 211205\_1305.docx

## 버전 관리

맨 나중 파일과, 이전 변경사항만 남겨



**가정1** : 만약 레포트 1개 당 1000장이라면?

**가정2** : 만약 레포트 1개당 10억장이라면?

~~마케팅관리\_레포트\_211201\_1604.docx~~

~~마케팅관리\_레포트\_211202\_1215.docx~~

마케팅관리\_변경사항\_ 211202\_1215.docx

~~마케팅관리\_레포트\_211203\_0530.docx~~

마케팅관리\_변경사항\_ 211203\_0530.docx

~~마케팅관리\_레포트\_211204\_0921.docx~~

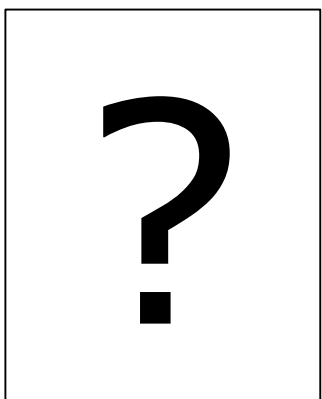
마케팅관리\_변경사항\_ 211204\_0921.docx

마케팅관리\_레포트\_211205\_1305.docx

마케팅관리\_변경사항\_ 211205\_1305.docx

## 버전관리

맨 나중 파일과, 이전 변경사항만 남겨



마케팅관리\_변경사항\_ 211202\_1215.docx

마케팅관리\_변경사항\_ 211203\_0530.docx

마케팅관리\_변경사항\_ 211204\_0921.docx

마케팅관리\_변경사항\_ 211205\_1305.docx

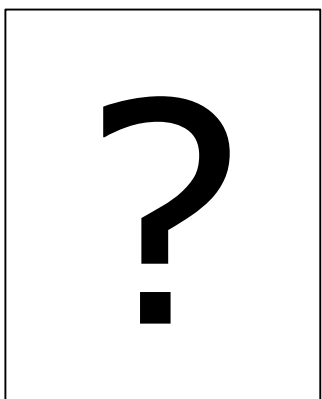
마케팅관리\_레포트\_211205\_1305.docx

**가정1** : 만약 레포트 1개 당 1000장이라면?  
면?

**가정2** : 만약 레포트 1개당 10억장이라면?

## 버전관리

맨 나중 파일과, 이전 변경사항만 남겨



마케팅관리\_변경사항\_ 211202\_1215.docx

마케팅관리\_변경사항\_ 211203\_0530.docx

마케팅관리\_변경사항\_ 211204\_0921.docx

마케팅관리\_변경사항\_ 211205\_1305.docx

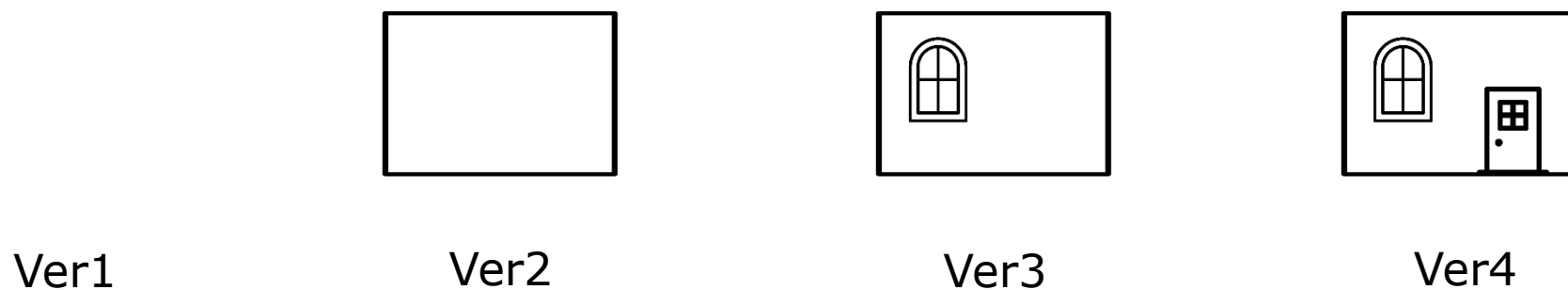
마케팅관리\_레포트\_211205\_1305.docx

가정1 : 만약 레포트 1개 당 1000장이라면?

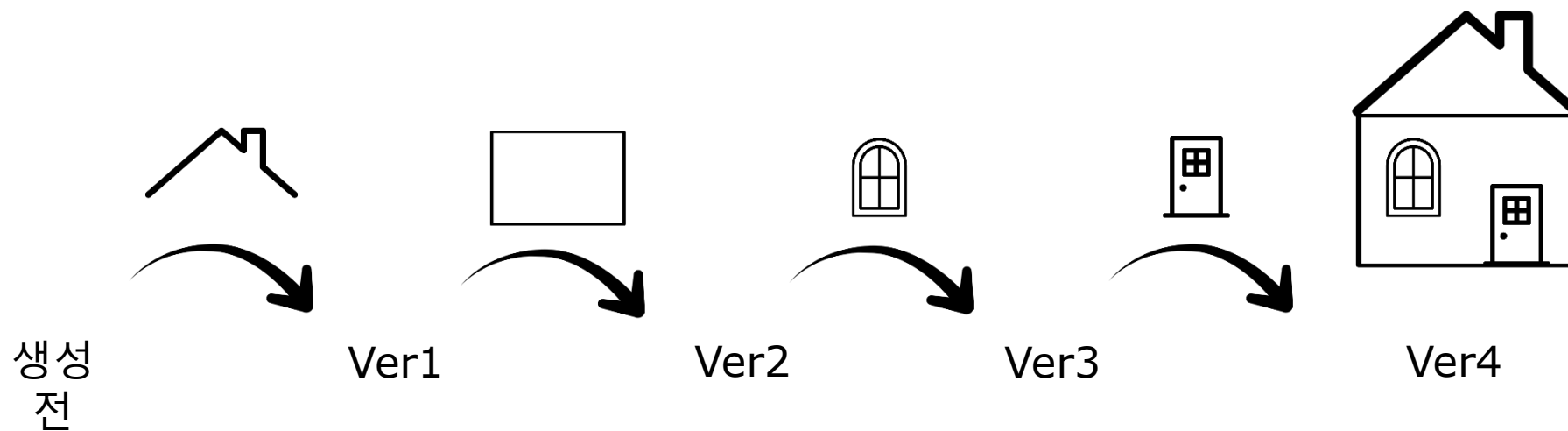
가정2 : 만약 레포트 1개당 10억장이라면?

# 버전관리

기존



개선



## 버전 관리


 난 소프트웨어라서 빠르니까 내가 해줄게!

버전 관리  
프로그램



마케팅관리\_변경사항\_ 211202\_1215.docx

마케팅관리\_변경사항\_ 211203\_0530.docx

마케팅관리\_변경사항\_ 211204\_0921.docx 

마케팅관리\_변경사항\_ 211205\_1305.docx

마케팅관리\_레포트\_211205\_1305.docx

## 버전 관리



난 소프트웨어라서 빠르니까 내가 해줄게!

버전 관리  
프로그램



마케팅관리\_변경사항\_ 211202\_1215.docx  
마케팅관리\_변경사항\_ 211203\_0530.docx  
마케팅관리\_변경사항\_ 211204\_0921.docx  
마케팅관리\_변경사항\_ 211205\_1305.docx

마케팅관리\_레포트\_211205\_1305.docx

2021년 12월 05일 13시 05분 수정  
작성자: 홍길동  
위치: 637번째 장, 12번째 줄  
내용: "있었습니다" -> "있습니다"  
이유: 현재형을 과거형으로 잘못 썼음

## | Git이란



분산 버전 관리 프로  
그램



## Git이란

마케팅관리\_레포트\_211201\_1604.docx

마케팅관리\_레포트\_211202\_1215.docx

마케팅관리\_변경사항\_

211202\_1215.docx

마케팅관리\_레포트\_211203\_0530.docx

마케팅관리\_변경사항\_

211203\_0530.docx

마케팅관리\_레포트\_211204\_0921.docx

마케팅관리\_변경사항\_

211204\_0921.docx



마케팅관리\_변경사항\_

211202\_1215.docx

마케팅관리\_변경사항\_

211203\_0530.docx

마케팅관리\_변경사항\_

211204\_0921.docx

마케팅관리\_변경사항\_

211205\_1305.docx

마케팅관리\_레포트\_211205\_1305.docx

## Git이란



### 분산 버전 관리 프로 그램

- 코드의 **히스토리(버전)**을 관리하는 도구
- 개발되어온 **과정** 파악 가능
- 이전 버전과의 **변경 사항 비교 및 분석**

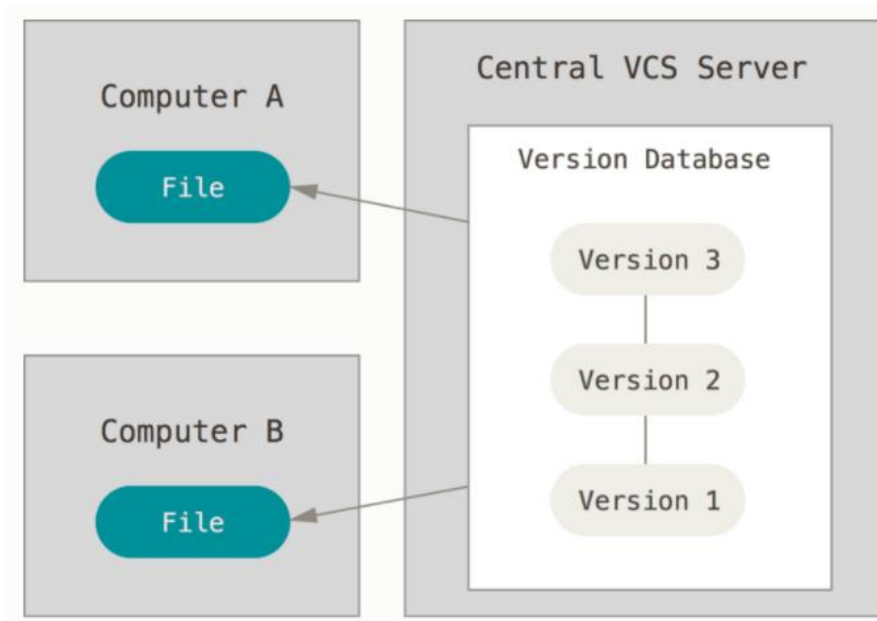
## | Git이란



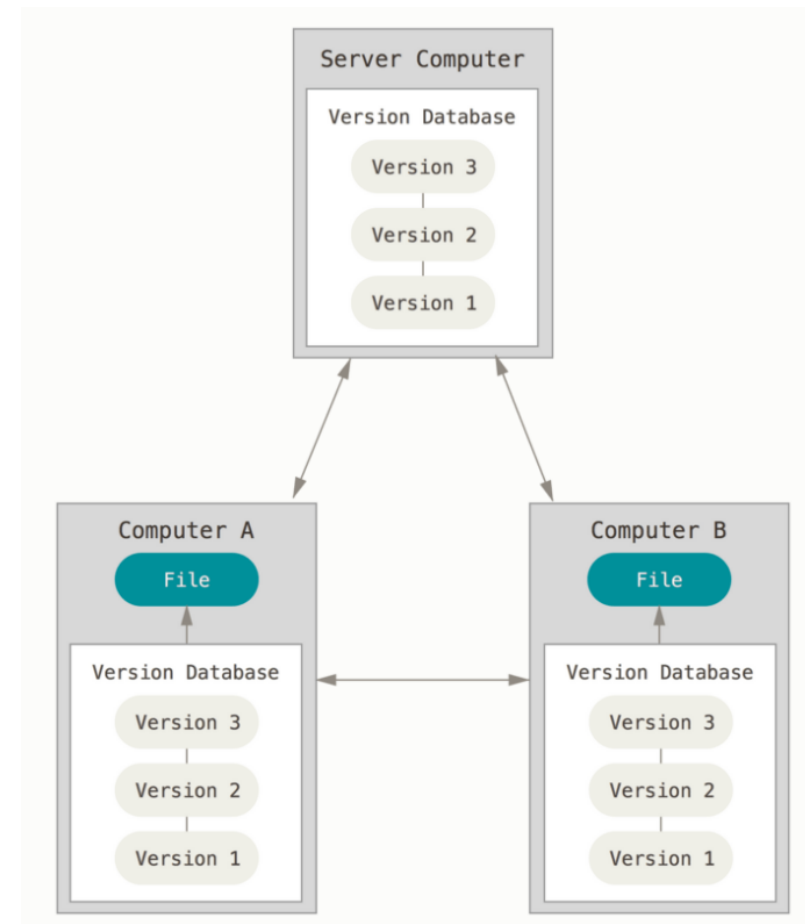
분산 버전 관리 프로  
그램

## Git이란

중앙 집중식 버전 관리

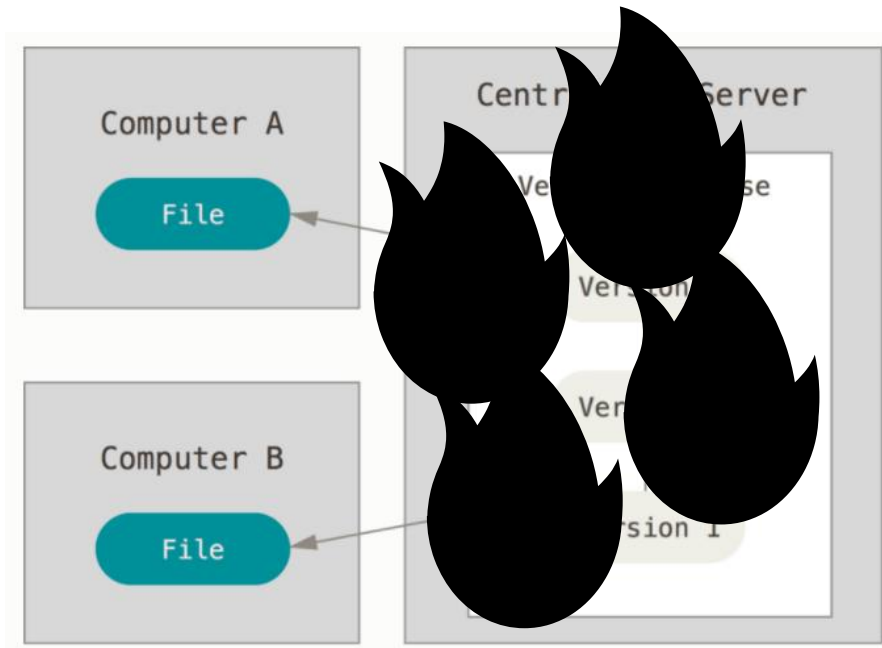


분산 버전 관리

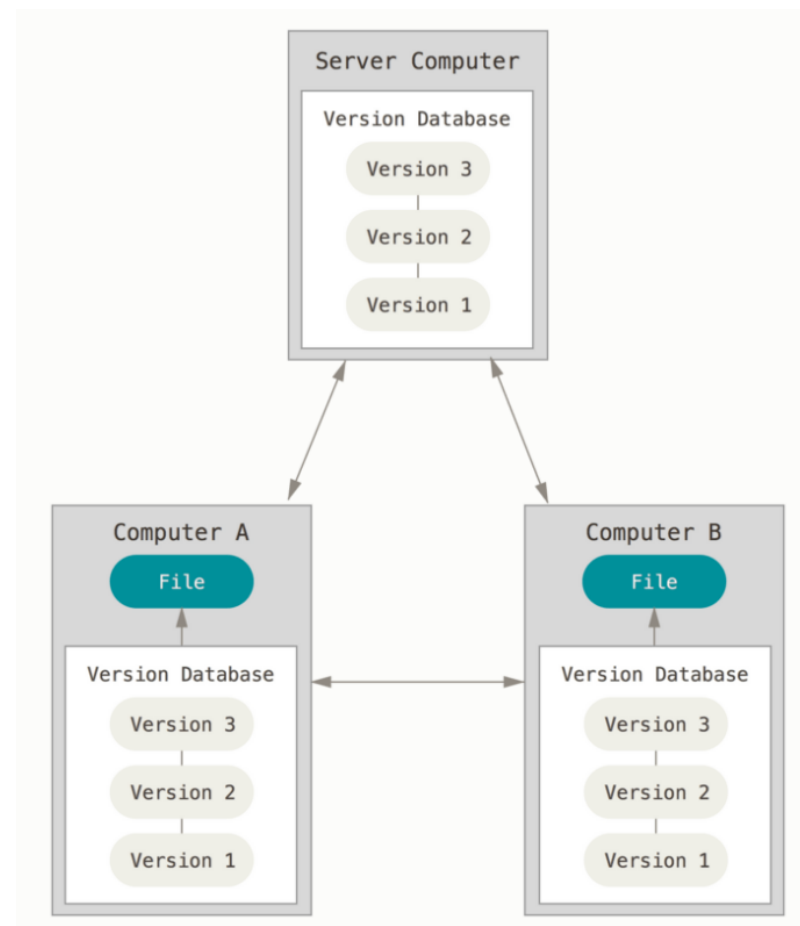


## Git이란

중앙 집중식 버전 관리

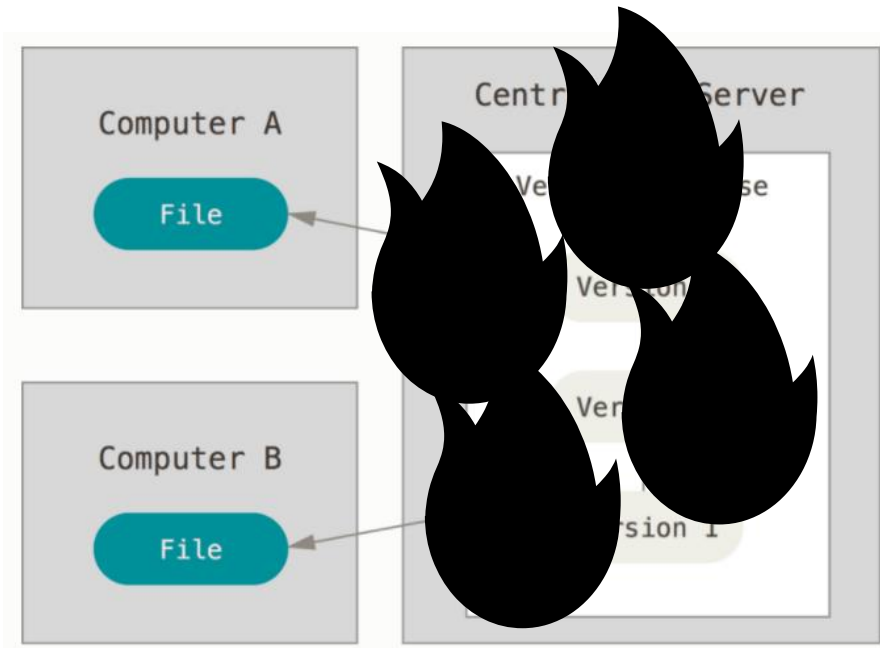


분산 버전 관리

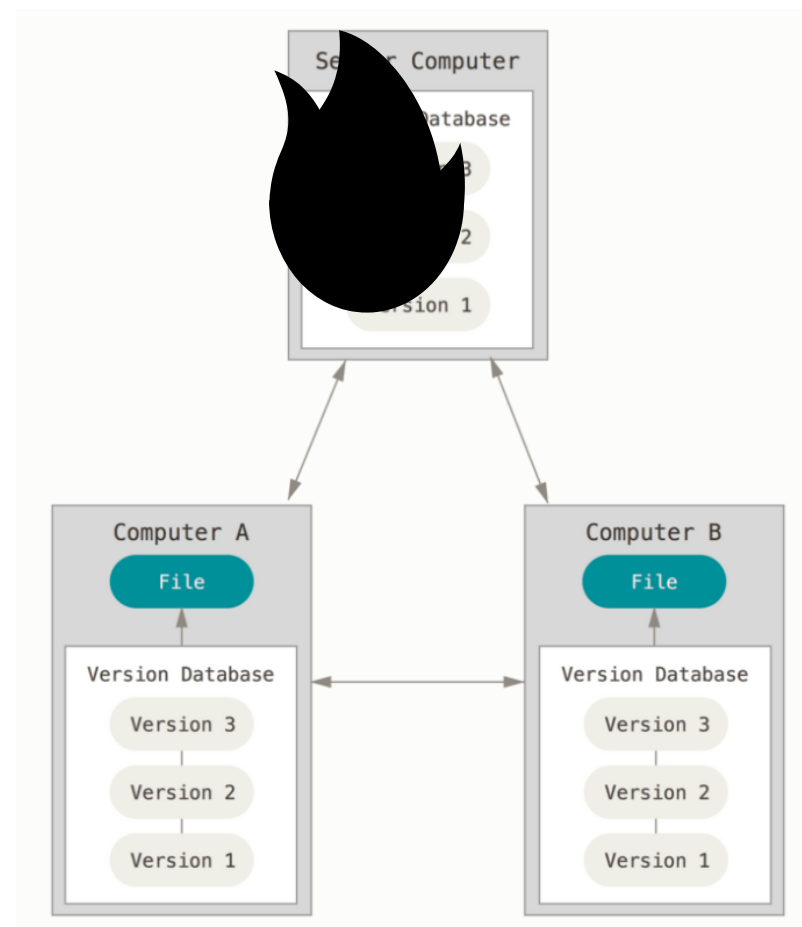


## Git이란

중앙 집중식 버전 관리

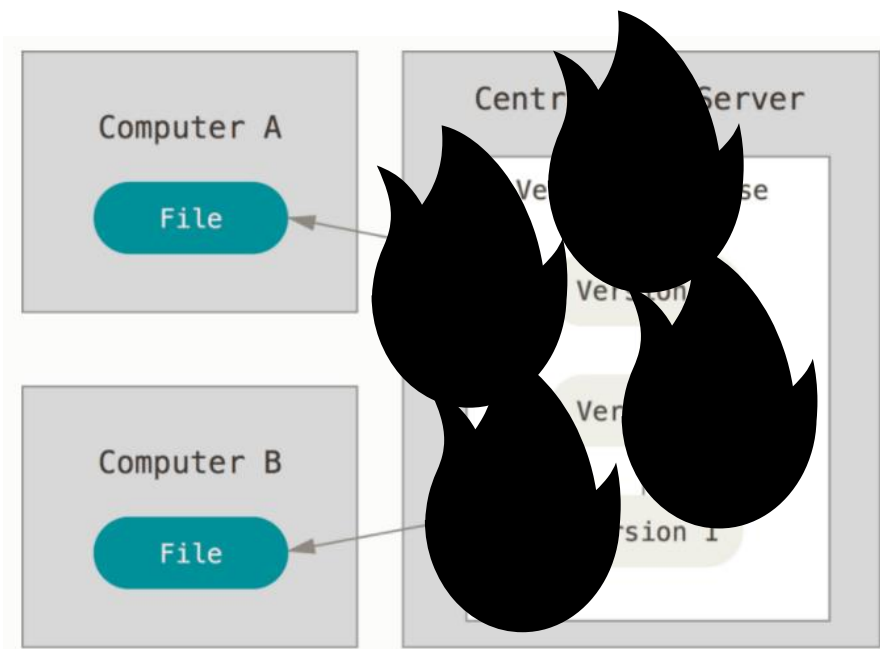


분산 버전 관리

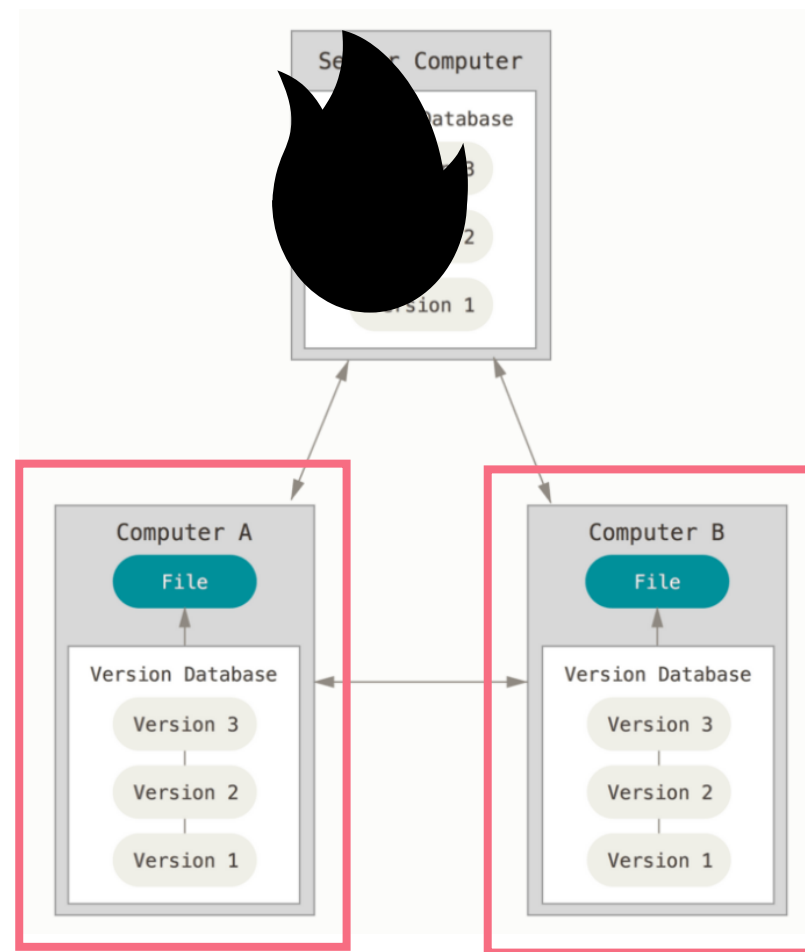


## Git이란

중앙 집중식 버전 관리

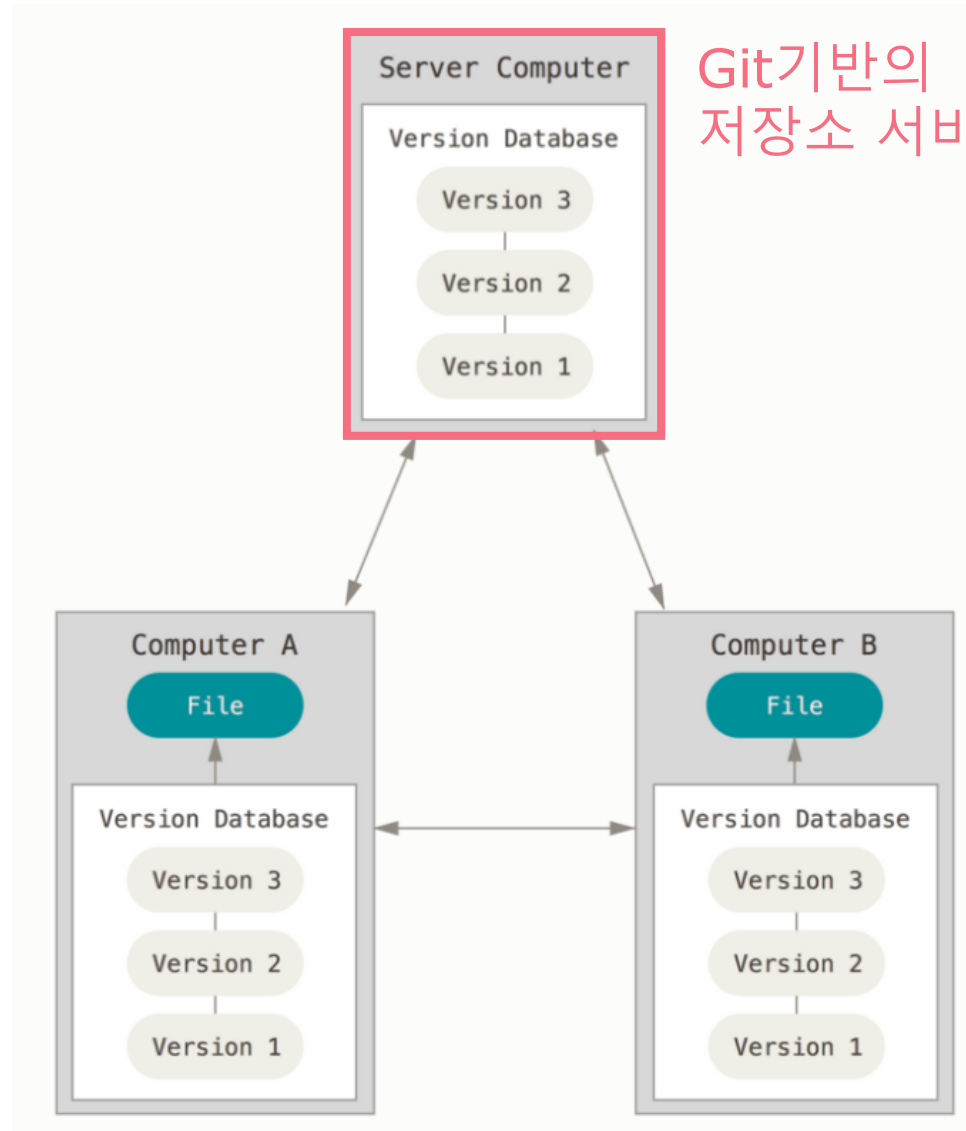


분산 버전 관리



안전하게 살아있다!

# Git이란



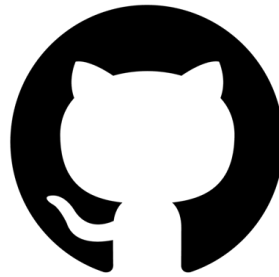
Git기반의  
저장소 서비스를 제공하는 서버



## | Git이란



GitLab



GitHub



Bitbucket

## | Git이란



분산 버전 관리 프로그램

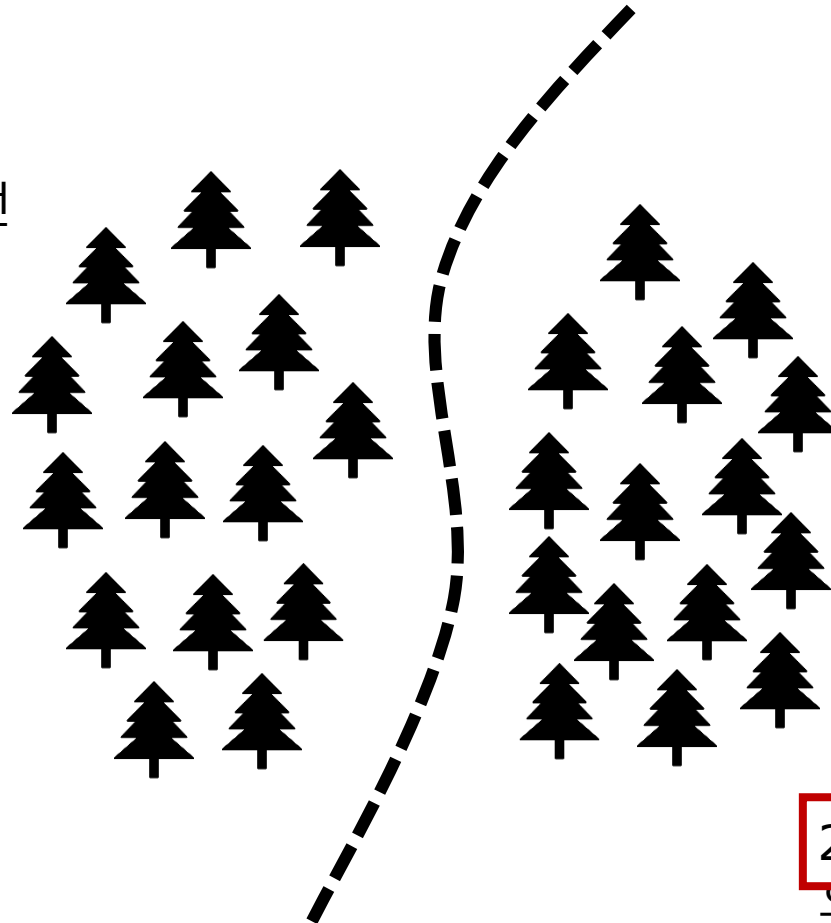


Git기반의 저장소 서비스

# Github

## | Github을 사용하면 뭐가 좋을까?

1. Git을 이용한 버전  
관리



2. Github를 이용한 포트폴리

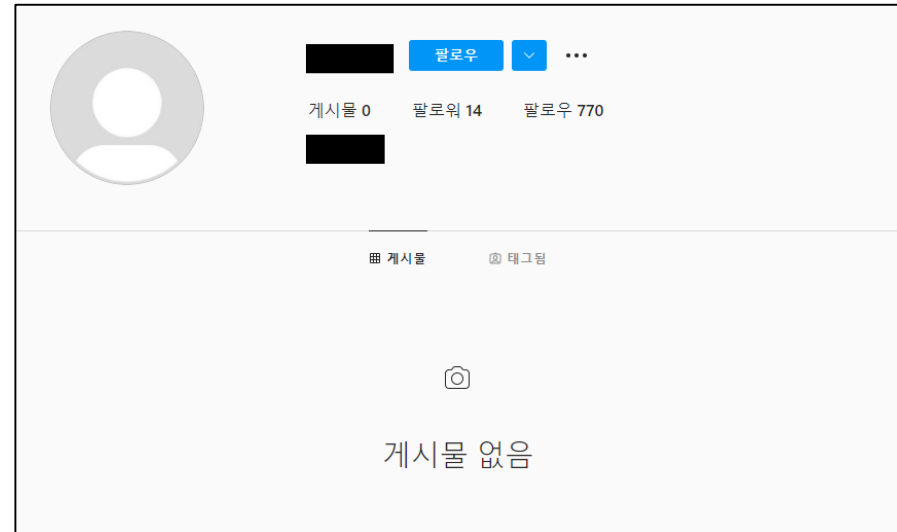
오

| Github을 사용하면 뭐가 좋을까?

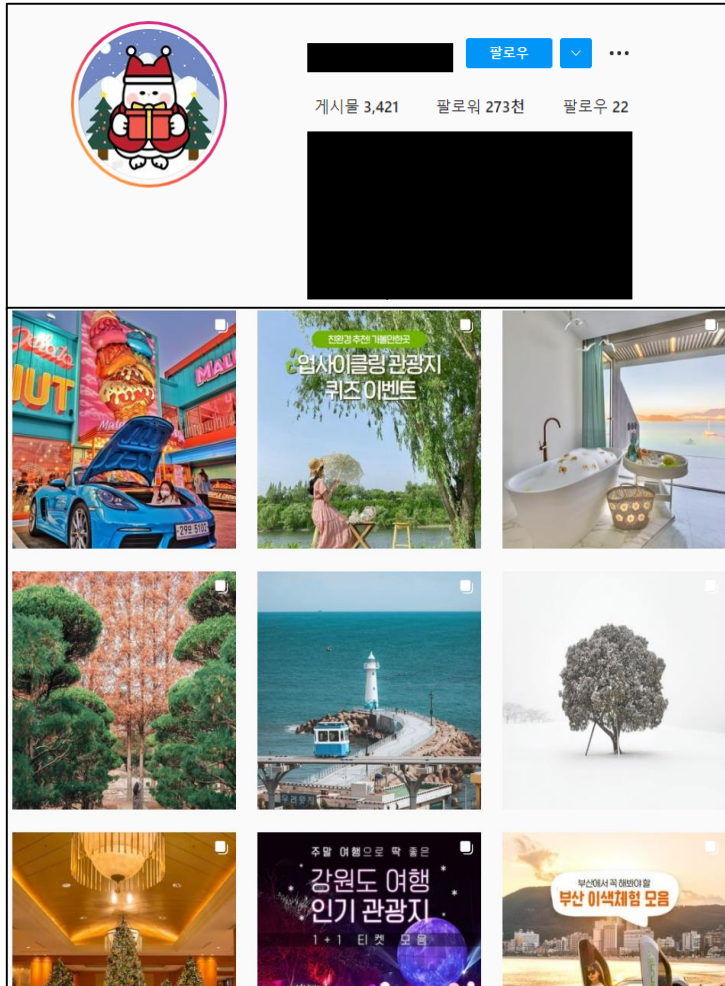
여러분은 대형 여행사의 인사팀장입니다.

다음 두 지원자 중 어떤 분을 채용하시겠습니까?

## Github을 사용하면 뭐가 좋을까?



## Github을 사용하면 뭐가 좋을까?



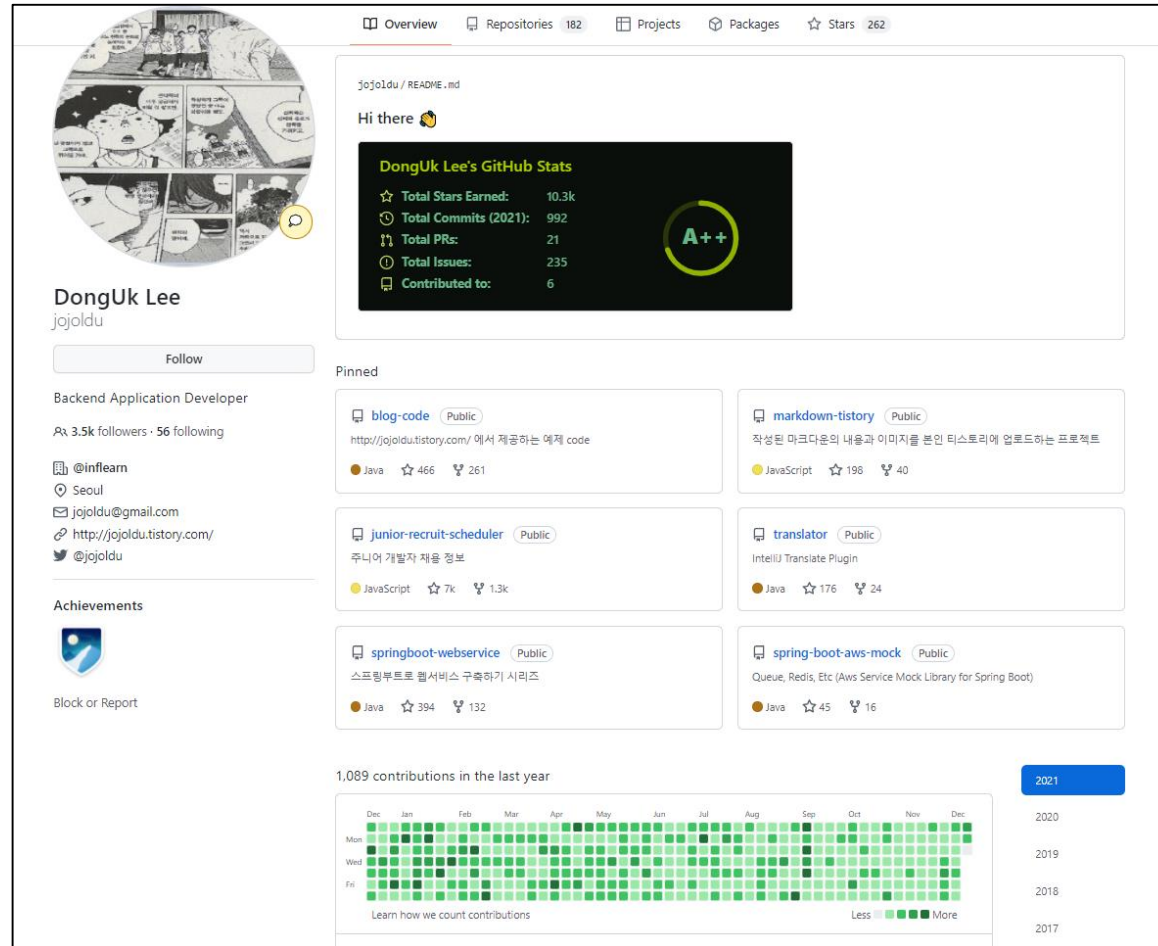
왜일까요?

1. 높은 팔로워, 팔로우
2. 3421개의 많은 게시물
3. 깔끔하고 예쁜 게시물



여행에 대한 **열정, 성실함, 홍보 능력**

## Github을 사용하면 뭐가 좋을까?



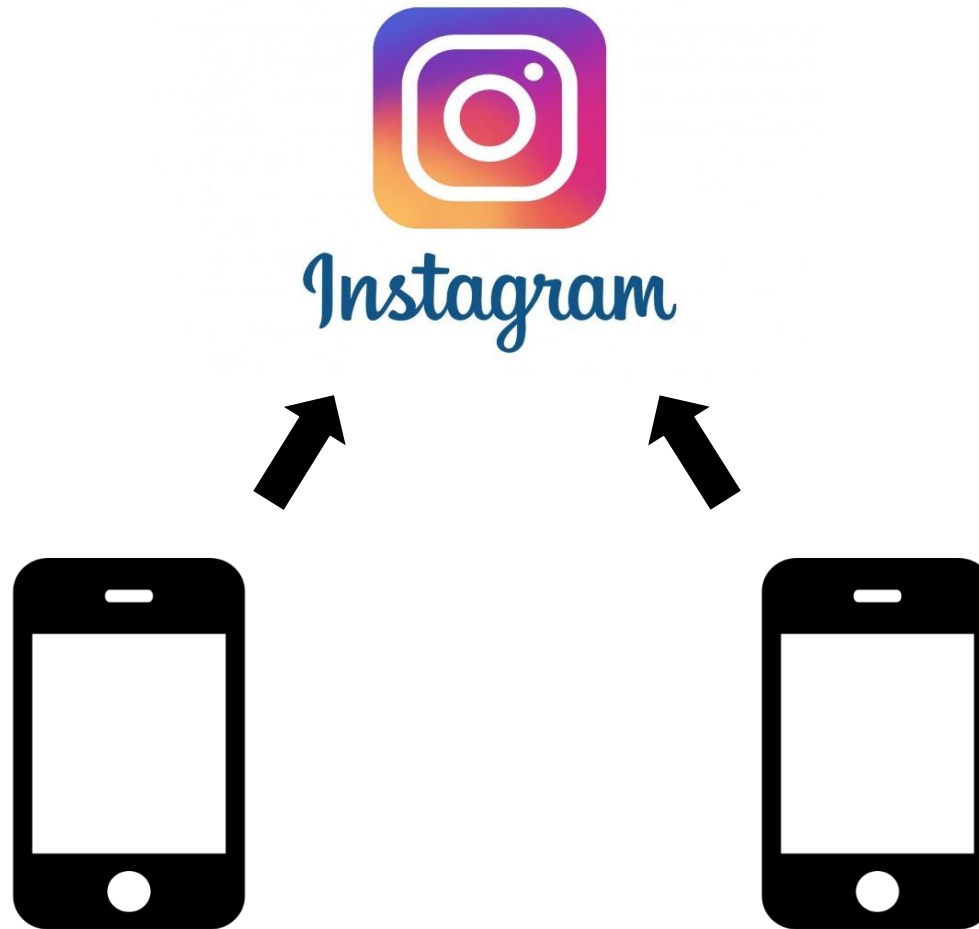
<https://github.com/jojoldu>



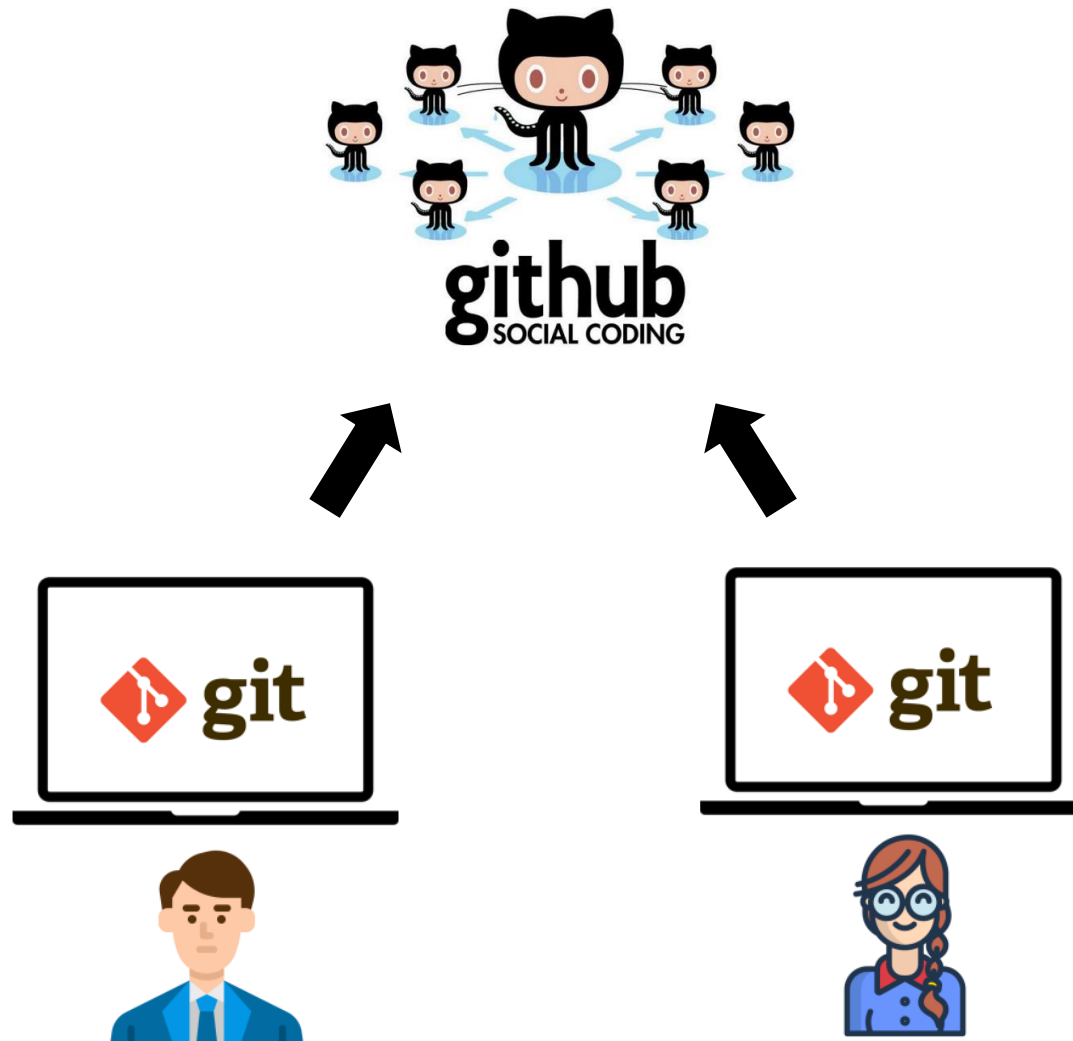
| Github을 사용하면 뭐가 좋을까?



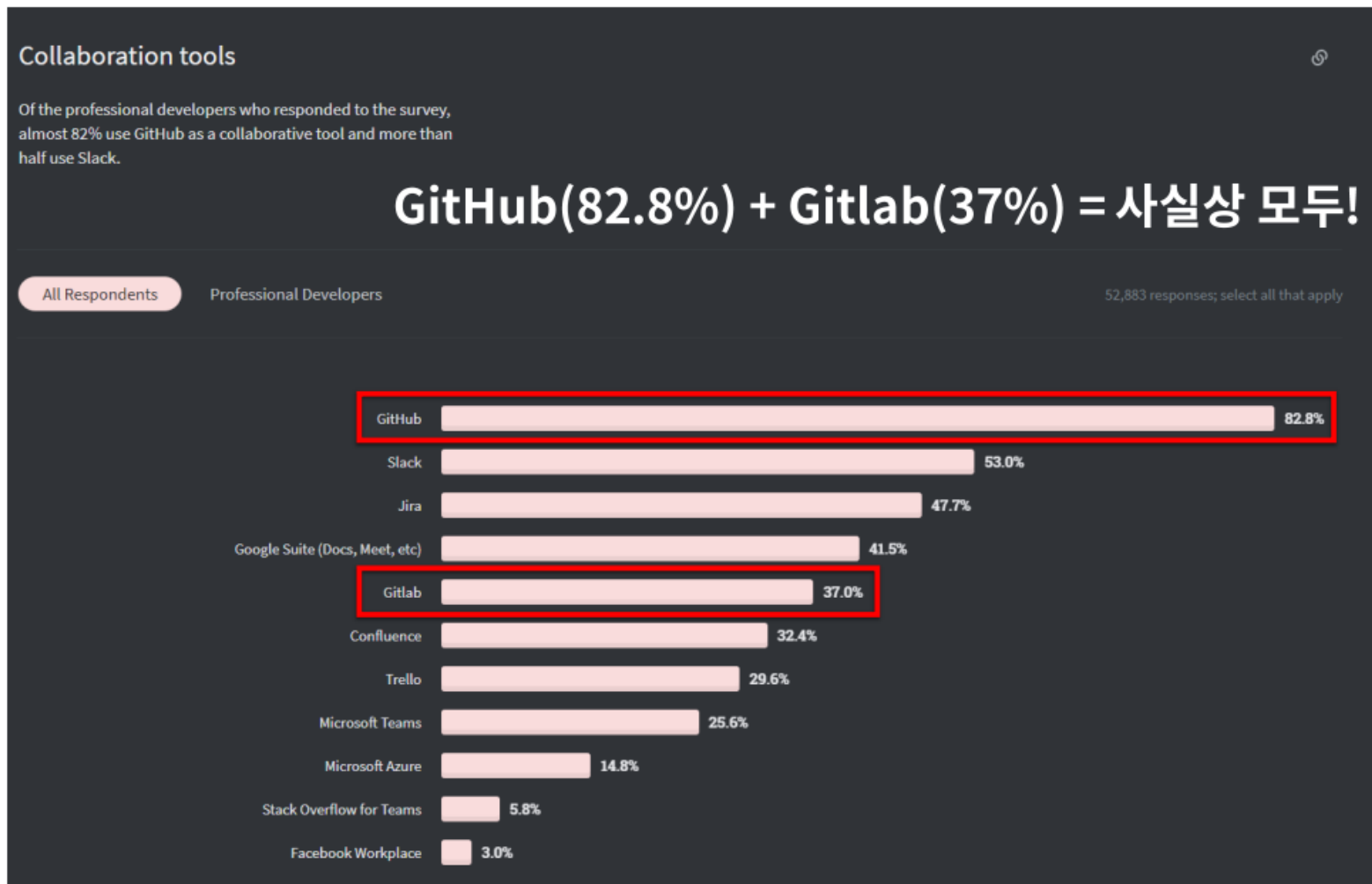
| Github을 사용하면 뭐가 좋을까?



Github을 사용하면 뭐가 좋을까?



## Github을 사용하면 뭐가 좋을까?



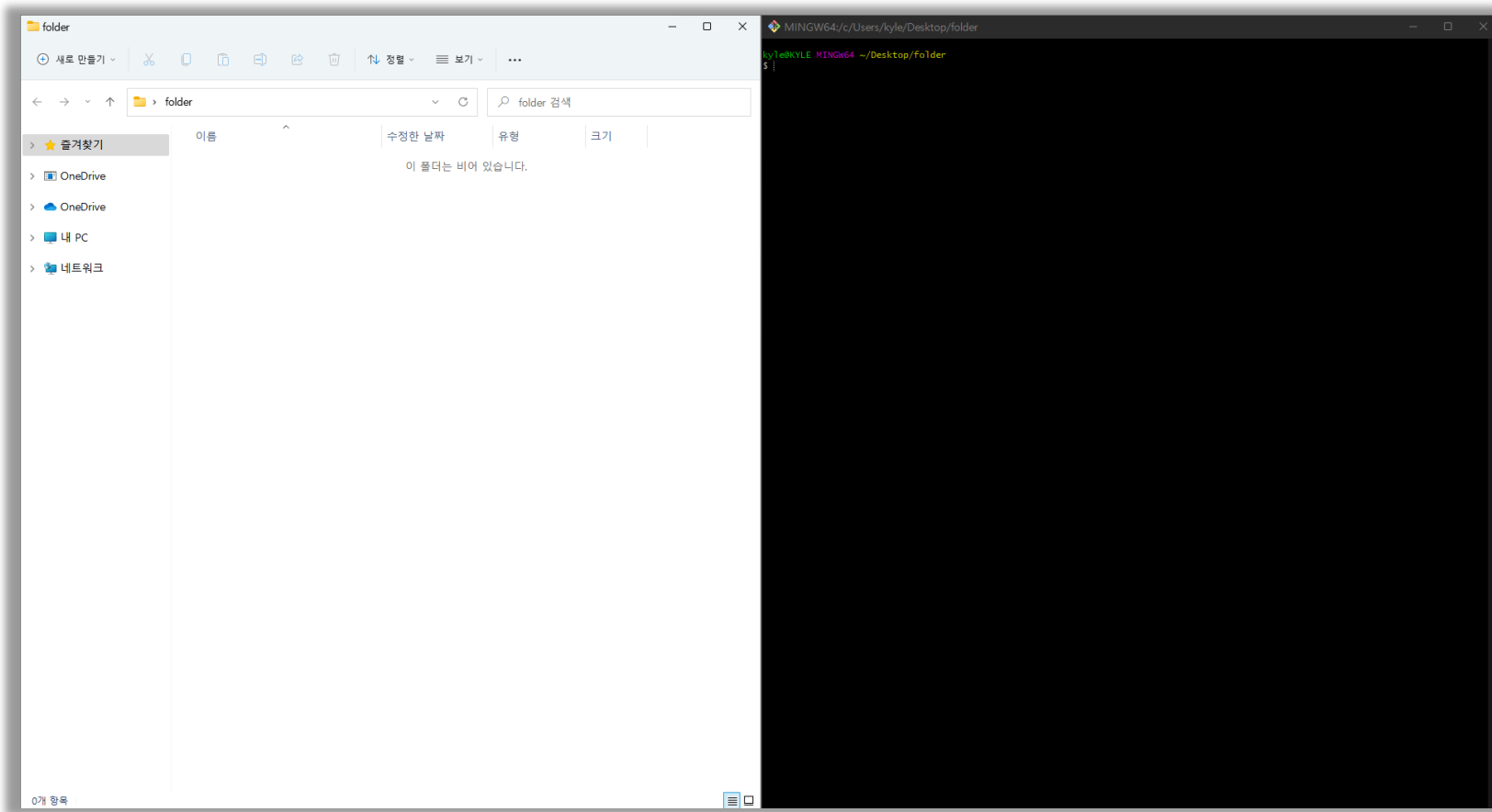
# 이어서...

삼성 청년 SW 아카데미

# CLI & Markdown

**CLI**

# GUI와 CLI



GUI

CLI



## | GUI와 CLI

- GUI (Graphic User Interface)
  - **그래픽**을 통해 사용자와 컴퓨터가 상호 작용하는 방식
- CLI (Command Line Interface)
  - **명령어**를 통해 사용자와 컴퓨터가 상호 작용하는 방식

## Why CLI

- GUI는 CLI에 비해 사용하기 쉽지만 단계가 많고 컴퓨터의 성능을 더 많이 소모
- 수 많은 서버 / 개발 시스템이 CLI를 통한 조작 환경을 제공

## 기본적인 명령어

- **touch**

- 파일을 생성하는 명령어

- **Mkdir**

- 새 폴더를 생성하는 명령어

- **ls**

- 현재 작업 중인 디렉토리의 폴더/파일 목록을 보여주는 명령어

- **cd**

- 현재 작업 중인 디렉토리를 변경하는 명령어

- **start, open**

- 폴더/파일을 여는 명령어 (Windows에서는 start를, Mac에서는 open을 사용)

- **rm**

- 파일을 삭제하는 명령어
- -r 옵션을 주면 폴더 삭제 가능



Git bash를 열고 하나씩 해봅시다!

## 절대경로 **VS** 상대경로

- 절대 경로
  - 루트 디렉토리부터 목적 지점까지 거치는 모든 경로를 전부 작성한 것
  - 윈도우 바탕 화면의 절대 경로 - C:/Users/ssafy/Desktop
- 상대 경로
  - 현재 작업하고 있는 디렉토리를 기준으로 계산된 상대적 위치를 작성한 것
  - 현재 작업하고 있는 디렉토리가 C:/Users일 때  
윈도우 바탕 화면으로의 상대 경로는 ssafy/Desktop
  - ./ : 현재 작업하고 있는 폴더                      ../ : 현재 작업하고 있는 폴더의 부모  
폴더

# 이어서...

삼성 청년 SW 아카데미

# Markdown

## Markdown

### 마크다운(markdown)

- 텍스트 기반의 가벼운 **마크업(markup)** 언어
- 문서의 구조와 내용을 같이 쉽고 빠르게 적고  
자 탄생

**마크업(markup)**

태그(tag)를 이용하여 문서의 구조를 나타내는 것

## Markdown

### # 개발자로 성장하기

- 대체 어디서부터 시작해서 어디까지 해야할까?
- Python과 Java를 배우면 개발자가 되는걸까?

제일 중요한건 **\*\*꾸준한 학습\*\***을 할 수 있는 사람인지를 보여줘야한다!



### 개발자로 성장하기

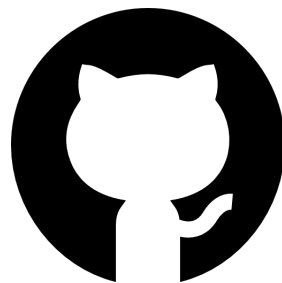
텍스트 에디터, 웹 환경 등

- 대체 어디서부터 시작해서 어디까지 해야할까?
- Python과 Java를 배우면 개발자가 되는걸까?

제일 중요한건 꾸준한 학습을 할 수 있는 사람인지를 보여줘야한다!



## Markdown



## Github 문서의 시작과 끝!

- README.md 파일을 통해 오픈 소스의 공식 문서 작성
- 개인 프로젝트의 소개 문서 작성
- 매일 학습한 내용 정리
- 마크다운을 이용한 블로그 운영

= 개발문서의 시작과  
끝

## Markdown

### 개발 문서의 시작과 끝!

- 대부분의 웹 에디터에서 지원 (각종 블로그 사이트 등)
- Jupyter Notebook, Notion, 다양한 메모장 프로그램



## Markdown



## Typora

- 실시간 마크다운 변환 (미리보기) 제공
- 이미지 또는 표 삽입시 매우 편한 UI 제공
- VS Code 등의 프로그램도 마크다운을 지원하지만  
전용 프로그램을 사용하면 더 편하게 사용가능

## Markdown

#

## 헤딩(Heading)

문서의 제목이나 소제목으로 사용해요

- #의 개수에 따라 제목의 수준을 구별 (h1 ~ h6)
- 문서 구조의 기본
- 글자 크기를 키우기 위해서 사용하면 안돼요

## Markdown

1. 2. 3.      \* \_

### 리스트(List)

순서가 있는 리스트와 순서가 없는 리스트

- 목록을 표시하기 위해 사용
- 많이 사용하는 태그 중 하나

## Markdown

`code block`    `inline code`  
block

코드 블록 (Code Block)

일반 텍스트와 다르게 코드를 이쁘게 출력  
해요

- 개발자가 마크다운을 사랑하는 이유 중 하나
- 사용하는 프로그램에 따라 특정 언어를 명시하면 구문 강조(Syntax Highlighting) 지원

## Markdown

```
[string](url)
```

링크 (link)

`string`은 보여지는 부분, `url`은 연결할 곳을 나타내요

- 다른 페이지로 이동하는 링크를 삽입
- 필요하다면 파일의 경로를 넣어 다운로드 가능한 링크로 만들 수 있어요

## Markdown

```
![string](img_url)
```

이미지 (image)

링크와 비슷하지만 이미지를 삽입합니다

- 이미지의 너비(width)와 높이(height)는 조절할 수 없어요
- 조절이 필요하다면 HTML을 사용해야 합니다.



## Markdown


**Bold** *italic* ~~strikeout~~

### 텍스트 강조 (Text Emphasis)

순서대로 굵게, 기울임, 취소선을 이용해 텍스트를 강조합니다.

- \*를 \_(underbar)로 대체 할 수 있어요.
- 취소선은 프로그램에 따라 지원하지 않을 수 있습니다.

## Markdown



### 수평선 (horizontal line)

가로로 긴 수평선을 작성합니다  
대개 단락을 구분할 때 사용해요

- - (hyphen)을 \*나 \_(underbar)로 대체할 수 있어요
- - 3개 이상만 적으면 똑같이 동작합니다

## Markdown 실습

### 마크다운(markdown)

- Typora를 이용해서 지금까지 배운 마크다운에 대한 정리 문서를 만들어 보세요. 형식은 자유입니다.
- 자신이 작성한 문서를 다른 사람과 비교해 보세요.

참고 자료

<https://www.markdownguide.org/cheat-sheet/>

# 이어서...

삼성 청년 SW 아카데미

# Git 기본기

# Git 기본기

## Git 기본기

- README.md
  - 프로젝트에 대한 설명 문서
  - Github 프로젝트에서 가장 먼저 보는 문서
  - 일반적으로 소프트웨어와 함께 배포
  - 작성 형식은 따로 없으나, 일반적으로 마크다운을 이용해 작성

## Repository



### Repository

특정 디렉토리를 버전 관리하는 **저장소**

- **git init** 명령어로 로컬 저장소를 생성
- **.git** 디렉토리에 **버전 관리에 필요한 모든 것이 들어 있음**



## Git 기본기

- README.md 생성하기
  - 새 폴더를 만들고 README.md 파일을 생성해 주세요.
  - 이 파일을 버전 관리하며 Git을 사용해 봅시다.
    - 특정 버전으로 남긴다 = “커밋(Commit)한다”

3가지 영역을 바탕으로 동작!

## | Git 기본기



Working Directory



Staging Area



Repository

커밋(commit)은 이 3가지 영역을 바탕으로 동작

## | Git 기본기



Working Directory

내가 작업하고 있는 **실제 디렉토리**



Staging Area

**커밋(commit)**으로 남기고 싶은,  
**특정 버전**으로 관리하고 싶은 파일이 있는 곳



Repository

**커밋(commit)**들이 저장되는 곳

## Git 기본기



Working Directory  
내가 작업하고 있는 **실제 디렉토리**



Staging Area  
**커밋(commit)**으로 남기고 싶은,  
**특정 버전으로** 관리하고 싶은 파일이 있는 곳



Repository  
**커밋(commit)**들이 저장되는 곳



untracked

git add

## Git 기본기



Working Directory

내가 작업하고 있는 **실제 디렉토리**



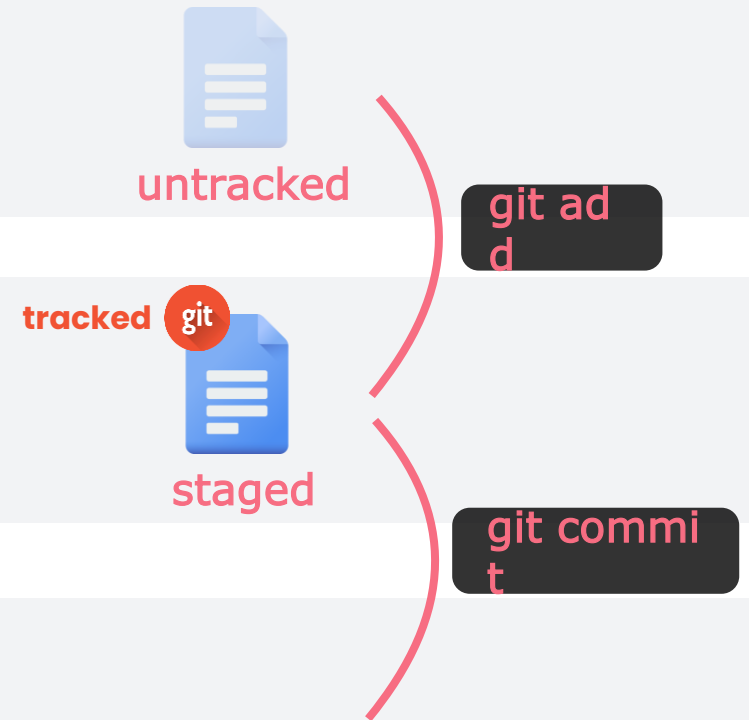
Staging Area

**커밋(commit)**으로 남기고 싶은,  
**특정 버전으로** 관리하고 싶은 파일이 있는 곳



Repository

**커밋(commit)**들이 저장되는 곳



## Git 기본기



Working Directory

내가 작업하고 있는 **실제 디렉토리**



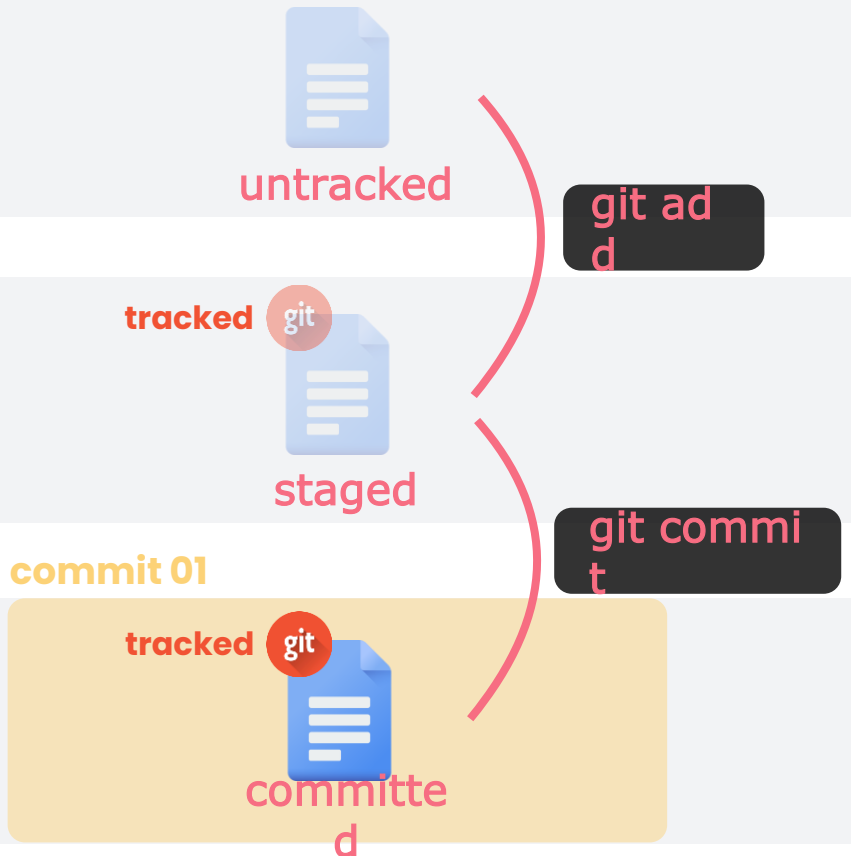
Staging Area

**커밋(commit)**으로 남기고 싶은,  
특정 **버전으로** 관리하고 싶은 파일이 있는 곳



Repository

**커밋(commit)**들이 저장되는 곳



## Git 기본기



Working Directory

내가 작업하고 있는 **실제 디렉토리**



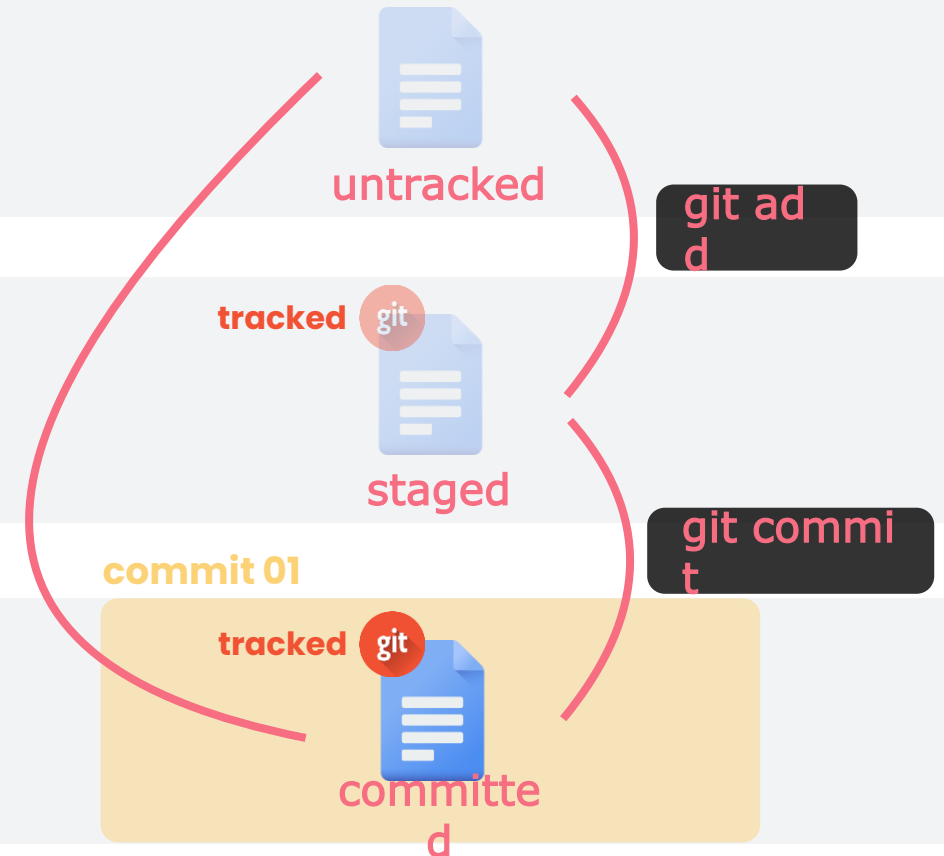
Staging Area

**커밋(commit)**으로 남기고 싶은,  
**특정 버전으로** 관리하고 싶은 파일이 있는 곳



Repository

**커밋(commit)**들이 저장되는 곳



## Git 기본기



### Working Directory

내가 작업하고 있는 **실제 디렉토리**



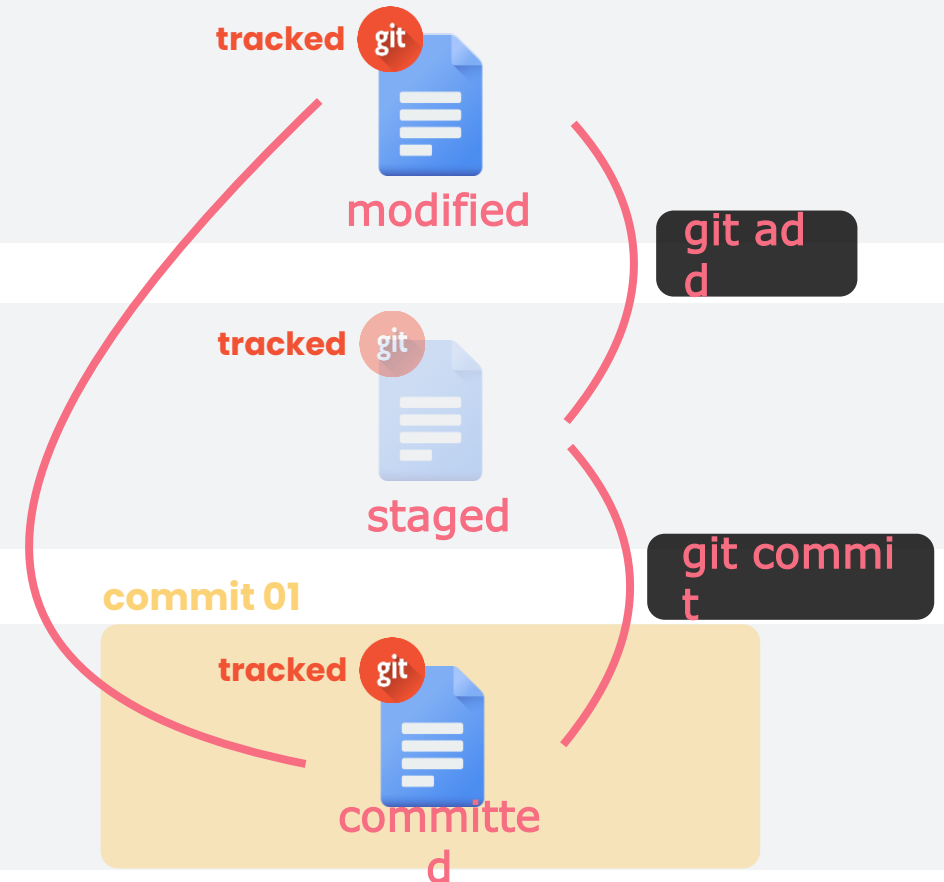
### Staging Area

**커밋(commit)**으로 남기고 싶은,  
**특정 버전으로** 관리하고 싶은 파일이 있는 곳



### Repository

**커밋(commit)**들이 저장되는 곳





## Git 기본기



### Working Directory

내가 작업하고 있는 **실제 디렉토리**



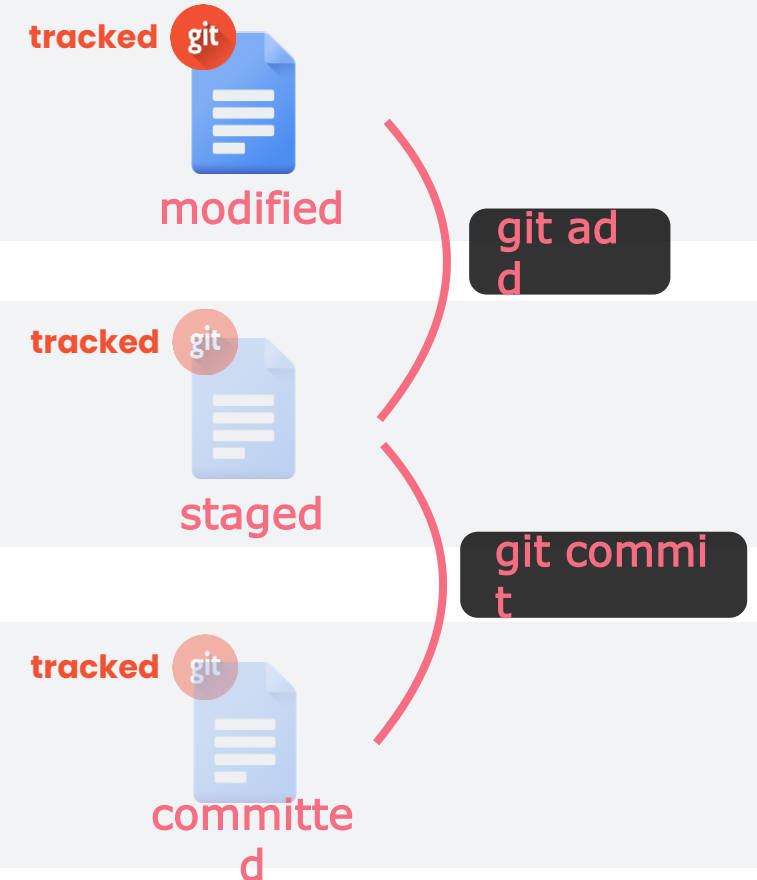
### Staging Area

**커밋(commit)**으로 남기고 싶은,  
**특정 버전으로** 관리하고 싶은 파일이 있는 곳



### Repository

**커밋(commit)**들이 저장되는 곳



## Git 기본기



### Working Directory

내가 작업하고 있는 실제 디렉토리



### Staging Area

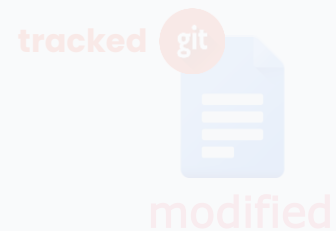
커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



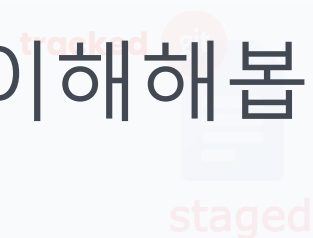
### Repository

커밋(commit)들이 저장되는 곳

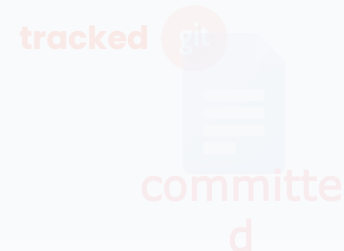
직접 **commit**을 남기면서 이해해봅시다



git add



git commit

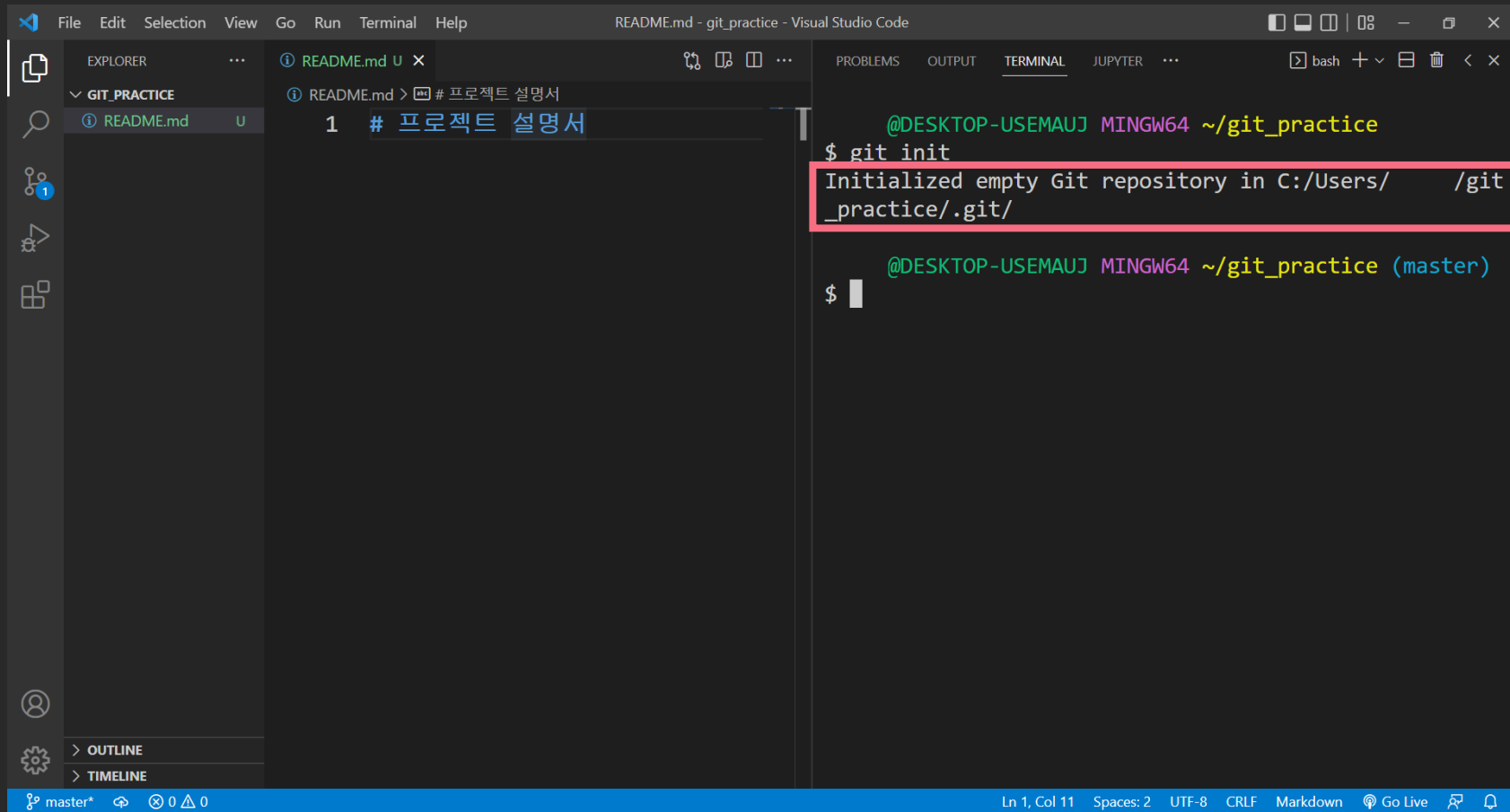


## | Git 기본기

# git status

현재 git으로 관리되고 있는 파일들의 상태를 알 수 있어요

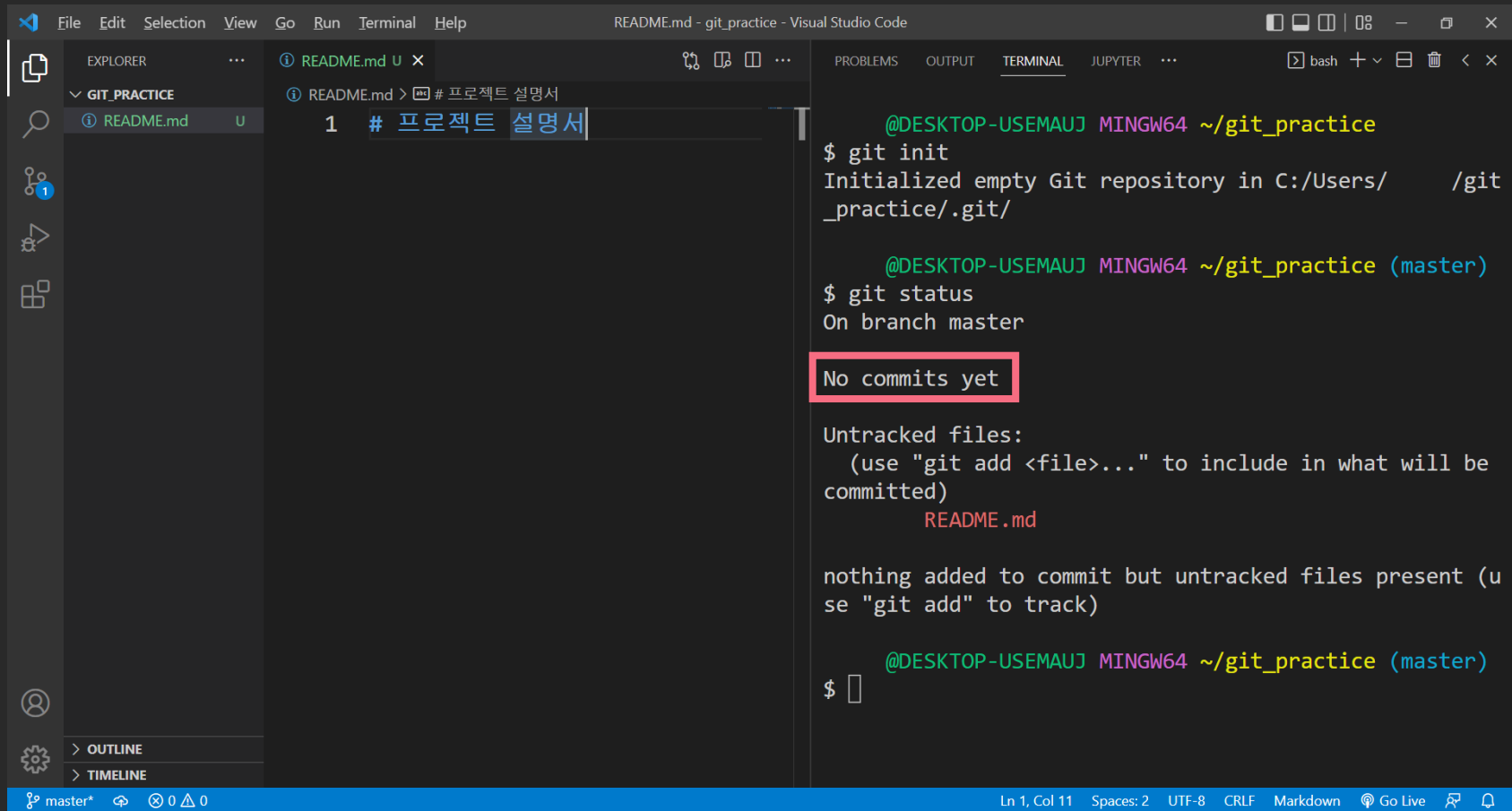
# Git 기본기



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the command `$ git init` and its output: `Initialized empty Git repository in C:/Users/ /git_practice/.git/`. The output line is highlighted with a red box. The terminal prompt is `@DESKTOP-USEMAUJ MINGW64 ~/git_practice`. The status bar at the bottom indicates the current branch is `master*`.

```
@DESKTOP-USEMAUJ MINGW64 ~/git_practice
$ git init
Initialized empty Git repository in C:/Users/ /git_practice/.git/
@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

## Git 기본기



```
File Edit Selection View Go Run Terminal Help
README.md - git_practice - Visual Studio Code

EXPLORER
  GIT_PRACTICE
    README.md U

1 # 프로젝트 설명서

PROBLEMS OUTPUT TERMINAL JUPYTER ...
bash + - - x

@DESKTOP-USEMAUJ MINGW64 ~/git_practice
$ git init
Initialized empty Git repository in C:/Users/ /git_practice/.git/

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git status
On branch master

No commits yet

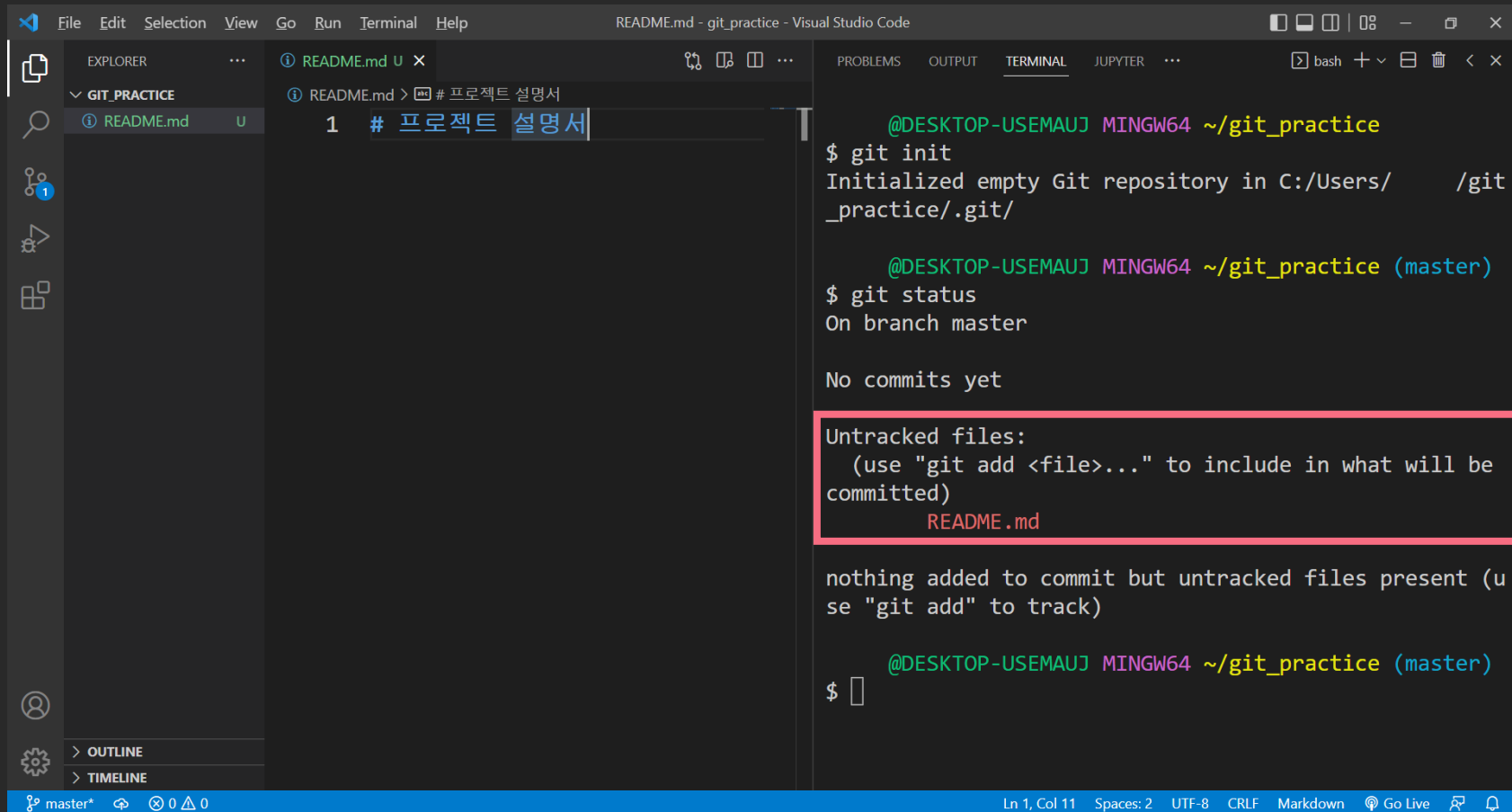
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  README.md

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

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

master\* 0 0 0 Ln 1, Col 11 Spaces: 2 UTF-8 CRLF Markdown Go Live

## Git 기본기



```
File Edit Selection View Go Run Terminal Help
README.md - git_practice - Visual Studio Code

EXPLORER
  GIT_PRACTICE
    README.md U

1 # 프로젝트 설명서

PROBLEMS OUTPUT TERMINAL JUPYTER ...
bash + - - x

@DESKTOP-USEMAUJ MINGW64 ~/git_practice
$ git init
Initialized empty Git repository in C:/Users/ /git_practice/.git/

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git status
On branch master

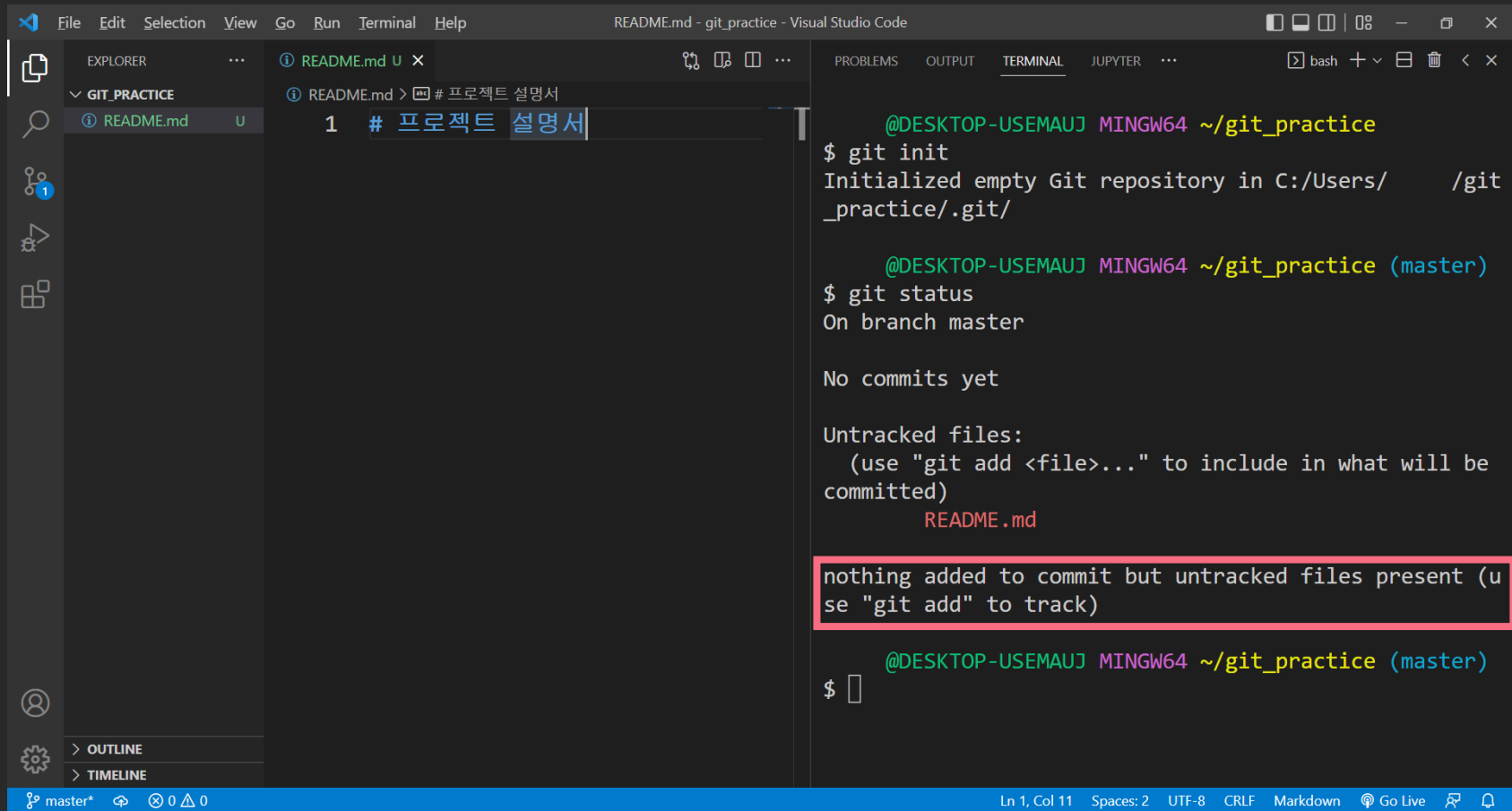
No commits yet

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)

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

## Git 기본기



```
File Edit Selection View Go Run Terminal Help
README.md - git_practice - Visual Studio Code

EXPLORER
  GIT_PRACTICE
    README.md U

1 # 프로젝트 설명서

TERMINAL
  bash
  @DESKTOP-USEMAUJ MINGW64 ~/git_practice
  $ git init
  Initialized empty Git repository in C:/Users/ /git_practice/.git/

  @DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
  $ git status
  On branch master

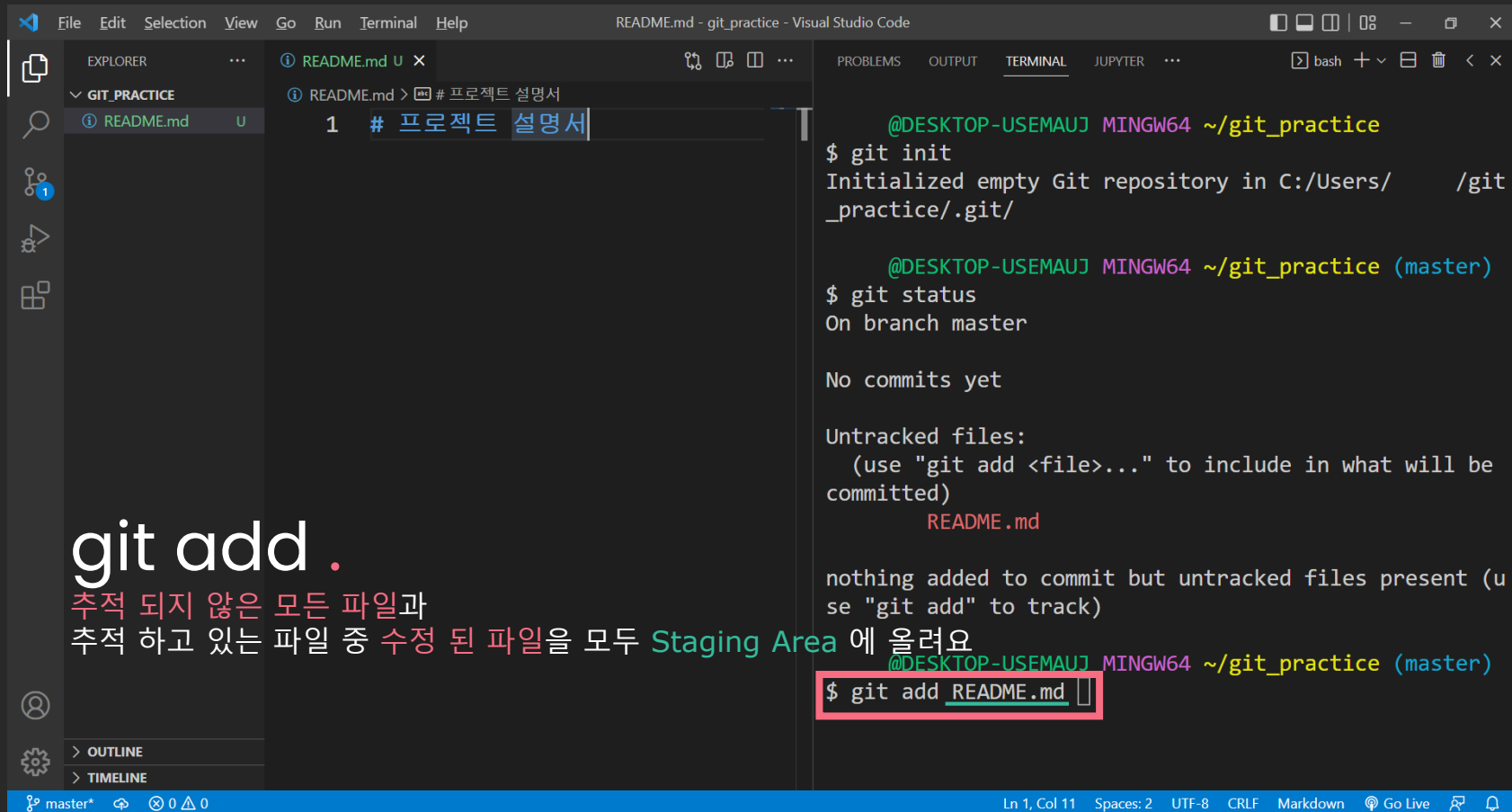
  No commits yet

  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)

  @DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
  $
```

## Git 기본기



The screenshot shows the Visual Studio Code interface with a Git repository named 'git\_practice'. The Explorer panel on the left shows a file named 'README.md' with a status icon. The Editor panel shows the content of 'README.md' with the text '# 프로젝트 설명서'. The Terminal panel on the right shows the output of the following commands:

```
@DESKTOP-USEMAUJ MINGW64 ~/git_practice
$ git init
Initialized empty Git repository in C:/Users/ /git_practice/.git/

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (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)

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git add README.md
```

The status bar at the bottom indicates the current branch is 'master' and there are 0 changes.

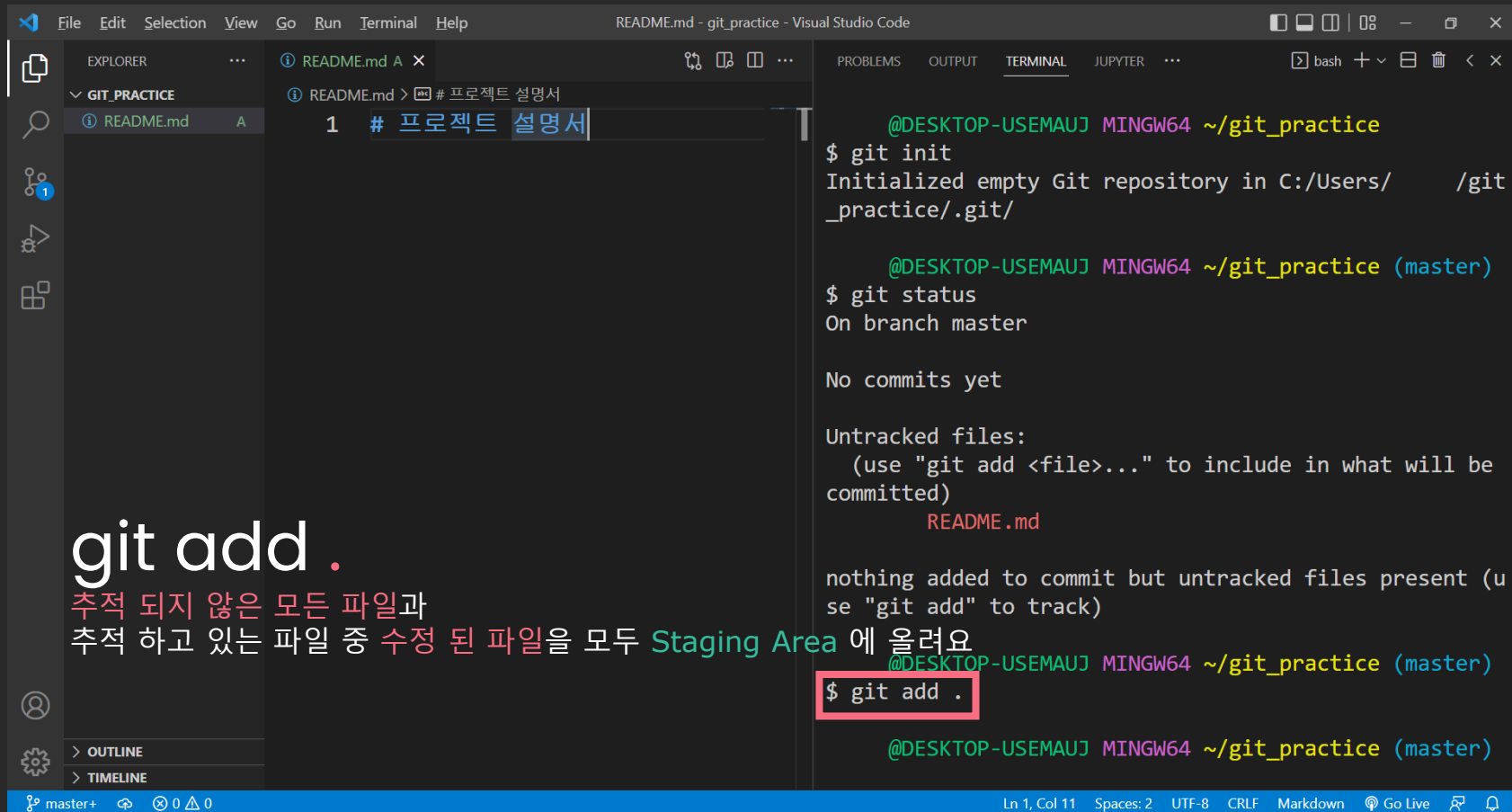
# git add .

추적 되지 않은 모든 파일과

추적 하고 있는 파일 중 수정 된 파일을 모두 Staging Area 에 올려요



## Git 기본기



```
File Edit Selection View Go Run Terminal Help
README.md - git_practice - Visual Studio Code

EXPLORER
  GIT_PRACTICE
    README.md A

1 # 프로젝트 설명서

git add .
추적 되지 않은 모든 파일과
추적 하고 있는 파일 중 수정 된 파일을 모두 Staging Area 에 올려요

@DESKTOP-USEMAUJ MINGW64 ~/git_practice
$ git init
Initialized empty Git repository in C:/Users/ /git_practice/.git/

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git status
On branch master

No commits yet

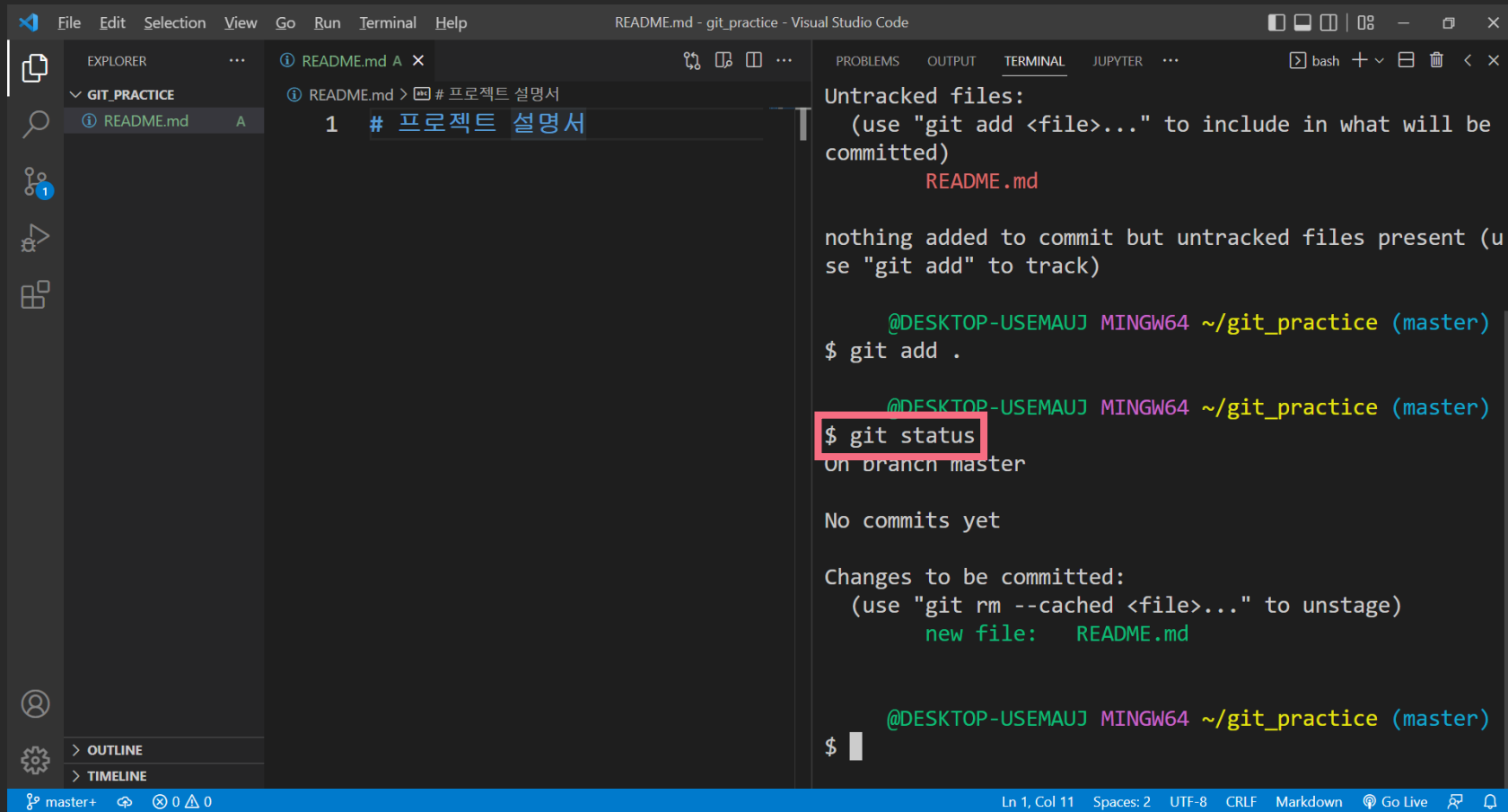
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  README.md

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

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git add .

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
```

## Git 기본기



The screenshot shows the Visual Studio Code interface with a terminal window open. The Explorer sidebar on the left shows a project named 'GIT\_PRACTICE' with a file 'README.md' marked as 'A' (added). The editor window shows the content of 'README.md' with the text '# 프로젝트 설명서' (Project Description) on line 1. The terminal window displays the following output:

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

nothing added to commit but untracked files present (u
se "git add" to track)

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git add .

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

The status bar at the bottom indicates the current branch is 'master+' and shows 0 changes.

## Git 기본기

```
File Edit Selection View Go Run Terminal Help
README.md - git_practice - Visual Studio Code

EXPLORER
  GIT_PRACTICE
    README.md A

1 # 프로젝트 설명서

PROBLEMS OUTPUT TERMINAL JUPYTER ...
bash + - & x

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)

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git add .

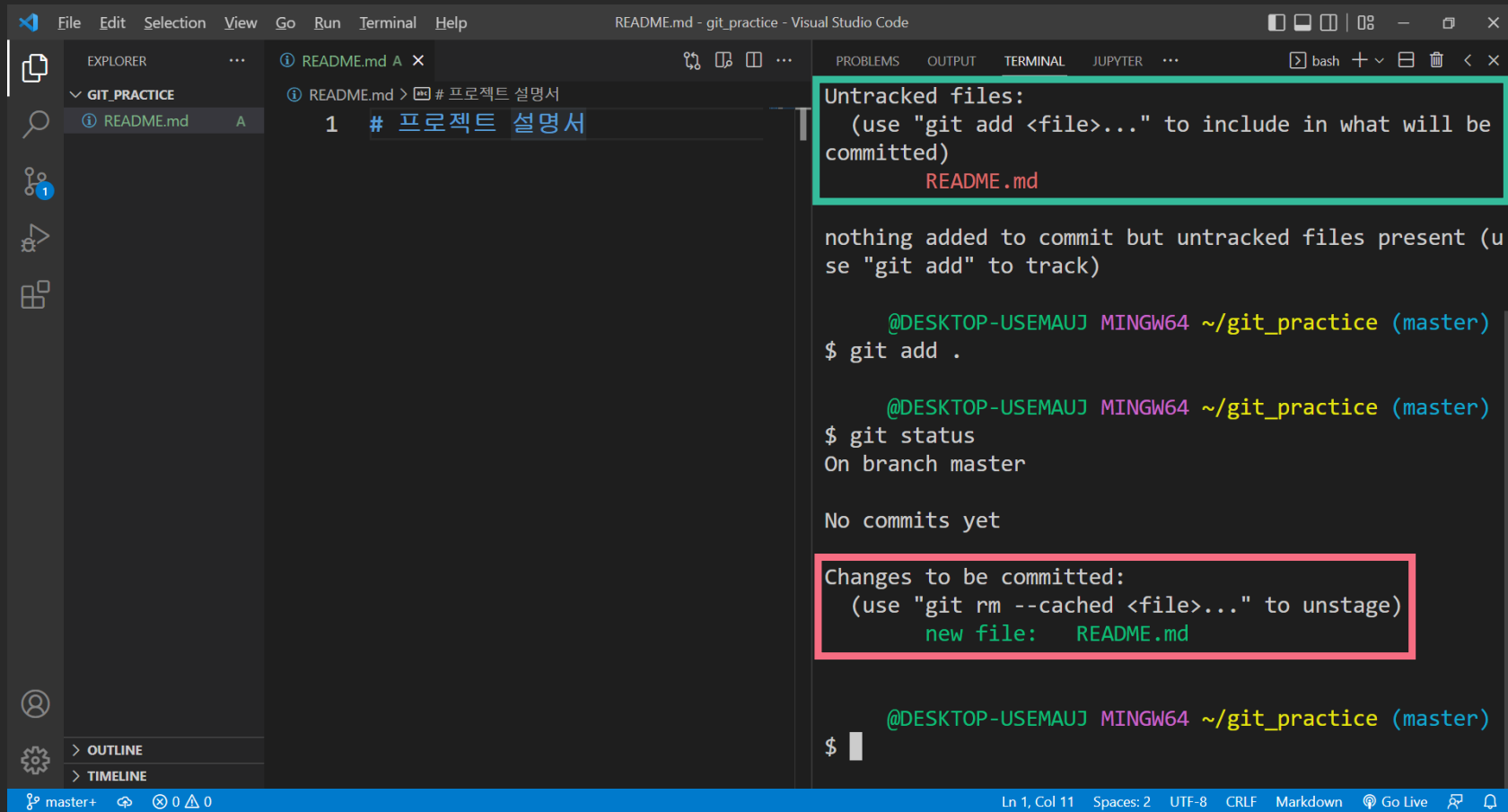
@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:   README.md

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

## Git 기본기



```
File Edit Selection View Go Run Terminal Help
README.md - git_practice - Visual Studio Code

EXPLORER
  GIT_PRACTICE
    README.md A

1 # 프로젝트 설명서

PROBLEMS OUTPUT TERMINAL JUPYTER ...
bash + - - x

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

nothing added to commit but untracked files present (u
se "git add" to track)

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git add .

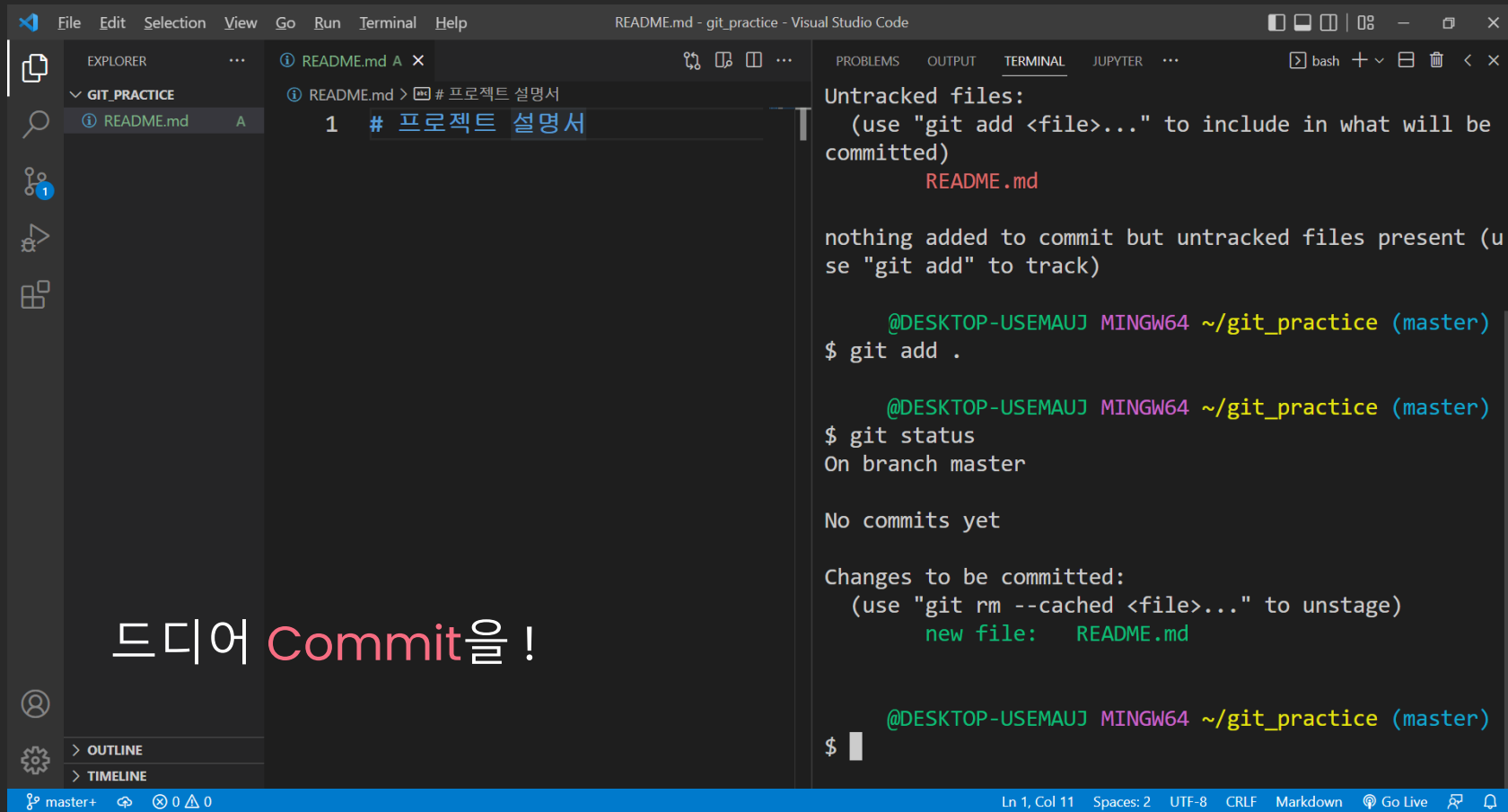
@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:   README.md

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

## Git 기본기



드디어 **Commit**을 !

```
File Edit Selection View Go Run Terminal Help
README.md - git_practice - Visual Studio Code

EXPLORER
  GIT_PRACTICE
    README.md A

1 # 프로젝트 설명서

PROBLEMS OUTPUT TERMINAL JUPYTER ...
bash + - & x

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)

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git add .

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git status
On branch master

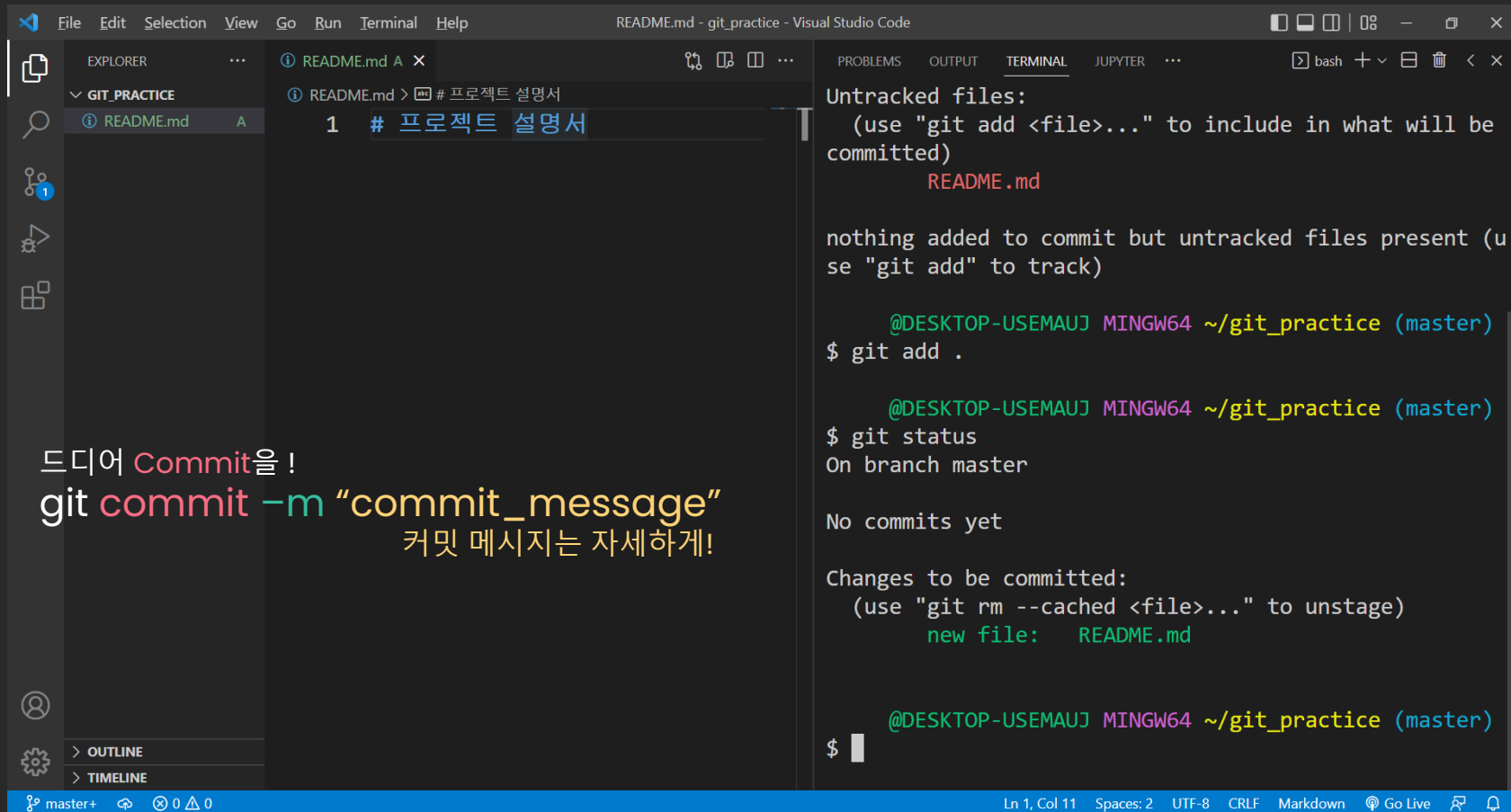
No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:   README.md

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

master+ 0 0 0 Ln 1, Col 11 Spaces: 2 UTF-8 CRLF Markdown Go Live

## Git 기본기



드디어 Commit을 !  
git commit -m "commit\_message"  
커밋 메시지는 자세하게!

```
File Edit Selection View Go Run Terminal Help
README.md - git_practice - Visual Studio Code

EXPLORER
  GIT_PRACTICE
    README.md A

1 # 프로젝트 설명서

PROBLEMS OUTPUT TERMINAL JUPYTER ...
bash + - - x

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

nothing added to commit but untracked files present (u
se "git add" to track)

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git add .

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:   README.md

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

master+ 0 0 0 Ln 1, Col 11 Spaces: 2 UTF-8 CRLF Markdown Go Live



## | Git 기본기

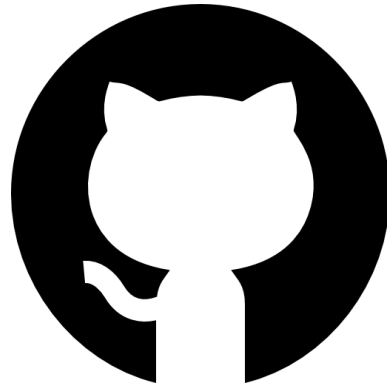


user.email?  
개발자는 Gmail이지!

Gmail 생성하기



## | Git 기본기



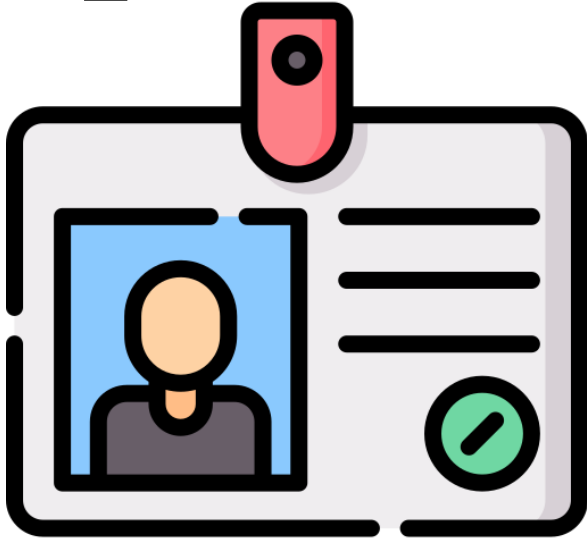
user.name?

**Github**의 username!

Github 회원가입

## Git 기본기

좋은 유저네임?



- 본명을 포함시키세요!
- 다른 곳에도 사용하세요!
- 대학, 현 직장 등을 포함시키지 마세요!
- 긴 이름보다 짧은 이름이 좋습니다!

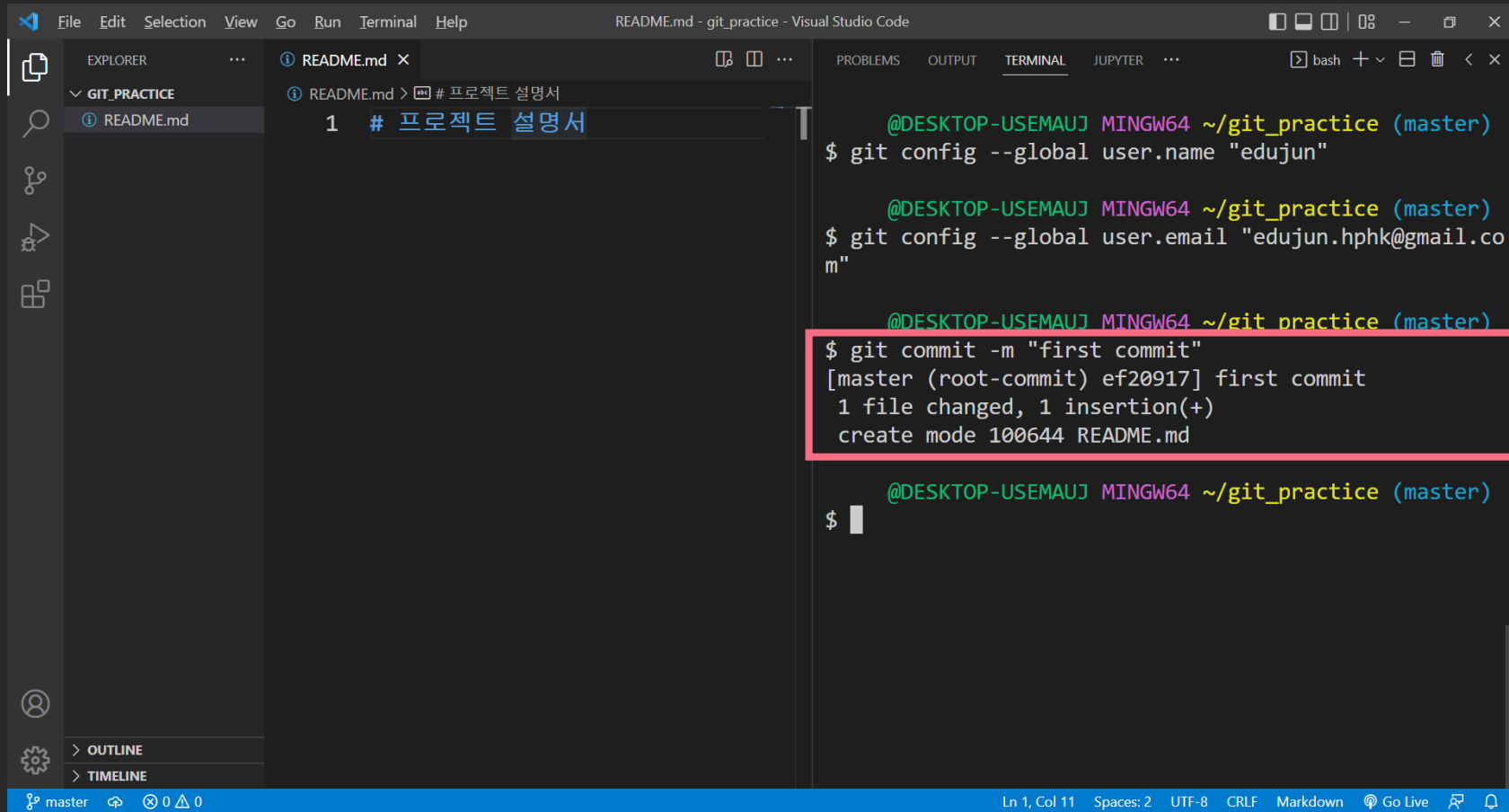
## | Git 기본기



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

```
git config --global user.email "user_email"
```

## Git 기본기



The screenshot shows the Visual Studio Code interface with a terminal window open. The Explorer panel on the left shows a project named 'GIT\_PRACTICE' with a file 'README.md'. The main editor shows the content of 'README.md', which is '# 프로젝트 설명서'. The terminal window shows the following commands and output:

```
@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git config --global user.name "edujun"

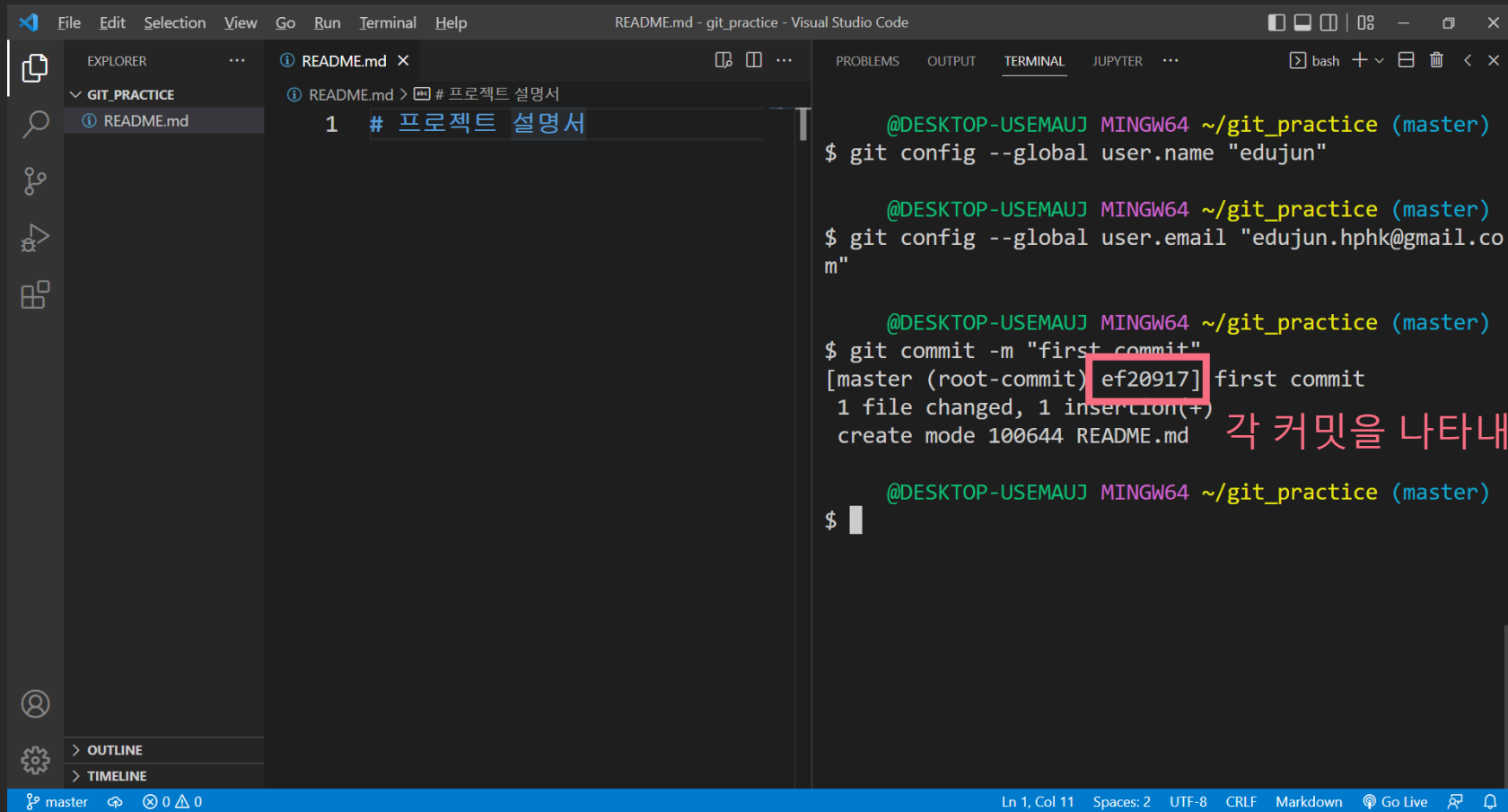
@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git config --global user.email "edujun.hphk@gmail.com"

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git commit -m "first commit"
[master (root-commit) ef20917] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$
```

The terminal output for the commit command is highlighted with a red box. The status bar at the bottom indicates the current branch is 'master' and the file encoding is UTF-8.

## Git 기본기



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the following commands and output:

```
@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git config --global user.name "edujun"

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git config --global user.email "edujun.hphk@gmail.com"

@DESKTOP-USEMAUJ MINGW64 ~/git_practice (master)
$ git commit -m "first commit"
[master (root-commit) ef20917] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

A red box highlights the commit hash `ef20917` in the terminal output. To the right of the terminal, the text "각 커밋을 나타내는 고유 아이디" (Unique ID representing each commit) is written in red.

The background shows the VS Code Explorer with a file named `README.md` and the editor with the content: `# 프로젝트 설명서`.

# Repository



## Repository

특정 디렉토리를 버전 관리하는 **저장소**

- **git init** 명령어로 로컬 저장소를 생성
- **.git** 디렉토리에 **버전 관리에 필요한 모든 것이 들어 있음**

## Git 기본기



Working Directory

내가 작업하고 있는 **실제 디렉토리**



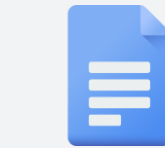
Staging Area

**커밋(commit)**으로 남기고 싶은,  
**특정 버전으로** 관리하고 싶은 파일이 있는 곳



Repository

**커밋(commit)**들이 저장되는 곳



untracked

git add

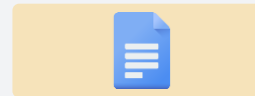
tracked



staged

git commit

commit 01

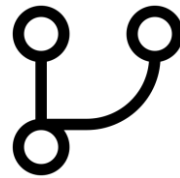


tracked



committed

## Git 기본기 실습



# Git commit 실습

- 바탕화면에 `edu_git_commit` 폴더를 만들고 `git` 저장소를 생성해주세요
- 해당 폴더 안에 `a.txt` 라는 텍스트 파일을 만들고,  
"`add a.txt`" 라는 커밋메세지로 커밋을 만들어주세요
- 이번에는 `b.txt` 라는 텍스트 파일을 만들고,  
"`add b.txt`" 라는 커밋메세지로 커밋을 만들어주세요
- `a.txt` 파일을 수정하고,  
"`update a.txt`" 라는 커밋메세지로 커밋을 만들어주세요



git status



## | Git 기본기

왜 굳이 중간에 **Staging Area**를 거쳐서 Commit을 할  
까?

## Git 기본기



### Working Directory

내가 작업하고 있는 **실제 디렉토리**



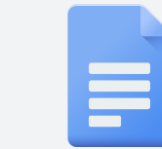
### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



### Repository

커밋(commit)들이 저장되는 곳



untracked

git add

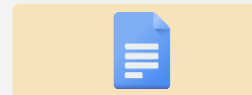
tracked



staged

git commit

commit 01



tracked



committed

## Git 기본기



### Working Directory

내가 작업하고 있는 **실제 디렉토리**

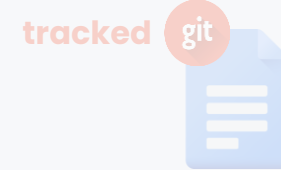


untracked



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



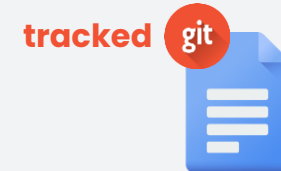
staged



### Repository

커밋(commit)들이 저장되는 곳

commit 01



committed

git commit

## Git 기본기



### Working Directory

내가 작업하고 있는 **실제 디렉토리**



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳

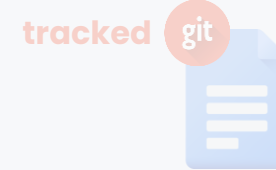


### Repository

커밋(commit)들이 저장되는 곳



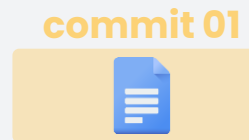
untracked



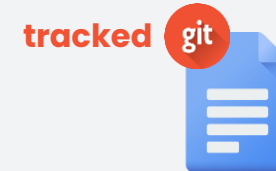
tracked

staged

git commit



commit 01



tracked

committed

## Git 기본기



### Working Directory

내가 작업하고 있는 실제 디렉토리

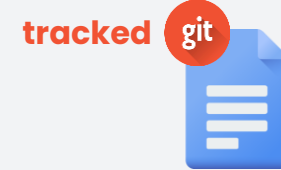


untracked



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳

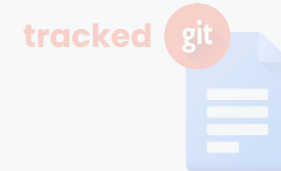
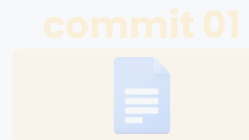


staged



### Repository

커밋(commit)들이 저장되는 곳



committed

## Git 기본기

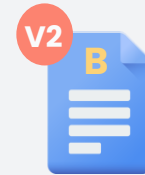


### Working Directory

내가 작업하고 있는 **실제 디렉토리**



개발 완료!



개발 완료!



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



### Repository

커밋(commit)들이 저장되는 곳

git add

## Git 기본기

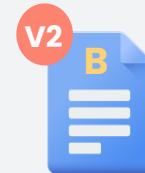


### Working Directory

내가 작업하고 있는 **실제 디렉토리**



개발 완료!



개발 완료!



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



### Repository

커밋(commit)들이 저장되는 곳

git commit

## Git 기본기

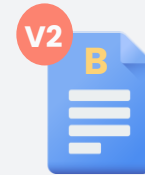


### Working Directory

내가 작업하고 있는 **실제 디렉토리**



개발 완료!

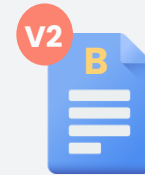


개발 완료!



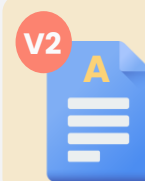
### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



### Repository

커밋(commit)들이 저장되는 곳



commit 01

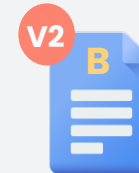


## Git 기본기



### Working Directory

내가 작업하고 있는 **실제 디렉토리**



개발 완료!

```
git add
```



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



### Repository

커밋(commit)들이 저장되는 곳

commit 01

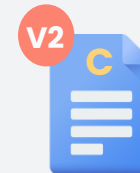
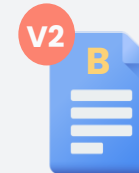


## Git 기본기



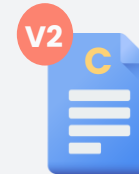
### Working Directory

내가 작업하고 있는 **실제 디렉토리**



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



### Repository

커밋(commit)들이 저장되는 곳

commit 01

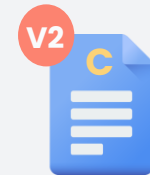
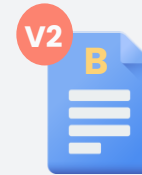
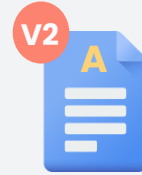


## Git 기본기



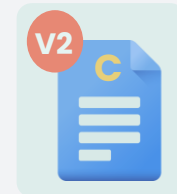
### Working Directory

내가 작업하고 있는 **실제 디렉토리**



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



git commit



### Repository

커밋(commit)들이 저장되는 곳

commit 01

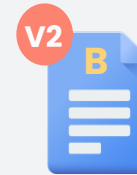


## Git 기본기



### Working Directory

내가 작업하고 있는 **실제 디렉토리**



### Staging Area

커밋(commit)으로 남기고 싶은,  
특정 버전으로 관리하고 싶은 파일이 있는 곳



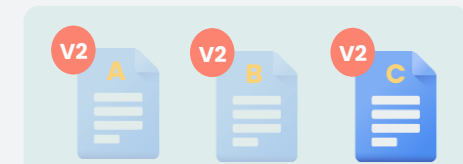
### Repository

커밋(commit)들이 저장되는 곳

commit 01



commit 02



## Git 기본기

- README.md 수정하기
  - README.md 파일을 수정하고 아래 명령어를 실행해 봅시다.
  - `git log` : git의 comit history 보기
  - `git diff` : 두 commit 간 차이 보기

## | Git basic

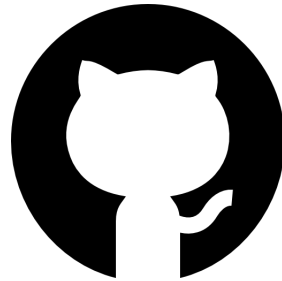
지금까지는 **Local Repository**에서만 작업했죠?

## | Git basic

이제, **Remote** Repository를 연결해봅시다.

# Github





## Github Repository 생성

## Github



**Github Repository**



**Local Repository**

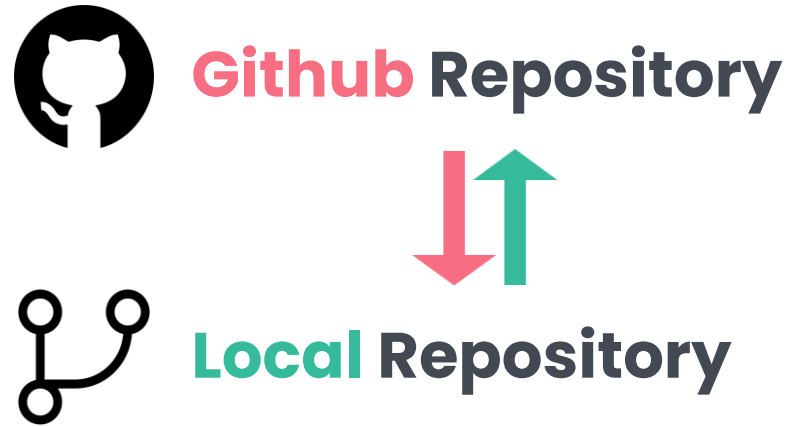


- git **remote** **add** origin {remote\_repo}
  - git **push** **-u** origin **master**
- <repo\_name> 별명** (points to {remote\_repo})
- <repo\_name> 별명** (points to origin)
- <local branch>** (points to master)

## | Github

- README.md 수정하고 **push** 하기
  - local repo에 **github repo**를 연결합니다.
  - README.md를 수정하고 **commit**을 생성합니다.
  - **github repo**에 **push** 합니다.

## | Github



- `git clone {remote_repo}`      remote repo를 local로 복사합니다
- `git push origin master`      local repo의 최신 커밋을  
remote repo로 push 합니다

## | TIL

# TIL Today I Learned

오늘 배운 내용을 정리해요

- 매일 내가 배운 것을 마크다운으로 정리해서 문서화하는 것
- 신입 개발자에게 요구되는 가장 큰 능력, 꾸준히 학습할 수 있나요?
- Github 관리의 가장 첫 걸음, 제일 중요한 장기 프로젝트



## TIL 프로젝트 실습

# TIL 프로젝트 실 습



- TIL이라는 이름의 Github Repository를 생성해 주세요.
- Local로 Clone 받아 README.md 파일을 생성해 주세요.
- Markdown 수업 내용을 정리하고 Commit을 생성한 뒤 push해 주세요.
- Git/Github 수업 내용을 정리하고 Commit을 생성한 뒤 push해 주세요.

### HINT

- 1) git **init**
- 2) git **commit** -m "message"
- 3) git **remote add** origin remote\_repo
- 4) git **push** -u origin master
- 5) git **add** .
- 6) git **push**

## | 알고리즘 Repo 만들기



## 알고리즘 Repo만들 기

- 바탕화면에 **algorithm** 폴더를 만들고, Repository로 설정해주세요
- **README.md** 파일을 생성해주세요
- 아래의 문제를 원하는 언어로 풀고 자유롭게 **README.md**의 내용을 작성해주세요.  
<https://www.acmicpc.net/problem/2557>  
<https://www.acmicpc.net/problem/1000>

- **algorithm** 이라는 이름의 **Github Repository**를 생성해주세요
- **commit**을 하고 **push** 해주세요

## Github Profile 만들기



## Github Profile 만들기

- 바탕화면에 **profile** 폴더를 만들고, **Repository**로 설정해주세요
- **README.md** 파일을 생성하고, 자유롭게 자기 소개를 작성해주세요
- 본인의 **username**으로 **Github Repository**를 생성해주세요
- **commit**을 하고 **push** 한 뒤, <https://github.com/username>으로 들어가보세요
- **README.md**를 수정하고 다시 **push** 해주세요



# 다음 방송에서 만나요!

삼성 청년 SW 아카데  
미