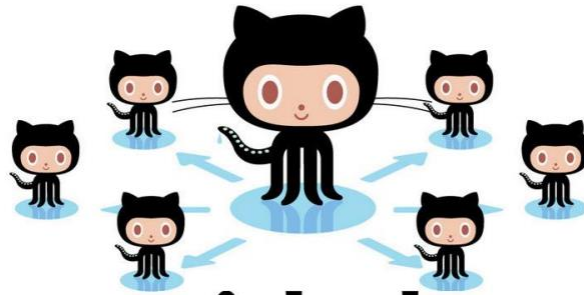


Git/GitHub

Git? GitHub?





github
SOCIAL CODING





DongUk Lee
jojoldu

Follow

Backend Application Developer

약 3.5k followers · 56 following

@infleam

Seoul

jojoldu@gmail.com

http://jojoldu.tistory.com/

@jojoldu

Achievements



Block or Report

Overview Repositories 182 Projects Packages Stars 262

jojoldu / README.md

Hi there 🙋

DongUk Lee's GitHub Stats

☆ Total Stars Earned: 10.3k
🕒 Total Commits (2021): 992
📄 Total PRs: 21
🔍 Total Issues: 235
👤 Contributed to: 6

A++

Pinned

blog-code (Public)

http://jojoldu.tistory.com/ 에서 제공하는 예제 code

Java 466 261

markdown-tistory (Public)

작성된 마크다운의 내용과 이미지를 본인 티스토리 에 업로드하는 프로젝트

JavaScript 198 40

junior-recruit-scheduler (Public)

주니어 개발자 채용 정보

JavaScript 7k 1.3k

translator (Public)

IntelliJ Translate Plugin

Java 176 24

springboot-webservice (Public)

스프링부트 서비스 구축하기 시리즈

Java 394 132

spring-boot-aws-mock (Public)

Queue, Redis, Etc (Aws Service Mock Library for Spring Boot)

Java 45 16

1,089 contributions in the last year



Learn how we count contributions

2021

2020

2019

2018

2017

Git/GitHub 특강

- Markdown을 활용한 문서 작성
- Git을 활용한 버전 관리
 - 버전 관리 기본
 - Git branch
- GitHub을 활용한 포트폴리오 관리 및 개발 프로젝트 시나리오
 - 개인 포트폴리오 관리
 - TIL (Today I Learned)
 - 개인 개발 프로젝트
 - 프로젝트(협업)
 - GitHub Flow를 활용한 개발 프로젝트 가이드라인
 - Shared repository model / Fork & Pull model

Markdown

마크다운 사용법 및 실습

마크다운 개요

- 2004년 존 그루버가 만든 텍스트 기반의 가벼운 마크업 언어
- 최초 마크다운에 비해 확장된 문법(표, 주석 등)이 있지만, 본 수업에서는 Github 에서 사용 가능한 문법(Github Flavored Markdown)을 기준으로 설명

Markdown is a **text-to-HTML conversion tool** for web writers.
Markdown allows you to write using an **easy-to-read, easy-to-write plain text format**, then convert it to structurally valid XHTML (or HTML).

Thus, “Markdown” is two things:

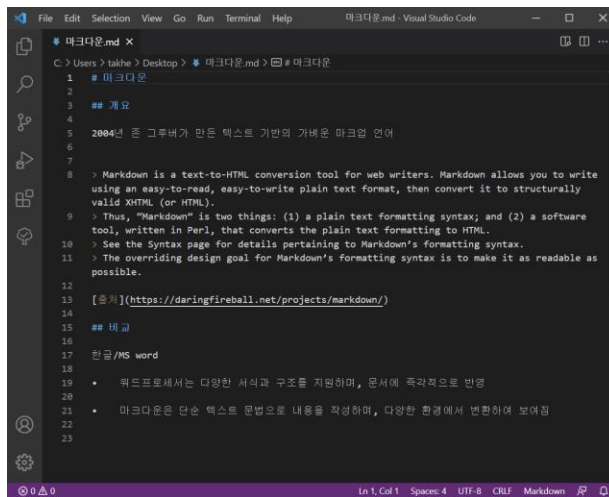
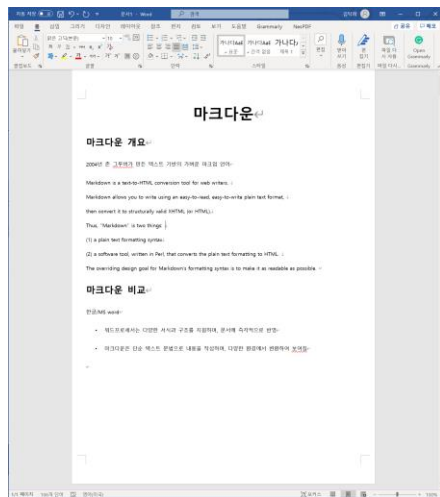
- (1) **a plain text formatting syntax**
- (2) a software tool, written in Perl, **that converts the plain text formatting to HTML.**

The overriding **design goal for Markdown’s formatting syntax is to make it as readable as possible.**

<https://daringfireball.net/projects/markdown/>

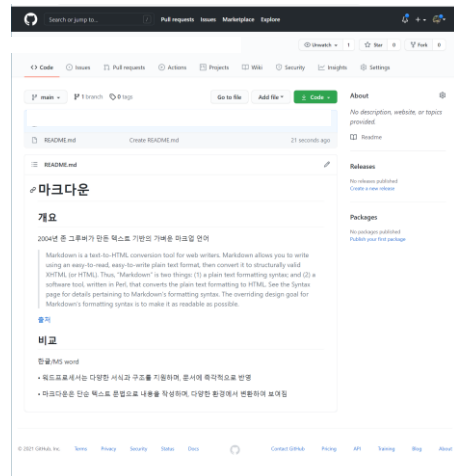
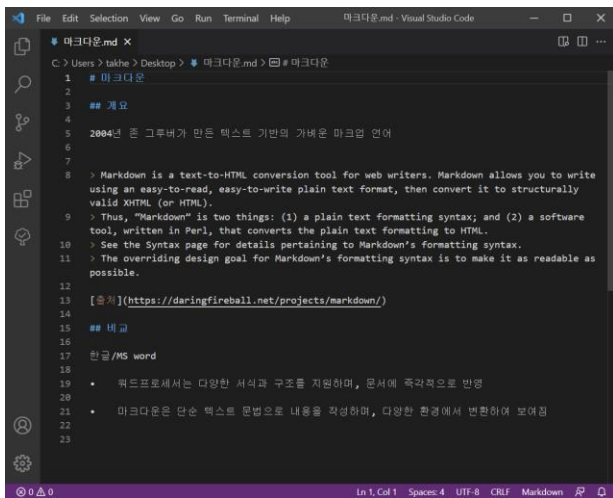
마크다운 특징

- 워드프로세서(한글/MS word)는 다양한 서식과 구조를 지원하며, 문서에 즉각적으로 반영
- 마크다운은 가능한 읽을 수 있도록 최소한의 문법으로 구조화 (make it as readable as possible)



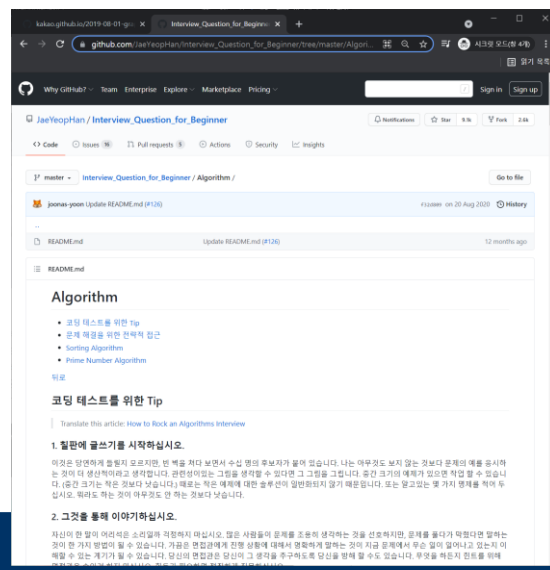
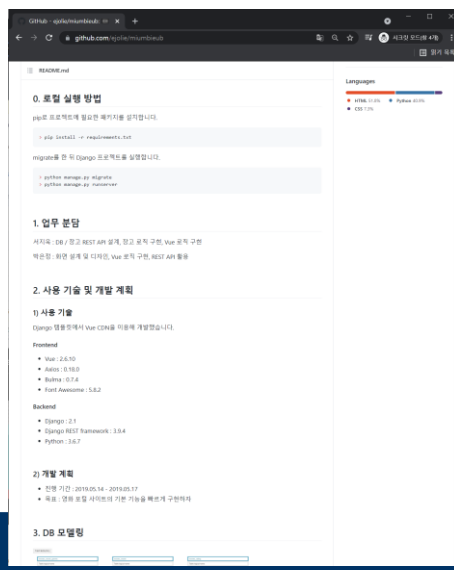
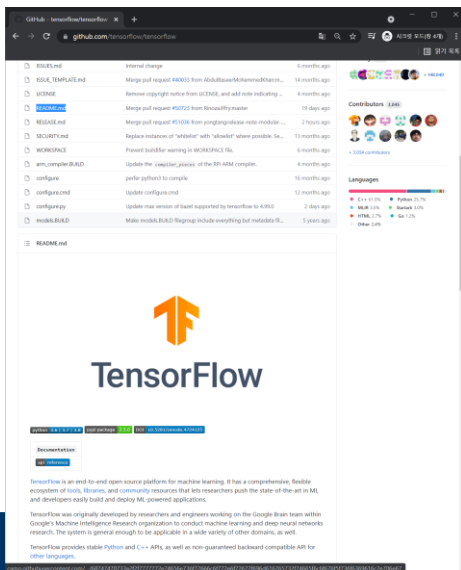
마크다운 특징

- 마크다운은 단순 텍스트 문법으로 내용을 작성하며, 다양한 환경에서 변환하여 보여짐
 - 다양한 text editor, 웹 환경에서 모두 지원



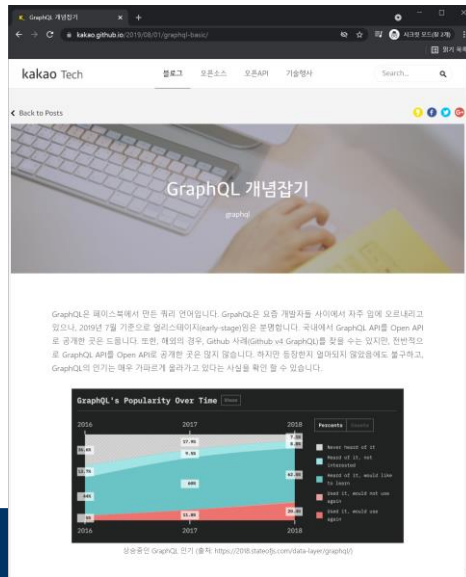
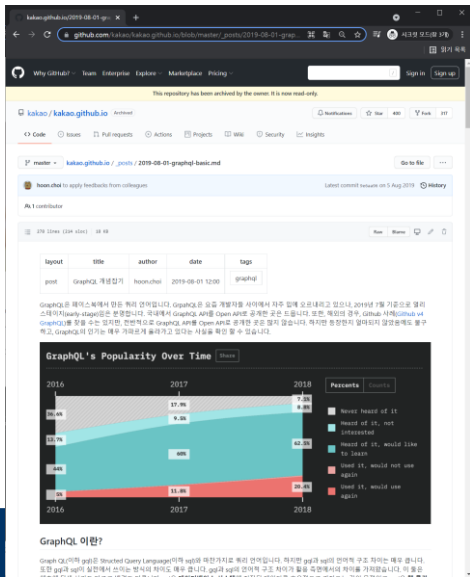
마크다운 활용 예 - README.md

- Github 등의 사이트에서는 파일명이 README.md인 것을 모두 보여줌
 - 오픈소스의 공식 문서를 작성하거나 개인 프로젝트의 프로젝트 소개서로 활용
 - 혹은 모든 페이지에 README.md를 넣어 문서를 바로 볼 수 있도록 활용



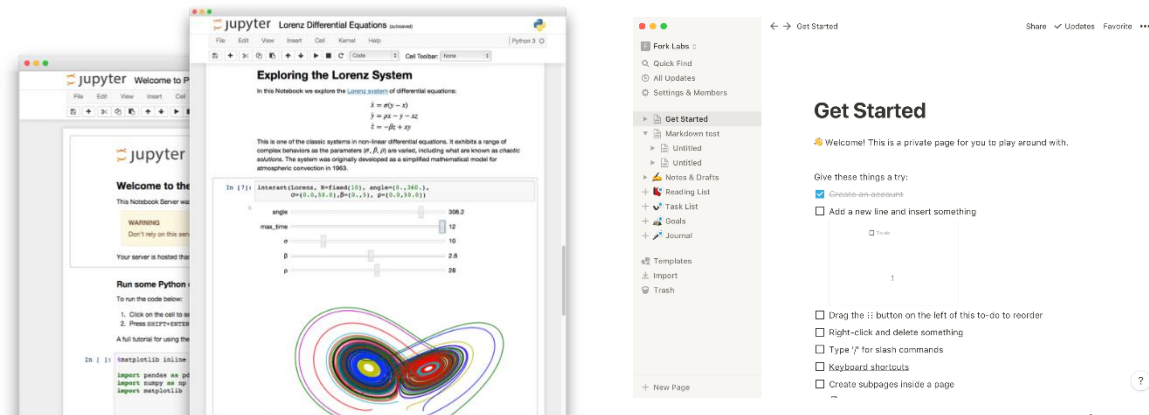
마크다운 활용 예 - 기술 블로그

- 다양한 기술블로그에서는 정적사이트생성기(Static site generator)
 - Jekyll, Gatsby, Hugo, Hexo 등을 통해 작성된 마크다운을 HTML, CSS, JS 파일 등으로 변환하고
 - Github pages 기능을 통해 호스팅 (외부 공개)



마크다운 활용 예 - 기타

- 다양한 개발 환경 뿐만 아니라 일반 SW에서도 많이 사용되고 있음
 - Jupyter notebook에는 별도의 마크다운 셀이 있어, 데이터 분석 등을 하는 과정에서 프로젝트 내용과 분석 결과를 정리함
 - Notion과 같은 메모/노트 필기 SW에서도 기본 문법으로 마크다운을 채택



<https://www.markdownguide.org/tools/>

마크다운 문법 - Heading

- Heading은 문서의 제목이나 소제목으로 사용
 - #의 개수에 따라 대응되는 수준(Heading level)이 있으며, h1 ~ h6까지 표현 가능
 - 문서의 구조를 위해 작성되며 글자 크기를 조절하기 위해 사용되어서는 안됨

Markdown	HTML	Rendered Output
# Heading level 1	<h1>Heading level 1</h1>	Heading level 1
## Heading level 2	<h2>Heading level 2</h2>	Heading level 2
### Heading level 3	<h3>Heading level 3</h3>	Heading level 3
#### Heading level 4	<h4>Heading level 4</h4>	Heading level 4
##### Heading level 5	<h5>Heading level 5</h5>	Heading level 5
##### Heading level 6	<h6>Heading level 6</h6>	Heading level 6

✔ Do this	✗ Don't do this
# Here's a Heading	#Here's a Heading

✔ Do this	✗ Don't do this
Try to put a blank line before... # Heading ...and after a heading.	Without blank lines, this might not look right. # Heading Don't do this!

<https://www.markdownguide.org/basic-syntax/#headings>

마크다운 문법 - List

- List는 순서가 있는 리스트(ol)와 순서가 없는 리스트(ul)로 구성

Markdown	HTML	Rendered Output
1. First item 2. Second item 3. Third item 4. Fourth item	<pre> First item Second item Third item Fourth item </pre>	1. First item 2. Second item 3. Third item 4. Fourth item

Markdown	HTML	Rendered Output
- First item - Second item - Third item - Fourth item	<pre> First item Second item Third item Fourth item </pre>	<ul style="list-style-type: none">• First item• Second item• Third item• Fourth item

<https://www.markdownguide.org/basic-syntax/#lists-1>

마크다운 문법 – Fenced Code block

- 코드 블록은 backtick 기호 3개를 활용하여 작성("` ` `")
- 코드 블록에 특정 언어를 명시하면 Syntax Highlighting 적용 가능
 - 일부 환경에서는 적용이 되지 않을 수 있음

```
```  
{
 "firstName": "John",
 "lastName": "Smith",
 "age": 25
}
```
```

The rendered output looks like this:

```
{  
  "firstName": "John",  
  "lastName": "Smith",  
  "age": 25  
}
```

```
```json  
{
 "firstName": "John",
 "lastName": "Smith",
 "age": 25
}
```
```

The rendered output looks like this:

```
{  
  "firstName": "John",  
  "lastName": "Smith",  
  "age": 25  
}
```

<https://www.markdownguide.org/extended-syntax/#fenced-code-blocks>

마크다운 문법 – Inline Code block

- 코드 블록은 backtick 기호 1개를 인라인에 활용하여 작성(``)

| Markdown | HTML | Rendered Output |
|--|---|--------------------------------------|
| At the command prompt, type
`nano`. | At the command prompt, type
<code>nano</code>. | At the command prompt,
type nano. |

마크다운 문법 – Link

- [문자열](url)을 통해 링크를 작성 가능
 - 특정 파일들 포함하여 연결 시킬 수도 있음

To create a link, enclose the link text in brackets (e.g., [Duck Duck Go]) and then follow it immediately with the URL in parentheses (e.g., (https://duckduckgo.com)).

```
My favorite search engine is [Duck Duck Go](https://duckduckgo.com).
```

The rendered output looks like this:

My favorite search engine is [Duck Duck Go](https://duckduckgo.com).

마크다운 문법 – 이미지

- **!**[문자열](url)을 통해 이미지를 사용 가능
 - 특정 파일들 포함하여 연결 시킬 수도 있음

```
![The San Juan Mountains are beautiful!](/assets/images/san-juan-mountains.jpg "San Juan Mountains")
```

The rendered output looks like this:



<https://www.markdownguide.org/basic-syntax/#links>

마크다운 문법 – Blockquotes (인용문)

- >를 통해 인용문을 작성

To create a blockquote, add a > in front of a paragraph.

```
> Dorothy followed her through many of the beautiful rooms in her castle.
```

The rendered output looks like this:

```
Dorothy followed her through many of the beautiful rooms in her castle.
```

마크다운 문법 – Table (표)

- 표는 아래의 문법을 참고
 - 일부 지원 안되는 환경도 있음

```
Syntax	Description
Header	Title
Paragraph	Text
```

The rendered output looks like this:

| Syntax | Description |
|-----------|-------------|
| Header | Title |
| Paragraph | Text |

<https://www.markdownguide.org/extended-syntax/>

마크다운 문법 - text 강조

- 굵게(bold), 기울임(Italic)을 통해 특정 글자들을 강조

| Markdown | HTML | Rendered Output |
|--------------------------------|---|--------------------------------|
| I just love bold text . | I just love bold text. | I just love bold text . |
| I just love bold text . | I just love bold text. | I just love bold text . |
| Love is bold | Loveisbold | Love is bold |

| Markdown | HTML | Rendered Output |
|--|---|--|
| Italicized text is the <i>cat's meow</i> . | Italicized text is the cat's meow. | Italicized text is the <i>cat's meow</i> . |
| Italicized text is the <i>cat's meow</i> . | Italicized text is the cat's meow. | Italicized text is the <i>cat's meow</i> . |
| A <i>cat</i> meow | Acatmeow | A <i>cat</i> meow |

<https://www.markdownguide.org/basic-syntax/#emphasis>

마크다운 문법 - 수평선

- 3개 이상의 asterisks (***), dashes (---), or underscores (___)

To create a horizontal rule, use three or more asterisks (***), dashes (---), or underscores (___) on a line by themselves.

```
***
```

```
---
```

```
_____
```

The rendered output of all three looks identical:

마크다운 관련 자료

- GitHub Flavored Markdown (<https://github.github.com/gfm/>)
- Mastering Markdown (<https://guides.github.com/features/mastering-markdown/>)
- Markdown Guide (<https://www.markdownguide.org/>)

개발자에게 문서 작성이란?

- 백엔드 개발자를 꿈꾸는 학생 개발자들에게 (<https://d2.naver.com/news/3435170>)
 - 레벨 2 개발자 : ‘자신이 경험한 사용법을 문서화해서 팀 내에 전파할 수 있음’
- Google Technical Writing (<https://developers.google.com/tech-writing>)
 - Every engineer is also a writer
- Technical writing conference (<https://engineering.linecorp.com/ko/blog/write-the-docs-prague-2018-recap/>)
 - Clova 기술 문서 작성 및 관리 업무

Markdown 실습

TYPORA를 활용한 문서 작성

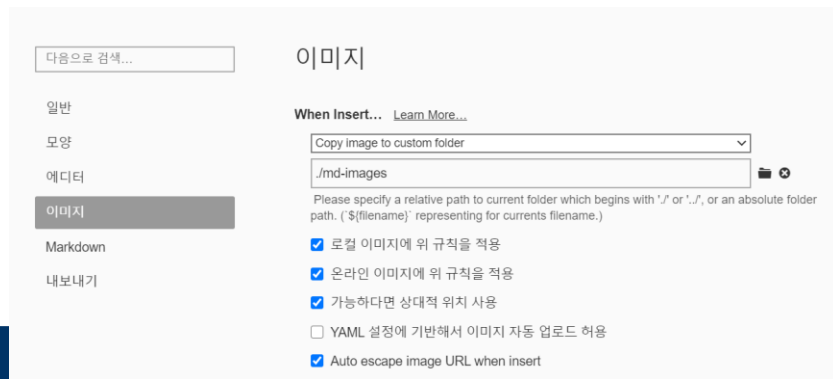
Typora

- 기존 텍스트 에디터(예- visual studio code), IDE 뿐만 아니라 마크다운 전용 에디터를 활용하여 문서를 작성할 수 있음
- Typora는 문법을 작성하면 바로 일반적으로 보이는 모습으로 변하여 처음 작성할 때 많은 도움을 주며, 표 같은 복잡한 문법이나 이미지를 드래그 앤 드랍으로 적용 가능함

<https://typora.io/>

Typora Tip

- 이미지는 아래의 설정을 해두면 마크다운 파일이 있는 위치에 md-images 폴더를 만들고, 가능한 이미지들을 모두 복사하여 상대경로로 표현함
 - 상대 경로 예시: ./md-images/untitle.png
 - 절대 경로 예시: C:/HPHK/Desktop/TIL/untitle.png



마크다운 실습 1. 마크다운 문법 정리

- 지금까지 배운 마크다운 문법으로 마크다운 정리 문서를 만들고 제출하세요.
 - 참고) <https://www.markdownguide.org/cheat-sheet/>

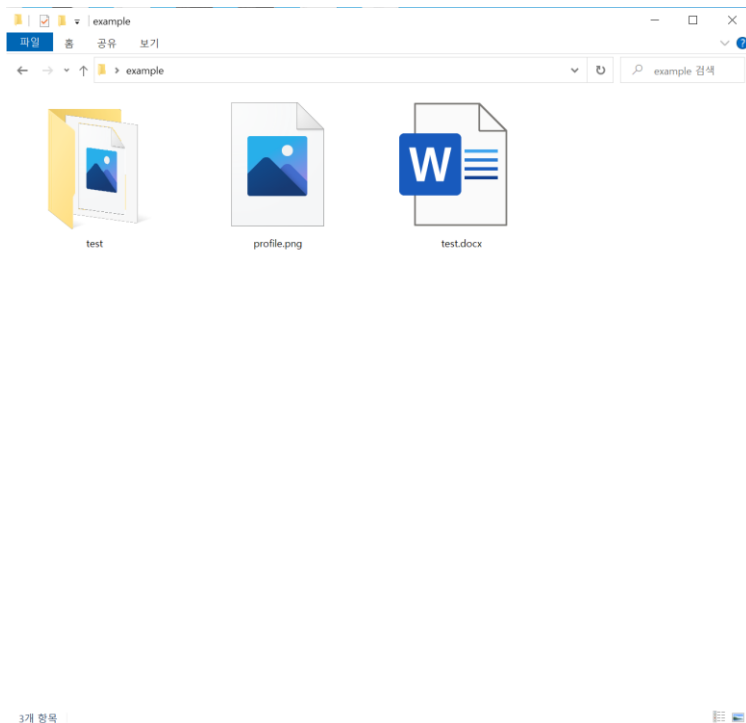
기초 CLI

GIT BASH를 활용하기 위한 기본 개념

CLI (Command Line Interface)

- CLI, 커맨드 라인 인터페이스) 또는 명령어 인터페이스는 가상 터미널 또는 텍스트 터미널을 통해 사용자와 컴퓨터가 상호 작용하는 방식을 뜻한다.
- 작업 명령은 사용자가 툴바 키보드 등을 통해 문자열의 형태로 입력하며, 컴퓨터로부터의 출력 역시 문자열의 형태로 주어진다.
- 이 같은 인터페이스를 제공하는 프로그램을 명령 줄 해석기 또는 셸이라고 부른다. 이를테면, 유닉스 셸(sh, ksh, csh, tcsh, bash 등)과 CP/M, 도스의 command.com("명령 프롬프트") 등이 있다.

CLI (Command Line Interface)



```
MINGW64/c/Users/lec/Desktop/example
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop/example
$ ls
profile.png test/ test.docx
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop/example
$
```

터치 기반의 스마트폰
키보드 마우스 기반의 컴퓨터
‘전혀 다르게’ 생각하고 조작

GUI - 그래픽 기반의 인터페이스

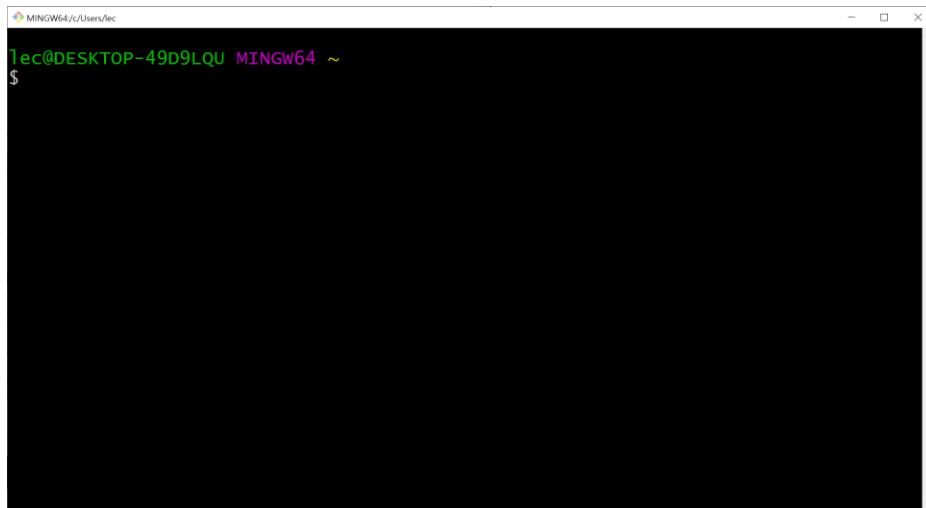
CLI - 명령 기반의 인터페이스

내가 무엇인가를 알고 싶으면,
명령을 하고 그 결과를 읽어야한다.

**불편한 것이 아니라
‘전혀 다르게’ 생각하고 조작하자.**

CLI (Command Line Interface)

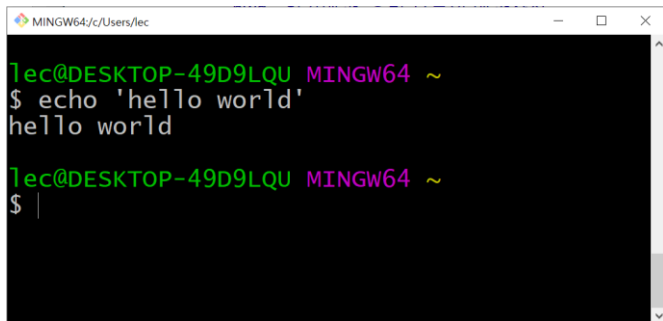
- 프롬프트 기본 인터페이스
 - 컴퓨터 정보
 - 디렉토리
 - \$



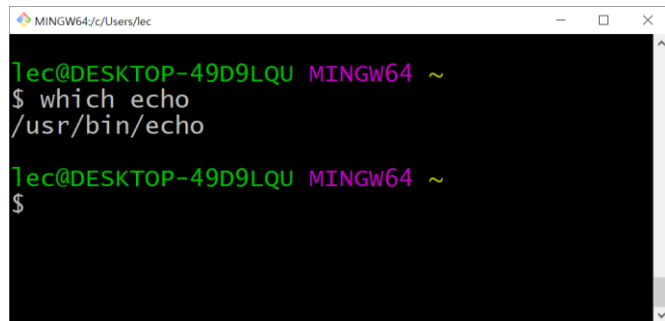
```
lec@DESKTOP-49D9LQU MINGW64 ~  
$
```

CLI (Command Line Interface)

- 명령어 기본 구조
 - 특정 프로그램을 어떠한 인자와 함께 호출하도록 명령
 - 예) echo 프로그램을 'hello world'를 호출하도록



```
MINGW64/c/Users/lec
lec@DESKTOP-49D9LQU MINGW64 ~
$ echo 'hello world'
hello world
lec@DESKTOP-49D9LQU MINGW64 ~
$ |
```



```
MINGW64/c/Users/lec
lec@DESKTOP-49D9LQU MINGW64 ~
$ which echo
/usr/bin/echo
lec@DESKTOP-49D9LQU MINGW64 ~
$
```

디렉토리 관리

- pwd (print working directory) : 현재 디렉토리 출력
- cd 디렉토리이름(change directory) : 디렉토리 이동
 - . : 현재 디렉토리, .. : 상위 디렉토리
- ls (list) : 목록
- mkdir (make directory) : 디렉토리 생성
- touch : 파일 생성
- rm 파일명: 파일 삭제하기
 - rm -r 폴더명 : 폴더 삭제하기

버전관리

버전 관리가 무엇일까





분산 버전 관리 시스템

버전 관리?

버전 관리!

컴퓨터 소프트웨어의 특정 상태

우리가 알고 있는 버전관리

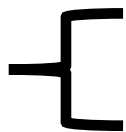
- 일반적인 우리의 버전관리 방식



버전관리



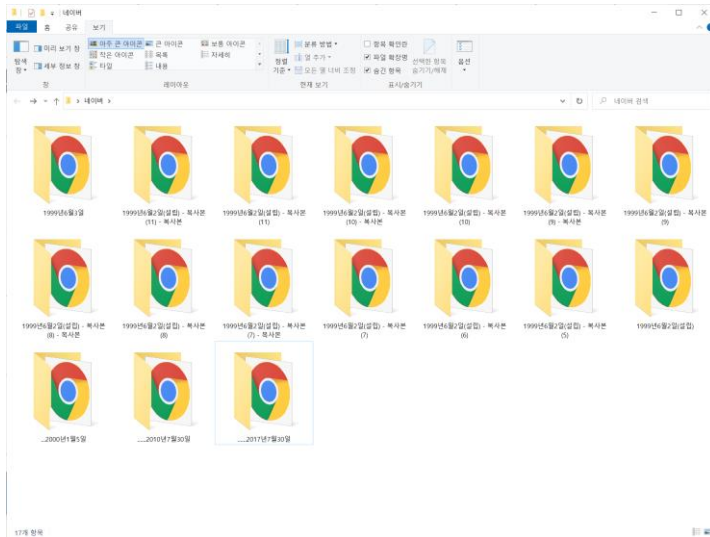
이 자료 간에 뭐가 바뀌었는지
차이(diff)를 알 수 없다.



졸업논문_1차.hwp
졸업논문_수정.hwp
졸업논문_수정_2.hwp
졸업논문_최종본.hwp
졸업논문_진짜최종.hwp
졸업논문_진짜최종이다.hwp
졸업논문_레알진짜킹갓최종.hwp

버전관리

- 1999년에 설립된 네이버의 소스코드 버전 관리(?)
 - 매일 업데이트된다고 할 때, 모든 소스코드를 복제해서 관리하면 용량은? 파일의 개수는?




**파일을 버전별로
저장하여 관리!**
네이버 소스코드는 어떨까?

실제 오픈소스는?

버전관리

- 크로미움(크롬 브라우저의 오픈소스)
 - 최신 버전의 용량 1.58GB
 - 현재까지 1,000,000여개의 커밋(버전) 20,000여개의 릴리즈



Chromium

Chromium is an open-source browser project that aims to build a safer, faster, and more stable way for all users to experience the web.

The project's web site is <https://www.chromium.org>.

To check out the source code locally, don't use `git clone` ! Instead, follow [the instructions on how to get the code](#).

Documentation in the source is rooted in `docs/README.md`.

Learn how to [Get Around the Chromium Source Code Directory Structure](#) .

For historical reasons, there are some small top level directories. Now the guidance is that new top level directories are for product (e.g. Chrome, Android WebView, Ash). Even if these products have multiple executables, the code should be in subdirectories of the product.

If you found a bug, please file it at <https://crbug.com/new>.

master 8 branches 21,117 tags

Go to file Code

chromium-autoroll and Chromium LUCI CQ Roll Chromite from 923a131e2a... 8743f39 40 minutes ago 1,029,316 commits

| | | |
|-----------------|---|----------------|
| android_webview | Add cookie_partition_key argument to CanonicalCookie::Create. | 1 hour ago |
| apps | Merge //base/utl/avalues into //base/son | 7 days ago |
| ash | Refactor how AppListView gets drag state | 2 hours ago |
| base | Check getClassificationStatus() before getConfidenceScore() | 1 hour ago |
| build | Android: Switch build scripts to use python3 (reland) | 1 hour ago |
| build_overrides | Reland "Replace 'blacklist' with 'ignorelist' in //tools/msan/." | 2 months ago |
| buildtools | update re-client/chromium-win-nad-cross to follow nad roll | 18 days ago |
| cc | Add GLES2 usage for creating shared image with Passthrough | 1 hour ago |
| chrome | Chromium Updater system level install should have different permissions | 40 minutes ago |
| chromecast | Glue Together Pieces of CastRuntimeService | 23 hours ago |
| chromeos | Personalization: button background polish | 1 hour ago |
| cloud_print | Update OWNERS for translation artifacts | 11 days ago |
| code_labs | [owners] Ensure OWNERS file ends with newline in // | 2 months ago |
| components | Add null check for embedding app's package name | 43 minutes ago |
| content | Add cookie_partition_key argument to CanonicalCookie::Create. | 1 hour ago |

About

The official GitHub mirror of the Chromium source

[chromium.googlesource.com/chromium...](#)

[Readme](#)

[BSD-3-Clause License](#)


Releases

21,117 tags

Packages

No packages published

Contributors 2,178



+ 2,167 contributors

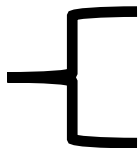
**모든 버전의 용량
약 25GB
하나의 폴더**

버전 관리를 해주는 Git 덕분!

버전관리



차이(diff)와 수정 이유를
메시지로 남길 수 있다.



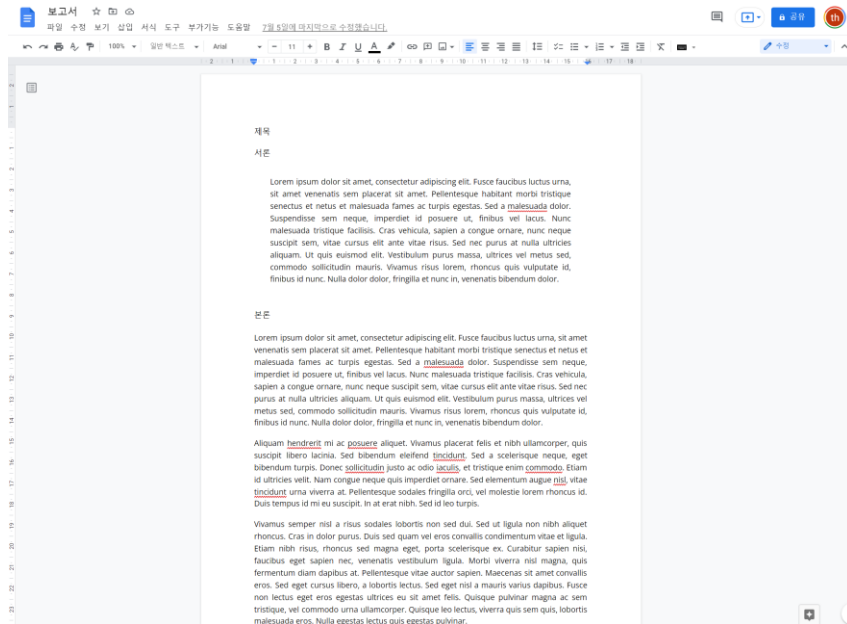
벼대 코드구성
메인 기능 구현
로그인 기능 구현
채팅 기능 구현
디자인 적용
...

현재 파일들을 안전한 상태로
과거 모습 그대로 복원 가능 (반대도 마찬가지)



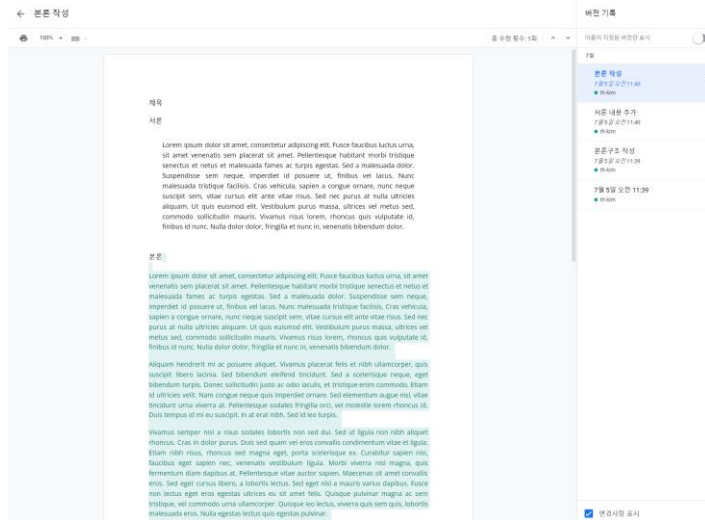
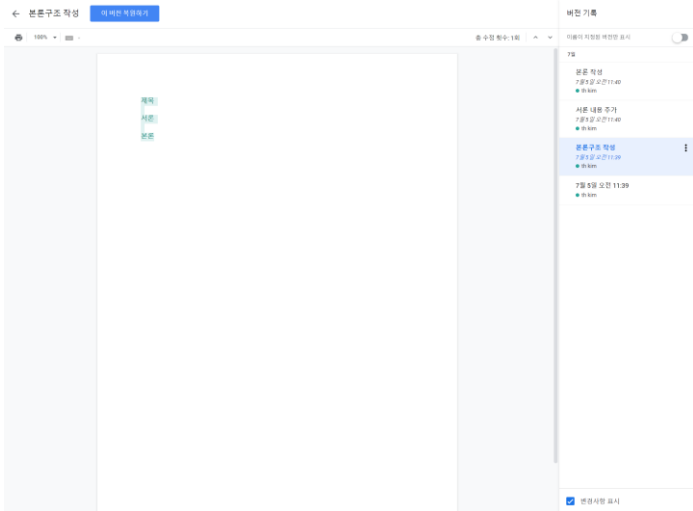
버전관리

- Google Document 버전 관리



버전관리

- Google Document 버전 관리
 - 문서는 하나지만 버전이 기록되어 있으면, 이전 시점을 조회하거나 복원시킬 수도 있음



버전관리시스템

- In software engineering, version control (also known as revision control, source control, or source code management) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information.
- 버전관리, 소스코드 관리란 동일한 정보에 대한 여러 버전을 관리하는 것을 말한다.

https://en.wikipedia.org/wiki/Version_control
<https://ko.wikipedia.org/wiki/버전관리시스템>

버전관리시스템

| V · T · E | | | 버전 관리 소프트웨어 | [접기] |
|--|--|---|-------------|------|
| 연도로 표시된 부분은 최초 안정판의 날짜를 가리킨다. 별표(*)로 표시된 시스템은 더 이상 유지보수가 되지 않거나 EOL 날짜가 예정되어 있음을 나타낸다. | | | | |
| 로컬 전용 | 자유/오픈 소스 | RCS (1982) · SCCS (1972) | | |
| | 사유 | PVCS (1985) · QVCS* (1991) | | |
| 클라이언트-서버 | 자유/오픈 소스 | CVS (1986, C의 경우 1990) · CVSNT (1998) · QVCS 엔터프라이즈 (1998) · 서브버전 (2000) | | |
| | 사유 | AccuRev SCM (2002) · ClearCase (1992) · CMVC* (1994) · Dimensions CM (1980년대) · DSEE* (1984) · Endavor (1980년대) · Integrity (2001) · Panvalet (1970년대) · 퍼포스 헬릭스 (1995) · Software Change Manager (1970년대) · 스타팀 (1995) · 서라운드 SCM (2002) · Synergy (1990) · Team Concert (2008) · 팀 파운데이션 서버 (2005) · 마이크로소프트 비주얼 스튜디오 (2014) · Vault (2003) · 비주얼 소스세이프* (1994) | | |
| 분산 | 자유/오픈-소스 | ArX* (2003) · 비트키퍼 (1998) · Codeville* (2005) · Darcs (2002) · DCVS* (2002) · Fossil (2007) · 깃 (2005) · GNU arch* (2001) · Bazaar (2005) · 머큐리얼 (2005) · 모노톤 (2003) · SVK* (2003) · Veracity (2010) | | |
| | 사유 | TeamWare* (1990년대) · Code Co-op (1997) · 플라스틱 SCM (2006) · 팀 파운데이션 서버 (2013) · 마이크로소프트 비주얼 스튜디오 (2014) | | |
| 개념 | 브랜치 · 포크 · 체인지셋 · 커밋 (게이티드 커밋) · Interleaved deltas · 델타 압축 · 데이터 비교 · 머지 · 저장소 · 태그 · Trunk | | | |

Git 기초 흐름

Git

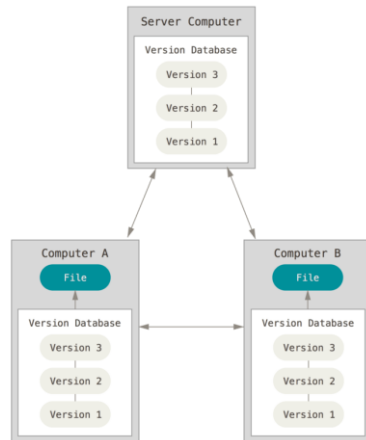
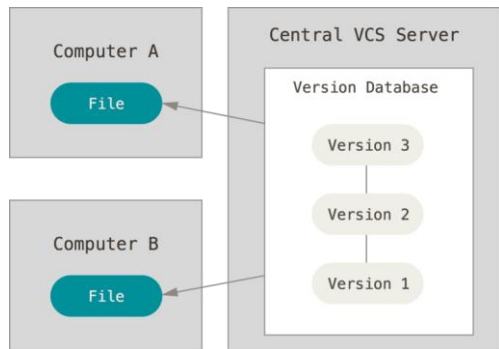
- Git은 분산버전관리시스템으로 코드의 버전을 관리하는 도구
- 2005년 리눅스 커널을 위한 도구로 리누스 토르발스가 개발
- 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율



분산 버전 관리?

분산버전관리시스템(DVCS)

- 중앙집중식버전관리시스템은 중앙에서 버전을 관리하고 파일을 받아서 사용
- 분산버전관리시스템은 원격 저장소(remote repository)를 통하여 협업하고, 모든 히스토리를 클라이언트들이 공유



저장소를 만들어보자!

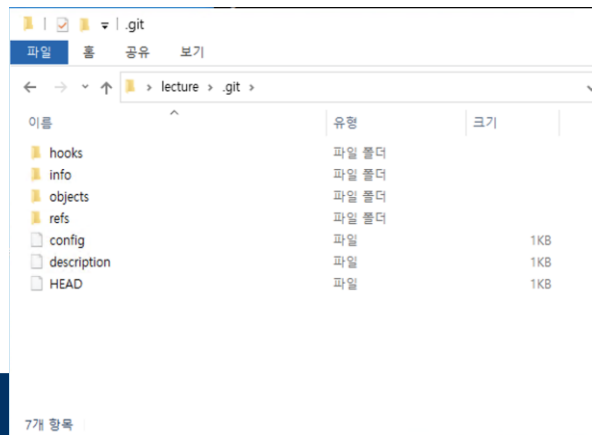
기본 명령어 – init

\$ git init

- 특정 폴더를 git 저장소(repository)를 만들어 git으로 관리
 - .git 폴더가 생성되며
 - git bash에서는 (master)라는 표기를 확인할 수 있음

```
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/lecture
$ git init
Initialized empty Git repository in C:/Users/takhe/Desktop/lecture/.git/

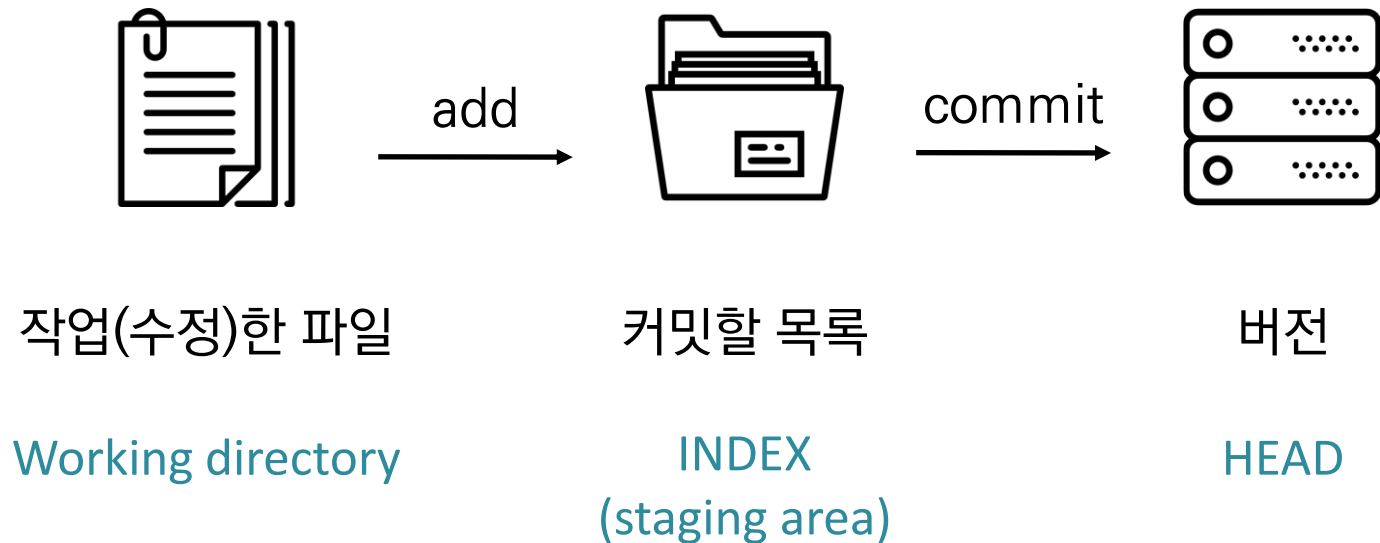
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/lecture (master)
$ ls -al
total 44
drwxr-xr-x 1 lecture 197121 0 Mar 29 04:19 ./
drwxr-xr-x 1 lecture 197121 0 Mar 29 04:16 ../
drwxr-xr-x 1 lecture 197121 0 Mar 29 04:19 .git/
```



버전은 어떻게 기록할까?

기본 흐름

- 1) 작업하면 2) add하여 Staging area에 모아 3) commit으로 버전 기록



1. 작업을 하고
2. 변경된 파일을 모아 (add)
3. 버전으로 남긴다. (commit)

Working Directory

파일의 변경사항



untracked

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록

Repository

커밋(버전)들이 기록되는 곳

Working Directory

파일의 변경사항



untracked



```
$ git add
```

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록

Repository

커밋(버전)들이 기록되는 곳

Working Directory

파일의 변경사항



untracked

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록



staged

\$ git add

Repository

커밋(버전)들이 기록되는 곳

Working Directory

파일의 변경사항



untracked

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록



staged

\$ git commit

Repository

커밋(버전)들이 기록되는 곳

Working Directory

파일의 변경사항

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록



staged

Repository

커밋(버전)들이 기록되는 곳



\$ git commit

기본 명령어 - add

\$ git add <file>

- working directory상의 변경 내용을 staging area에 추가하기 위해 사용
 - untracked 상태의 파일을 staged로 변경
 - modified 상태의 파일을 staged로 변경

```
MINGW64/c/Users/takhe/Desktop/git-1
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        b.txt

nothing added to commit but untracked files present (use "git add" to track)
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ |
```

```
MINGW64/c/Users/takhe/Desktop/git-1
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git add b.txt

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   b.txt

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$
```

기본 명령어 – commit

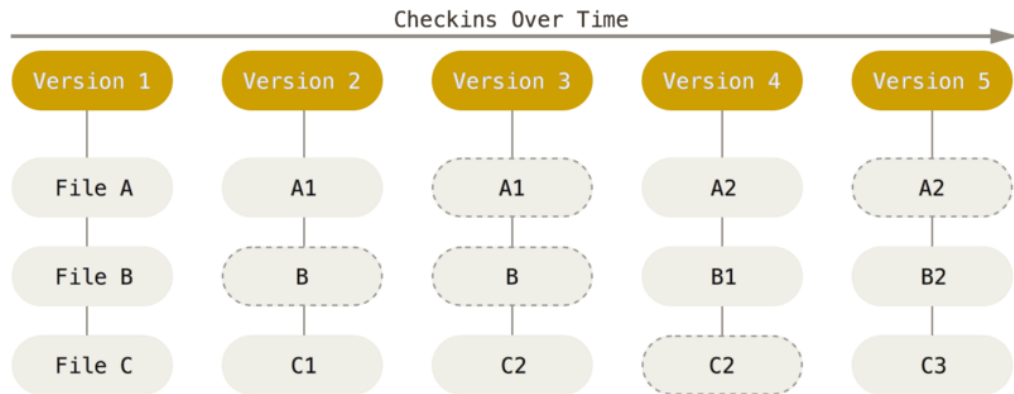
`$ git commit -m '커밋메시지'`

- staged 상태의 파일들을 커밋을 통해 버전으로 기록
- SHA-1 해시를 사용하여 40자 길이의 체크섬을 생성하고, 이를 통해 고유한 커밋을 표기
- 커밋 메시지는 변경 사항을 나타낼 수 있도록 명확하게 작성해야 함

```
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/lecture (master)
$ git commit -m 'First commit'
[master (root-commit) 14a0074] First commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
```

기본 명령어 – commit 이해하기

- Git은 데이터를 파일 시스템의 스냅샷으로 관리하고 매우 크기가 작음
- 파일이 달라지지 않으면 성능을 위해 파일을 새로 저장하지 않음
- 기존의 델타 기반 버전 관리시스템과 가장 큰 차이를 가짐



<https://git-scm.com/book/ko/v2/시작하기-Git-기초>

Working Directory

파일의 변경사항



a



b



c

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록

Repository

커밋(버전)들이 기록되는 곳

Working Directory

파일의 변경사항



\$ git add

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록



Repository

커밋(버전)들이 기록되는 곳

Working Directory

파일의 변경사항



a

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록



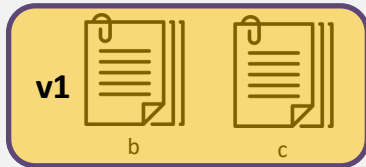
b



c

Repository

커밋(버전)들이 기록되는 곳



\$ git commit

Working Directory

파일의 변경사항



a



```
$ git add
```

Staging Area

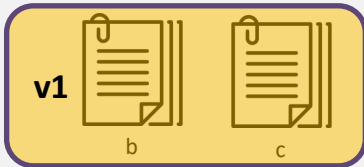
버전으로 기록하기 위한
파일 변경사항의 목록



a

Repository

커밋(버전)들이 기록되는 곳



Working Directory

파일의 변경사항

Staging Area

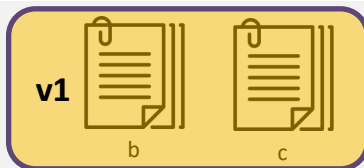
버전으로 기록하기 위한
파일 변경사항의 목록



```
$ git commit
```

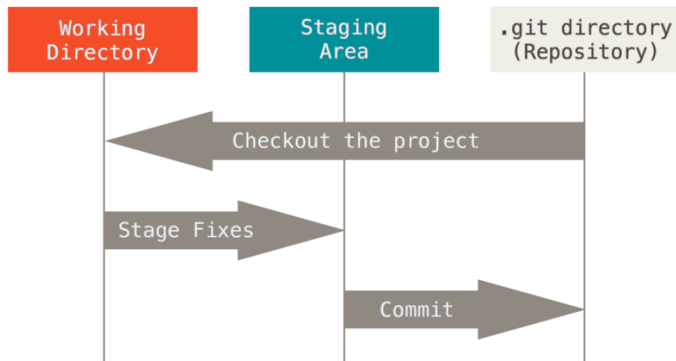
Repository

커밋(버전)들이 기록되는 곳



기본 흐름

- Git은 파일을 modified, staged, committed로 관리
 - modified : 파일이 수정된 상태 (add 명령어를 통하여 staging area로)
 - staged : 수정한 파일을 곧 커밋할 것이라고 표시한 상태 (commit 명령어로 저장소)
 - committed : 커밋이 된 상태



<https://git-scm.com/book/ko/v2/시작하기-Git-기초>

현재 상태를 알고 싶어요.

Working Directory

파일의 변경사항

```
$ git status
```

Staging Area

버전으로 기록하기 위한
파일 변경사항의 목록

```
$ git log
```

Repository

커밋(버전)들이 기록되는 곳

기본 명령어 - log

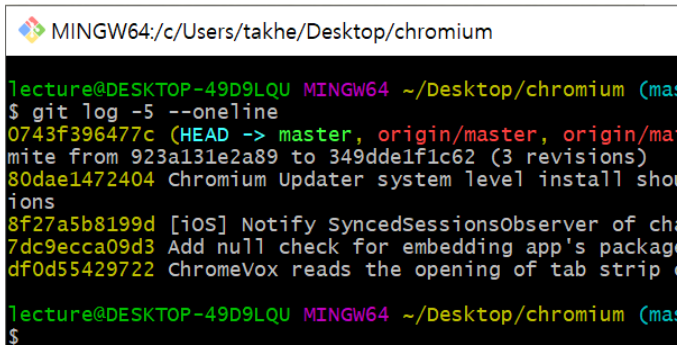
\$ git log

- 현재 저장소에 기록된 커밋을 조회
- 다양한 옵션을 통해 로그를 조회할 수 있음

\$ git log -1

\$ git log --online

\$ git log -2 --online

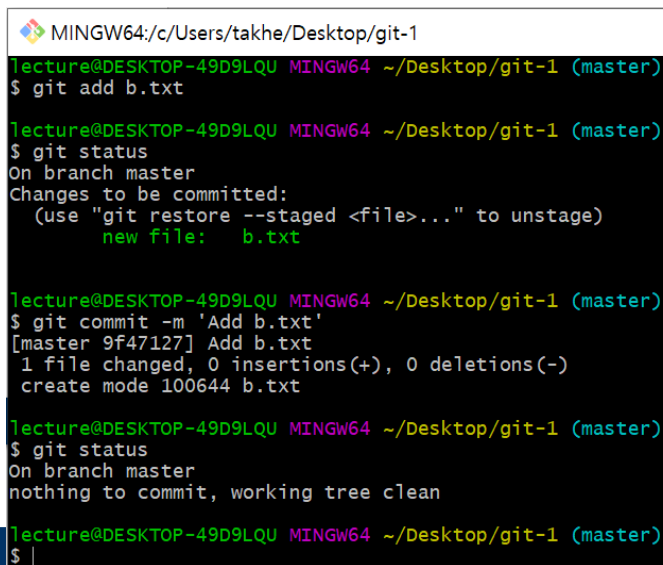
A terminal window titled 'MINGW64:/c/Users/takhe/Desktop/chromium' showing the output of the command '\$ git log -5 --online'. The output lists five commits with their hashes, branch names, and commit messages. The first commit is '0743f396477c (HEAD -> master, origin/master, origin/main) mite from 923a131e2a89 to 349dde1f1c62 (3 revisions)'. The second is '80dae1472404 Chromium Updater system level install show'. The third is '8f27a5b8199d [iOS] Notify SyncedSessionsObserver of ch'. The fourth is '7dc9ecca09d3 Add null check for embedding app's packag'. The fifth is 'df0d55429722 ChromeVox reads the opening of tab strip c'. The prompt '\$' is visible at the bottom.

```
MINGW64:/c/Users/takhe/Desktop/chromium
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/chromium (ma
$ git log -5 --online
0743f396477c (HEAD -> master, origin/master, origin/ma
mite from 923a131e2a89 to 349dde1f1c62 (3 revisions)
80dae1472404 Chromium Updater system level install sho
ions
8f27a5b8199d [iOS] Notify SyncedSessionsObserver of ch
7dc9ecca09d3 Add null check for embedding app's packag
df0d55429722 ChromeVox reads the opening of tab strip c
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/chromium (ma
$
```

기본 명령어 – status

\$ git status

- Git 저장소에 있는 파일의 상태를 확인하기 위하여 활용
 - 파일의 상태를 알 수 있음
 - Untracked files
 - Changes not staged for commit
 - Changes to be committed
 - Noting to commit, working tree clean



```
MINGW64:/c:/Users/takhe/Desktop/git-1
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git add b.txt

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   b.txt

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git commit -m 'Add b.txt'
[master 9f47127] Add b.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 b.txt

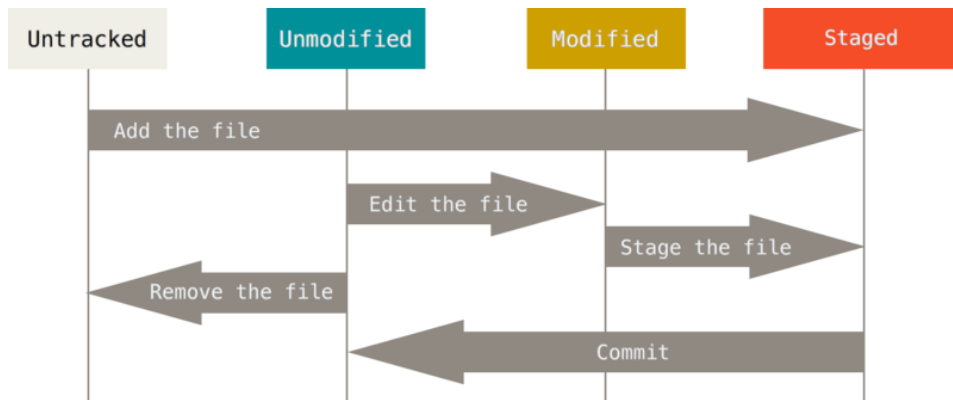
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git status
On branch master
nothing to commit, working tree clean

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$
```

파일 라이프사이클

- Status로 확인할 수 있는 파일의 상태
 - Tracked : 이전부터 버전으로 관리되고 있는 파일
 - Unmodified : git status에 나타나지 않음
 - Modified : Changes not staged for commit
 - Staged : Changes to be committed
 - Untracked : 버전으로 관리된 적 없는 파일 (파일을 새로 만든 경우)

파일 라이프사이클



<https://git-scm.com/book/ko/v2/Git의-기초-수정하고-저장소에-저장하기>

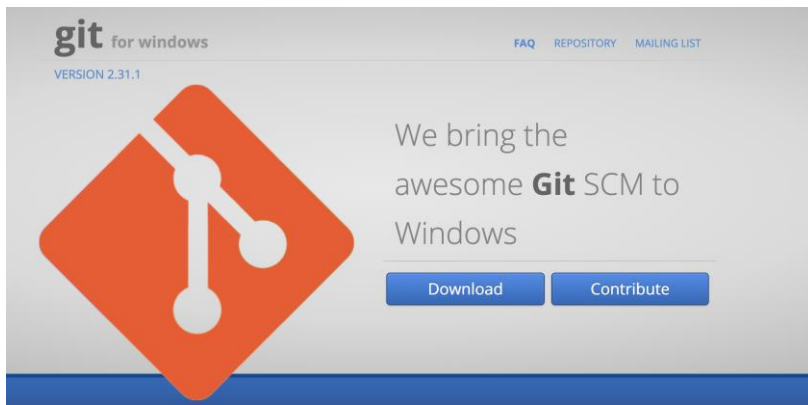
Git 기초 명령어

| 명령어 | 내용 |
|-------------------------|-------------------|
| git init | 로컬 저장소 생성 |
| git add <파일명> | 특정 파일/폴더의 변경사항 추가 |
| git commit -m '<커밋메시지>' | 커밋 (버전 기록) |
| git status | 상태 확인 |
| git log | 버전 확인 |

Git 활용하기

- Git bash 설치

<https://gitforwindows.org/>
<https://bit.ly/sw-install-guide>



Git 설정 파일 (config)

- 사용자 정보 (commit author) : 커밋을 하기 위해 반드시 필요
 - `git config --global user.name "username"`
 - Github에서 설정한 username으로 설정
 - `git config --global user.email "my@email.com"`
 - Github에서 설정한 email로 설정
- 설정 확인
 - `git config -l`
 - `git config --global -l`
 - `git config user.name`

Git 설정 파일 (config)

- —system
 - /etc/gitconfig
 - 시스템의 모든 사용자와 모든 저장소에 적용(관리자 권한)
- —global
 - ~/.gitconfig
 - 현재 사용자에게 적용되는 설정
- —local
 - .git/config
 - 특정 저장소에만 적용되는 설정

git 실습 1. 저장소 만들고 첫번째 버전 기록하기

- 1) 바탕화면에 test 폴더를 만들고 git 저장소를 만들기
- 2) a.txt 파일 넣고 커밋하기
- 3) 임의의 파일을 만들고 커밋하기
- 4) a.txt 파일 수정하고 커밋하기

각 단계별로 status와 log를 보세요.

- 주의
 - 어떠한 파일도 생성하지 않으면 당연히도 버전을 만들 대상이 없습니다.

(추가) 디렉토리에 대한 이해

- CLI에서 현재 폴더의 목록을 보면 다음과 같다.
 - . : 현재 디렉토리
 - 그래서 git add . 이 현재 폴더에 대한 모든 파일의 변경사항을 add 하는 것!
 - .. : 상위 디렉토리

```
$ ls -al
total 16
drwxr-xr-x 1 lec 197121 0  7월 15 11:54 ./ # <---
drwxr-xr-x 1 lec 197121 0  7월 15 11:32 ../ # <---
drwxr-xr-x 1 lec 197121 0  7월 15 11:26 .git/
-rw-r--r-- 1 lec 197121 0  7월 15 11:26 b.txt
-rw-r--r-- 1 lec 197121 0  7월 15 11:26 c.txt
drwxr-xr-x 1 lec 197121 0  7월 15 11:54 images/
drwxr-xr-x 1 lec 197121 0  7월 15 11:54 my_folder/
```

(추가) 디렉토리에 대한 이해 (상대 경로 / 절대 경로)

- README.md에서 b.png를 활용하기 위해서는?
 - 절대 경로 : C:/TIL/images/markdown/b.png
 - 상대 경로 : ./images/markdown/b.png

```
C: /  
  TIL/  
    images/  
      markdown/  
        a.png  
        b.png  
      README.md
```

(추가) 디렉토리에 대한 이해 (상대 경로 / 절대 경로)

- style.css에서 a.png를 활용하기 위해서는?
 - style.css 위치는 my_web의 css!
 - 즉, 상대 경로로 표현하면 ../images/a.png

```
my_web/  
  images/  
    a.png  
  css/  
    style.css  
  index.html
```


Git 실습

TIL

git 실습 2. TIL 프로젝트 관리

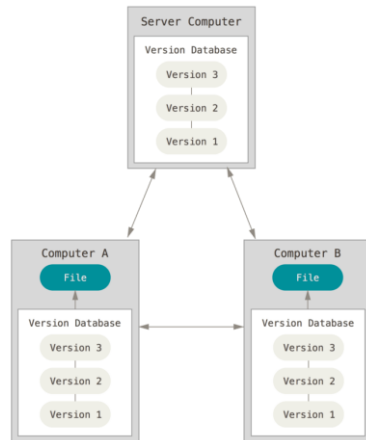
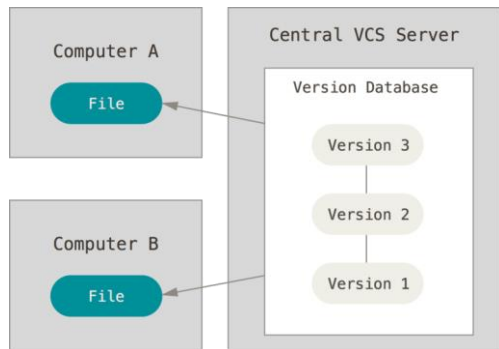
- 1) TIL 폴더를 만들고 git 저장소를 만들기
- 2) README.md 파일을 만들고 커밋하기
- 3) 오늘 작성한 마크다운 파일을 옮기고 커밋하기
 - 마크다운 파일별로 각각 커밋을 진행해보세요!
- 4) 이제까지 배운 내용들을 정리하고 커밋하기

원격저장소 활용하기

GITHUB

분산버전관리시스템(DVCS)

- 중앙집중식버전관리시스템은 중앙에서 버전을 관리하고 파일을 받아서 사용
- 분산버전관리시스템은 원격 저장소(remote repository)를 통하여 협업하고, 모든 히스토리를 클라이언트들이 공유



원격저장소 (Remote Repository)

- 네트워크를 활용한 저장소





github
SOCIAL CODING

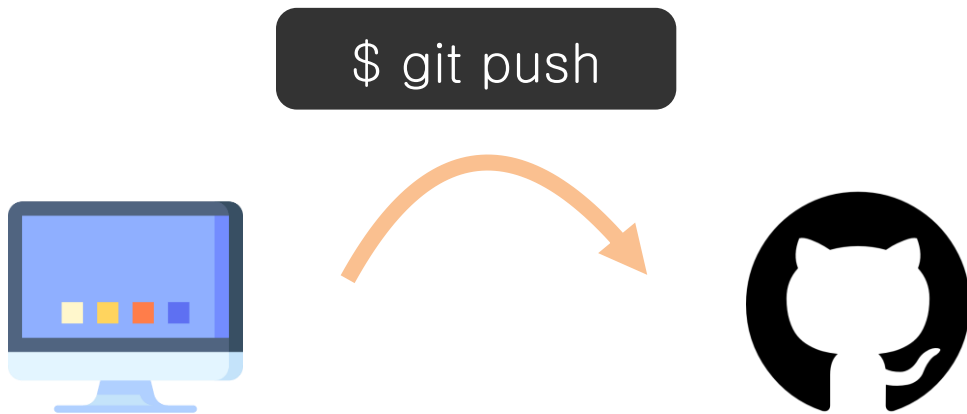
Git은 을 관리한다.

Git은 버전을 관리한다.

GitHub도 버전을 관리한다.

원격저장소 (Remote Repository) 기본 흐름

- 로컬 저장소의 버전을 원격저장소로 보낸다.



원격저장소 (Remote Repository) 기본 흐름

- 로컬 저장소의 버전을 원격저장소로 보낸다.

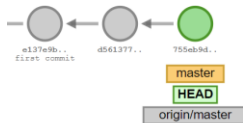
Local Repository
HEAD: master



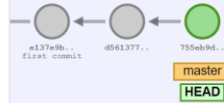
origin



Local Repository
HEAD: master



origin



원격저장소 (Remote Repository) 기본 흐름

- 로컬 저장소의 버전을 원격저장소로 보낸다.

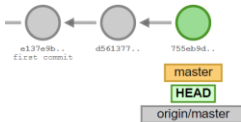
Local Repository
HEAD: master



origin

master
HEAD

Local Repository
HEAD: master



\$ git push

origin

master
HEAD

원격저장소 (Remote Repository) 기본 흐름

- 원격저장소의 버전을 로컬 저장소로 가져온다.



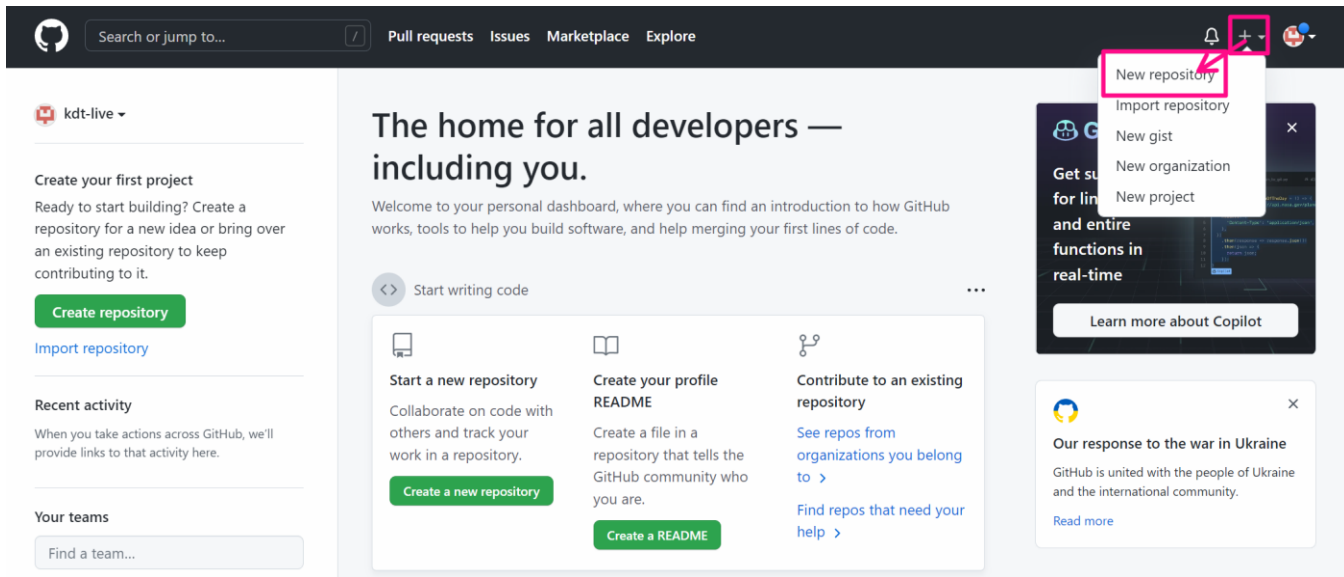
\$ git pull

실제로 활용해보자!

원격저장소를 만들고,
저장소 설정을 하고
로컬저장소의 버전을 push한다.

GitHub에서 원격 저장소 만들기

- New Repository



GitHub에서 원격 저장소 만들기

- 저장소 설정하기

The screenshot shows the GitHub 'Create a new repository' page. The page has a dark header with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main content area is titled 'Create a new repository' and includes a sub-header 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.'.

The form contains several sections:

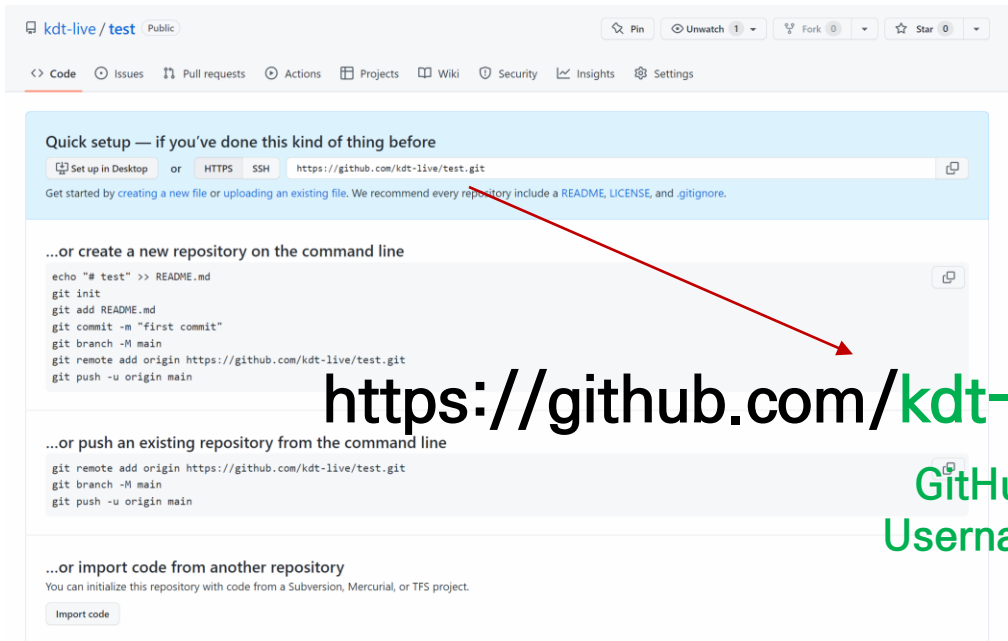
- Owner:** A dropdown menu showing 'kdt-live'.
- Repository name:** A text input field containing 'test', with a green checkmark to its right. This field is highlighted with a pink box and labeled '1. repository 이름 설정'.
- Description (optional):** A text input field. This section is highlighted with a pink box and labeled '2. 저장소 설명(옵션)'.
- Visibility:** Two radio buttons: 'Public' (selected) and 'Private'. The 'Public' option is highlighted with a pink box and labeled '3. 공개 여부 설정'.
- Initialize this repository with:** A section with a checkbox for 'Add a README file'.
- Add .gitignore:** A section with a dropdown menu for 'gitignore template' set to 'None'.
- Choose a license:** A section with a dropdown menu for 'License' set to 'None'.
- Create repository:** A green button at the bottom, highlighted with a pink box and labeled '4. 저장소 생성'.

Annotations on the right side of the form:

- A red arrow points from the 'Add .gitignore' section to the text: 'Git/GitHub을 이해하기 전에는 설정을 하지 않습니다. 체크 없이 모두 None!'.

GitHub에서 원격 저장소 만들기

- 확인하기



<https://github.com/kdt-live/test.git>

GitHub
Username

저장소
이름

<https://github.com/kdt-live/test.git>

GitHub
Username

저장소
이름

로컬저장소의 버전을 원격저장소로 보내주기

Quick setup — if you've done this kind of thing before

Set up in Desktop or **HTTPS** SSH `https://github.com/kdt-live/test.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "First commit"
git remote add origin https://github.com/kdt-live/test.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/kdt-live/test.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can use the `git clone` command to create a new repository from an existing one. For more information, see [Importing an existing repository](#).

\$ git remote add origin https://github.com/kdt-live/test.git

ProTip! Use the URL for this page when adding GitHub as a remote.

```
$ git remote add origin https://github.com/kdt-live/test.git
```

원격저장소

origin으로 추가

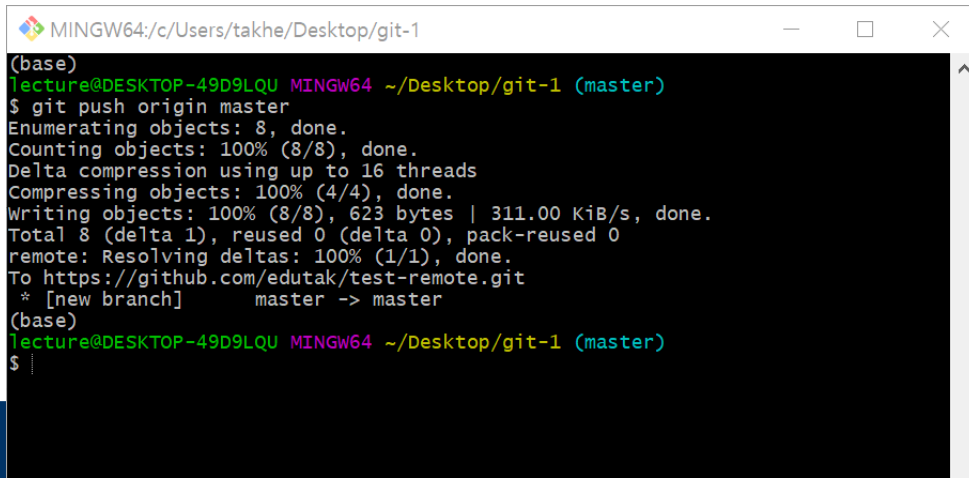
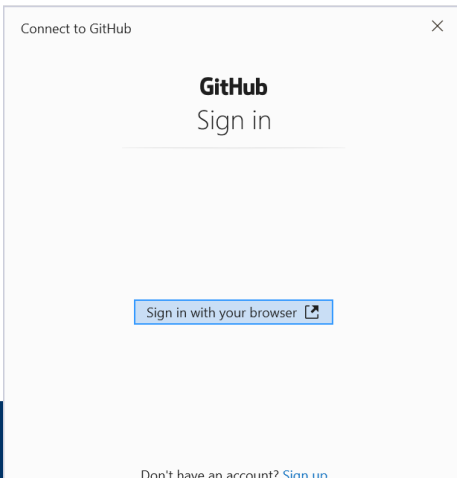
GitHub
Username

저장소
이름

원격저장소 활용 명령어 - push

\$ git push <원격저장소이름> <브랜치이름>

- 원격 저장소로 로컬 저장소 변경 사항(커밋)을 올림(push)
- 로컬 폴더의 파일/폴더가 아닌 저장소의 버전(커밋)이 올라감



원격저장소 활용 명령어 - pull

\$ git pull <원격저장소이름> <브랜치이름>

- 원격 저장소로부터 변경된 내역을 받아와서 이력을 병합함

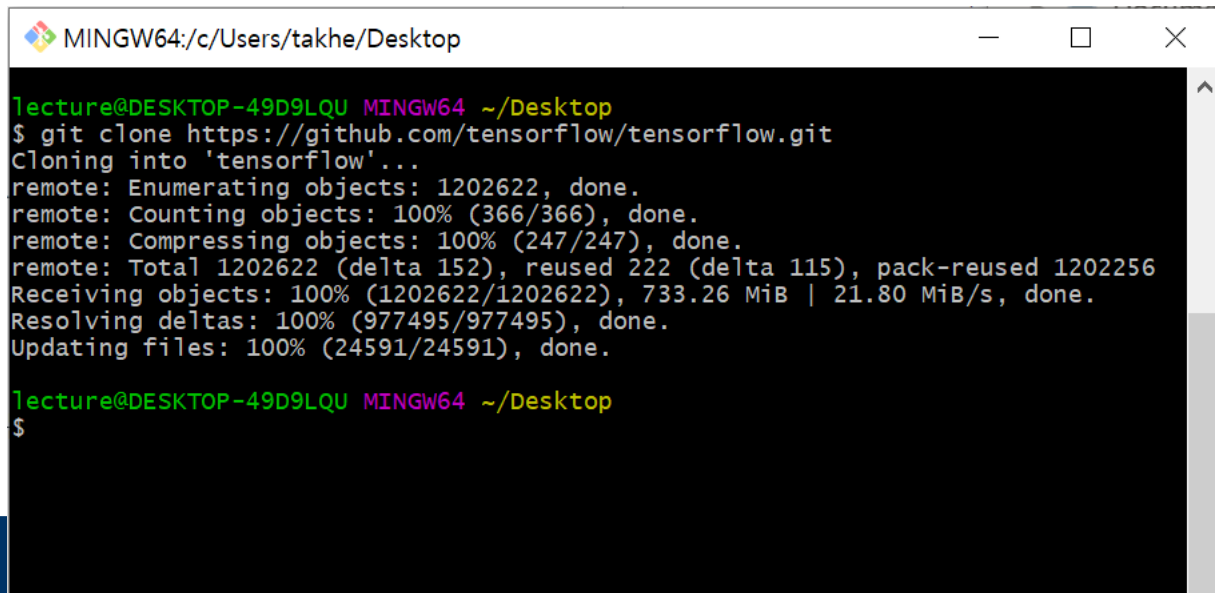
```
MINGW64:/c/Users/takhe/Desktop/git-1
(base)
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 718 bytes | 55.00 KiB/s, done.
From https://github.com/edutak/test-remote
* branch      master      -> FETCH_HEAD
   8499447..8196332 master  -> origin/master
Updating 8499447..8196332
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
(base)
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ |
```

로컬 저장소 없는데
가져오고 싶어요.

원격저장소 활용 명령어 - clone

\$ git clone <원격저장소주소>

- 원격 저장소를 복제하여 가져옴

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/takhe/Desktop'. The terminal shows the execution of the 'git clone' command to clone the TensorFlow repository from GitHub. The output displays progress for enumerating, counting, and compressing objects, as well as receiving and resolving deltas. The command is executed from the user's desktop directory.

```
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop
$ git clone https://github.com/tensorflow/tensorflow.git
Cloning into 'tensorflow'...
remote: Enumerating objects: 1202622, done.
remote: Counting objects: 100% (366/366), done.
remote: Compressing objects: 100% (247/247), done.
remote: Total 1202622 (delta 152), reused 222 (delta 115), pack-reused 1202256
Receiving objects: 100% (1202622/1202622), 733.26 MiB | 21.80 MiB/s, done.
Resolving deltas: 100% (977495/977495), done.
Updating files: 100% (24591/24591), done.

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop
$
```

원격저장소 설정 기본 명령어

| 명령어 | 내용 |
|------------------------------|----------------------------|
| git clone <url> | 원격 저장소 복제 |
| git remote -v | 원격저장소 정보 확인 |
| git remote add <원격저장소> <url> | 원격저장소 추가
(일반적으로 origin) |
| git remote rm <원격저장소> | 원격저장소 삭제 |
| git push <원격저장소> <브랜치> | 원격저장소에 push |
| git pull <원격저장소> <브랜치> | 원격저장소로부터 pull |