

# Git/GitHub 특강

해피해킹 김탁희

# Git? GitHub?

**SNL**  
김치총

**tvN**  
생방송

아니 X발 무슨 다 경력직만 뽑으면

면접 전쟁  
(An Job Jeongjeong)

# 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 Github에서 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:

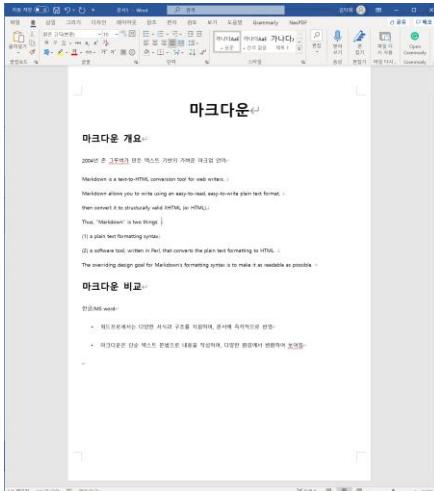
- (1) **a plain text formatting syntax**
- (2) a software tool, written in Perl, **that converts the plain text formatting to HTML**.

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)

A screenshot of Visual Studio Code showing the file "마크다운.md". The content of the file is the same as the Word document: "# 마크다운", followed by several lines of explanatory text about Markdown, and then the text "# 마크다운" again. The Visual Studio Code interface, including the status bar at the bottom, is visible.

# 마크다운 특징

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

```
C:\> Users > taeho > Desktop > 마크다운.md > # 마크다운
1 # 마크다운
2
3 ## 개요
4
5 2004년 존 그루버가 만든 빅스트 기반의 가벼운 마크업 언어
6
7 > 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).
8 Thus, "Markdown" is two things: (1) a plain text formatting syntax; and (2) a software tool, written in Perl, that converts the plain text formatting to HTML.
9 See the Syntax page for details pertaining to Markdown's formatting syntax.
10 The overriding design goal for Markdown's formatting syntax is to make it as readable as
11 possible.
12
13 [문서] (https://daringfireball.net/projects/markdown/)
14
15 ## 내고
16
17 한글/MS word
18
19 • 워드프로세서는 다양한 서식과 구조를 지원하여, 문서에 즉각적으로 반영
20
21 • 마크다운은 단순 텍스트 문법으로 내용을 작성하며, 다양한 환경에서 변환하여 보여짐
22
23
```

# 마크다운

## 개요

2004년 존 그루버가 만든 텍스트 기반의 가벼운 마크업 언어

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; and (2) a software tool, written in Perl, that converts the plain text formatting to HTML.

See the Syntax page for details pertaining to Markdown's formatting syntax.

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

## 한글/MS word

- 워드프로세서는 다양한 서식과 구조를 지원하여, 문서에 즉각적으로 반영
- 마크다운은 단순 텍스트 문법으로 내용을 작성하며, 다양한 환경에서 변환하여 보여짐

Search or jump to... Pull requests Issues Marketplace Explore

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main · branch · tag

Go to file Add file · Go back · About

README.md Create README.md 21 seconds ago

README.md

## 마크다운

## 개요

2004년 존 그루버가 만든 텍스트 기반의 가벼운 마크업 언어

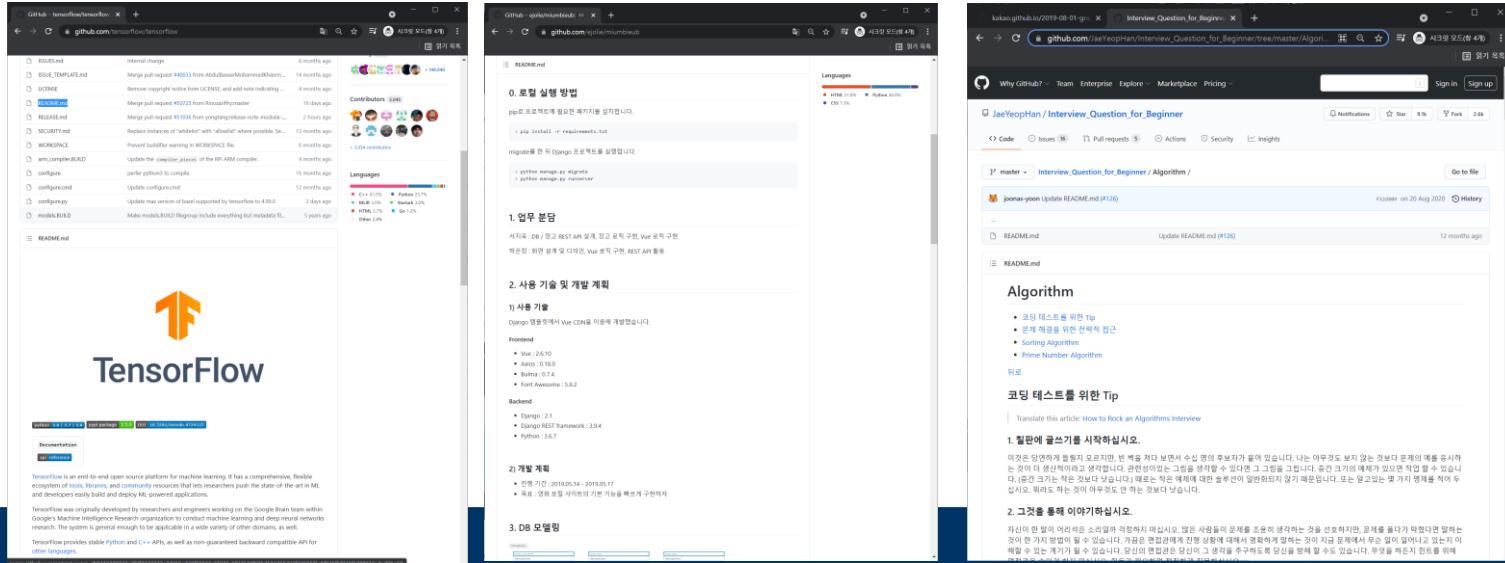
Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML). Thus, "Markdown" is two things: (1) a plain text formatting syntax; and (2) a software tool, written in Perl, that converts the plain text formatting to HTML. See the Syntax page for details pertaining to Markdown's formatting syntax. The overriding design goal for Markdown's formatting syntax is to make it as readable as possible.

## 한글/MS word

- 워드프로세서는 다양한 서식과 구조를 지원하여, 문서에 즉각적으로 반영
- 마크다운은 단순 텍스트 문법으로 내용을 작성하여, 다양한 환경에서 변환하여 보여짐

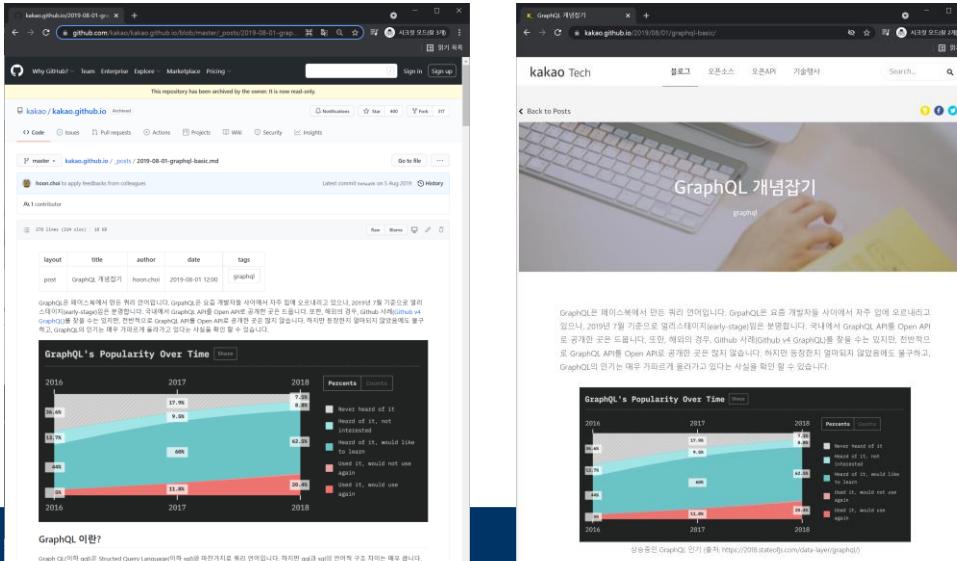
# 마크다운 활용 예 – README.md

- 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

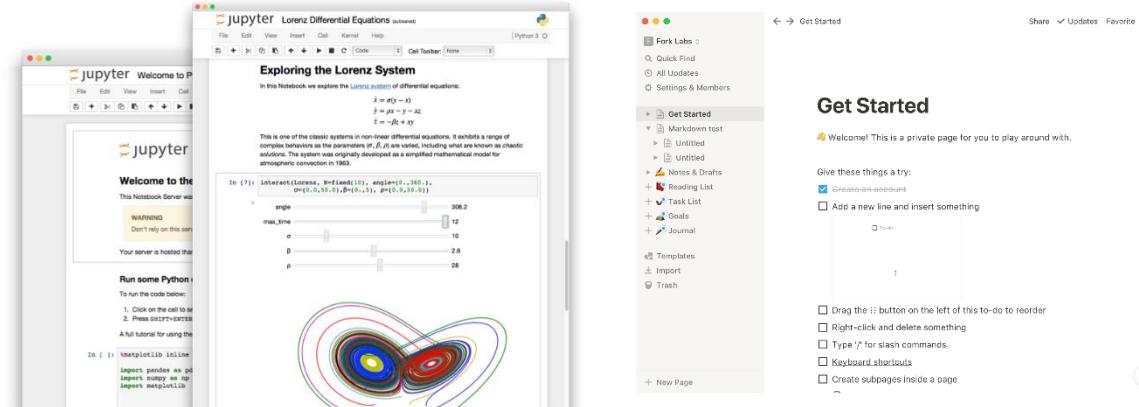
- 다양한 기술블로그에서는 정적사이트생성기(Static site generator)
  - Jekyll, Gatsby, Hugo, Hexo 등을 통해 작성된 마크다운을 HTML, CSS, JS 파일 등으로 변환하고
  - Github pages 기능을 통해 호스팅 (외부 공개)



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

# 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
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
- Tip) Typora에서 tab으로 하위 항목, shift+tab으로 상위 항목

Markdown	HTML	Rendered Output
1. First item		
2. Second item		
3. Third item		
4. Fourth item	<pre>&lt;ol&gt; &lt;li&gt;First item&lt;/li&gt; &lt;li&gt;Second item&lt;/li&gt; &lt;li&gt;Third item&lt;/li&gt; &lt;li&gt;Fourth item&lt;/li&gt; &lt;/ol&gt;</pre>	<ol style="list-style-type: none"><li>1. First item</li><li>2. Second item</li><li>3. Third item</li><li>4. Fourth item</li></ol>

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

<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<br>`nano`. | At the command prompt, type<br><code>nano</code>. | At the command prompt,<br>type nano. |

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

# 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](<a href="https://duckduckgo.com">https://duckduckgo.com</a>).
```

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
  - Typora Tip) ctrl + t로 생성 가능

|           |             |
|-----------|-------------|
| 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 <strong>bold text</strong>. | I just love <b>bold text</b> . |
| I just love __bold text__. | I just love <strong>bold text</strong>. | I just love <b>bold text</b> . |
| Love**is**bold             | Love<strong>is</strong>bold             | Love <b>is</b> bold            |

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

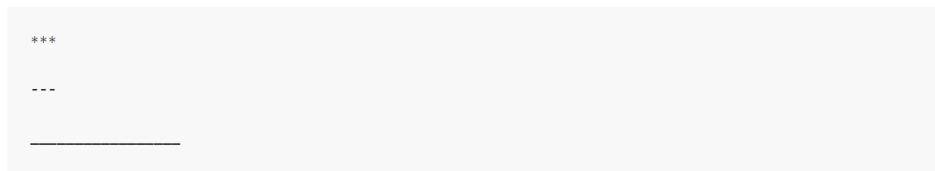
<https://www.markdownguide.org/basic-syntax/#emphasis>

# 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:

---

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

# 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/>)

# Markdown 실습

---

TYPORA를 활용한 문서 작성

# Typora

---

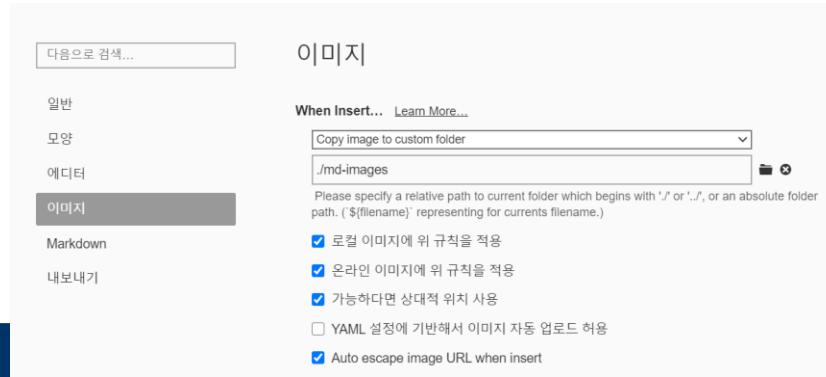
- 기존 텍스트 에디터(예- visual studio code), IDE 뿐만 아니라 마크다운 전용 에디터를 활용하여 문서를 작성할 수 있음
- Typora는 문법을 작성하면 바로 일반적으로 보이는 모습으로 변하여 처음 작성할 때 많은 도움을 주며, 표 같은 복잡한 문법이나 이미지를 드래그 앤 드랍으로 적용 가능함

<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/untile.png
  - 절대 경로 예시: C:/HPHK/Desktop/TIL/untile.png



# 마크다운 실습 1. 마크다운 문법 정리

---

- 지금까지 배운 마크다운 문법으로 마크다운 정리 문서를 만들고 제출하세요.
  - <https://www.markdownguide.org/cheat-sheet/>
  - GitHub Flavored Markdown (<https://github.github.com/gfm/>)
  - Mastering Markdown (<https://guides.github.com/features/mastering-markdown/>)
  - Markdown Guide (<https://www.markdownguide.org/>)
- 작성하고 채팅창에 공유
  - 미리 작성한 사람은 다른 사람 내용이랑 비교 해보세요 😊
  - 시트에 완료 체크 부탁드립니다.

# 마크다운 실습 2. TIL(Today I Learned)

- 어제 배운 내용을 마크다운으로 구조화하여 정리 해보세요.
  - 오늘 오후에 GitHub TIL (Today I Learned) 실습을 통해 제출 받습니다.

The screenshot shows a GitHub repository page for 'jbranchaud/til'. The repository has 437 stars, 9.7k forks, and 575 issues. The main page displays a list of 1,545 commits, each represented by a blue square icon and a title. The commits are organized into sections such as 'About', 'Releases', 'Packages', 'Contributors', and 'Languages'. The 'About' section includes a 'Today I Learned' badge with the URL 'writing.til/today-i-learned'. The 'Languages' section shows a single entry: 'Vim script 100.0%'. The commits themselves include titles like 'fix typo bug with ack list available file types command', 'Add Sign Up User With Email And Password as Amplify til', and 'Add Chrome Supports Many Unix Keyboard Shortcuts as a chrome til'.

| Commit Title   | Date          | Author |
|--|---------------|--------|
| fix typo bug with ack list available file types command                | 16 months ago | c7bea8 |
| Add Sign Up User With Email And Password as Amplify til                | 13 months ago |        |
| Add Chrome Supports Many Unix Keyboard Shortcuts as a chrome til       | 2 years ago   |        |
| Add List Functions For A Namespace as a clojure til                    | 5 months ago  |        |
| Add Define HSL Colors With Alpha Values as a CSS til                   | 5 months ago  |        |
| Add Allow Cross-Origin Requests To Include Cookies as a devops til     | 8 months ago  |        |
| Add Pipe Into A Case Statement as an elixir til                        | 2 years ago   |        |
| Add Add Javascript To Body Of The Document as a new Gatsby til         | 15 months ago |        |
| Add Push To A Branch On Another Remote as a git til                    | 3 months ago  |        |
| Add Reference An Encrypted Secret In An Action as a github actions til | 5 months ago  |        |
| Add Build For A Specific OS And Architecture as a go til               | 4 years ago   |        |
| Add Deploy A Review App To A Different Stack as a Heroku til           | 2 months ago  |        |
| Add Prevent Search Engines From Indexing A Page as an Html til         | 8 months ago  |        |
| Add What Counts As Cross-Origin With CORS? as an HTTP til              | 7 months ago  |        |
| Add Figure Out Your Public IP Address as an internet til               | 3 years ago   |        |
| Add Get The Response Status From An Axios Error as a javascript til    | 2 months ago  |        |
| Add Extract A List Of Values as a jq til                               | 8 months ago  |        |
| Add Set The Title Of A Window as a kitty til                           | 3 months ago  |        |
| Add Show Current System Time And Settings as a linux til               | 2 years ago   |        |
| Add Convert An HEIC Image File To JPG as a Mac til                     | 6 months ago  |        |
| Add Dump A Remote Database as a mongo til                              | 14 months ago |        |
| Add Doing Date Math as a mysql til                                     | 3 years ago   |        |

# 버전관리

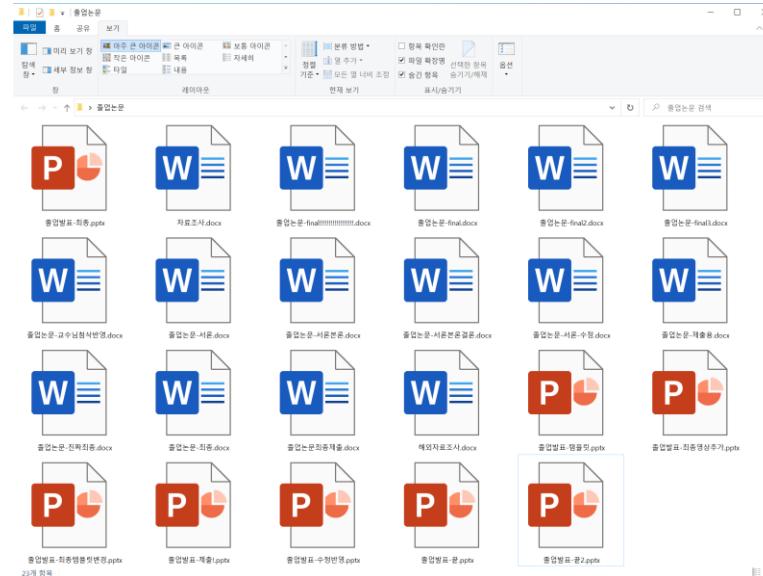
---

버전 관리가 무엇일까

# 버전관리

---

- 일반적인 우리의 버전관리 방식



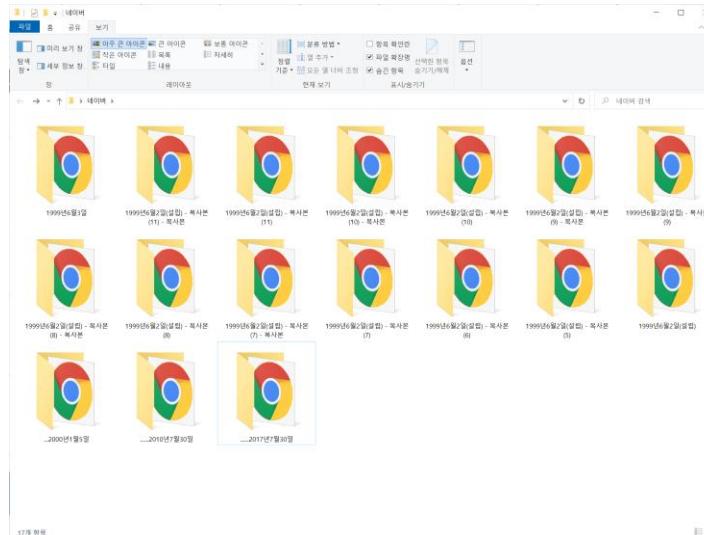
**파일을 버전별로**

**저장하여 관리!**

**네이버 소스코드는 어떨까?**

# 버전관리

- 1999년에 설립된 네이버의 소스코드 버전 관리(?)



# 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>.

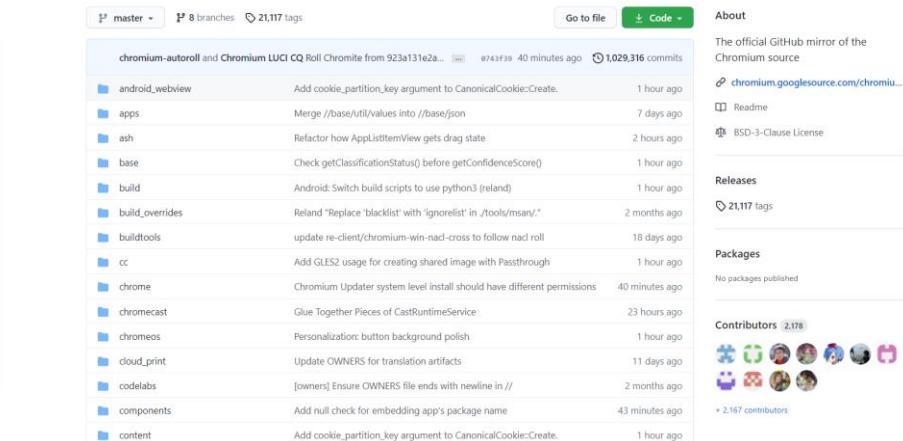
To check out the source code locally, don't use `git clone`! Instead, follow the [instructions](#) on how to get the code.

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.

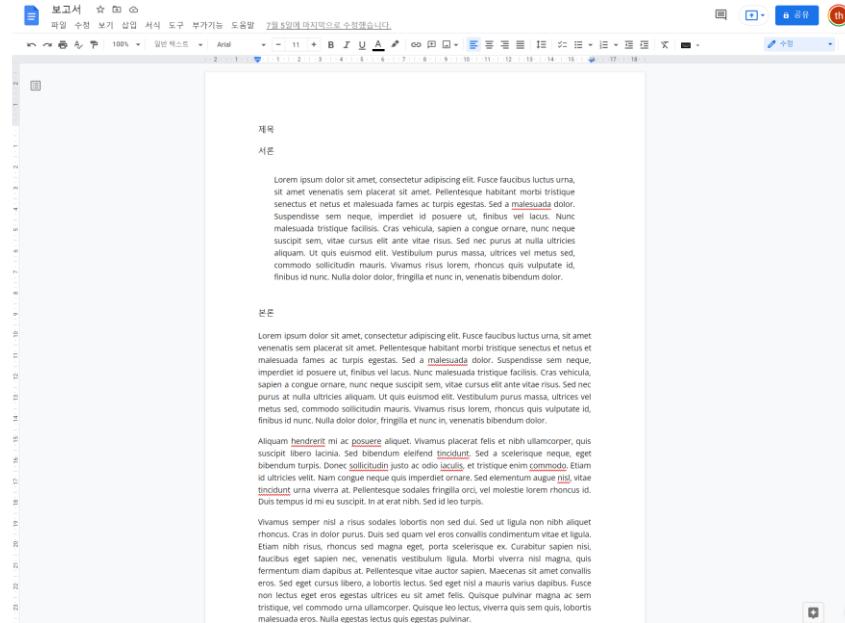
If you found a bug, please file it at <https://crbug.com/new>.



The screenshot shows the official GitHub mirror of the Chromium source code. It displays the master branch, 8 branches, 21,117 tags, and 1,029,316 commits. The commit list includes entries for android\_webview, apps, ash, base, build, build\_overrides, buildtools, cc, chrome, chromecast, chromeos, cloud\_print, codelabs, components, and content. The page also features sections for About, Releases, Packages, and Contributors, along with a sidebar for the official Google Source repository.

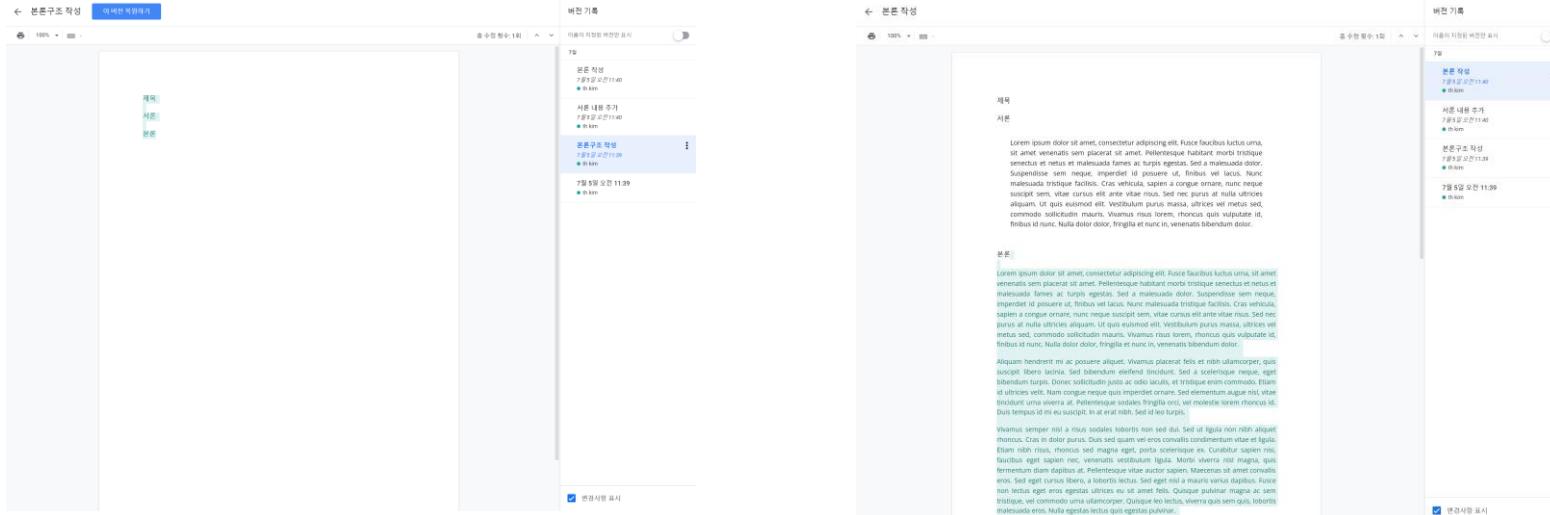
# 버전관리

- Google Document 버전 관리



# 버전관리

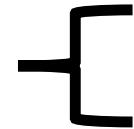
- Google Document 버전 관리
  - 문서는 하나지만 버전이 기록되어 있으면, 이전 시점을 조회하거나 복원시킬 수도 있음



# 버전관리

---

이 자료 간에 뭐가 바뀌었는지  
차이(diff)를 알 수 없다.

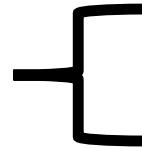


- 졸업논문\_1차.hwp
- 졸업논문\_수정.hwp
- 졸업논문\_수정\_2.hwp
- 졸업논문\_최종본.hwp
- 졸업논문\_진짜최종.hwp
- 졸업논문\_진짜최종이다.hwp
- 졸업논문\_레알진짜킹갓최종.hwp

# 버전관리

---

차이(diff) 가 무엇이고  
수정 이유를 log로 남길 수 있다.



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

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



# 버전관리시스템

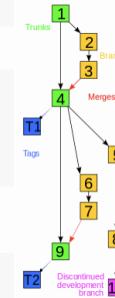
---

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

# 버전관리시스템

| 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) · Endevor (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 |   |      |



# 기초 CLI

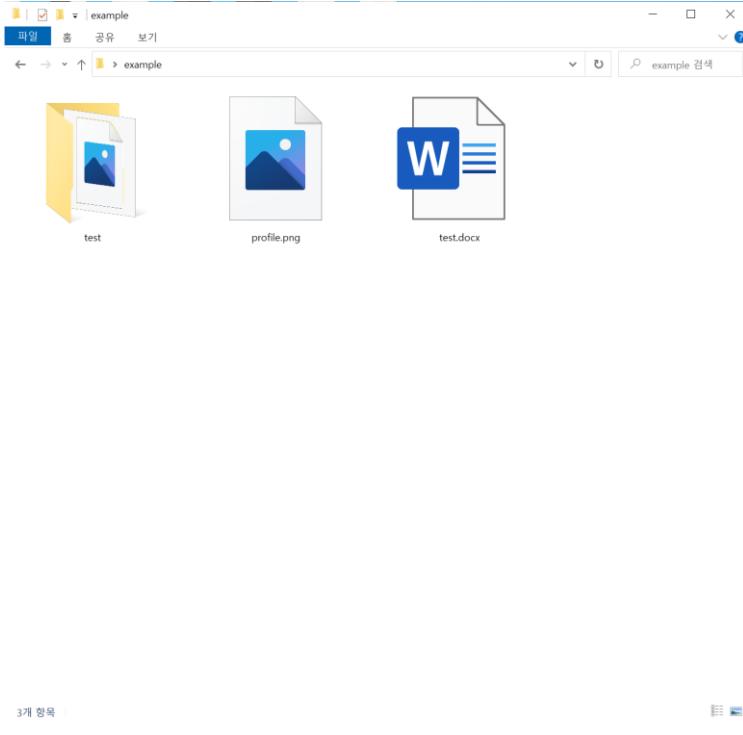
---

# 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
$ ls
profile.png  test/  test.docx

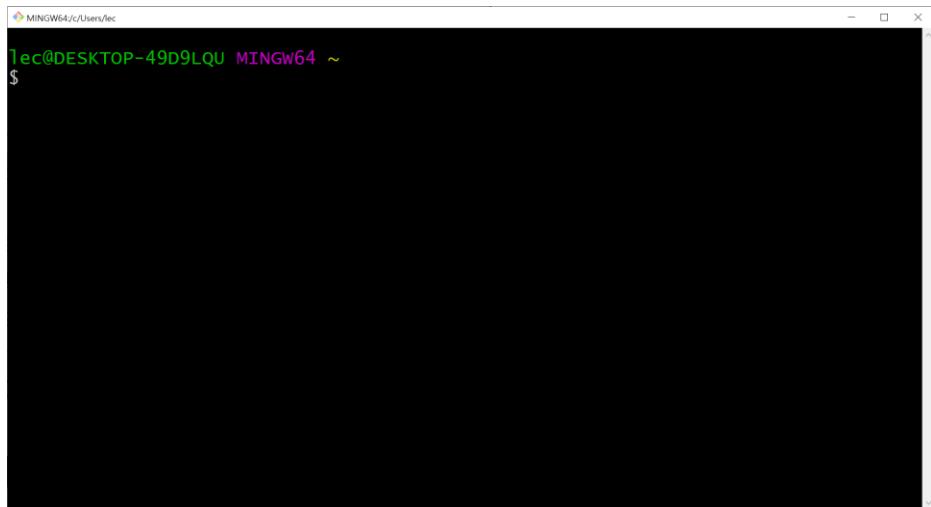
$
```

A screenshot of a terminal window titled 'MINGW64/c/Users/lec/Desktop/example'. The user has run the command '\$ ls', which lists the contents of the directory: 'profile.png', 'test/' (a folder), and 'test.docx'. The prompt '\$' appears again at the bottom.

# CLI (Command Line Interface)

---

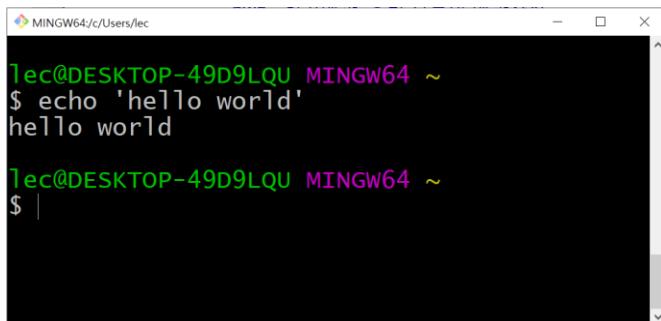
- 프롬프트 기본 인터페이스
  - 컴퓨터 정보
  - 디렉토리
  - \$



# CLI (Command Line Interface)

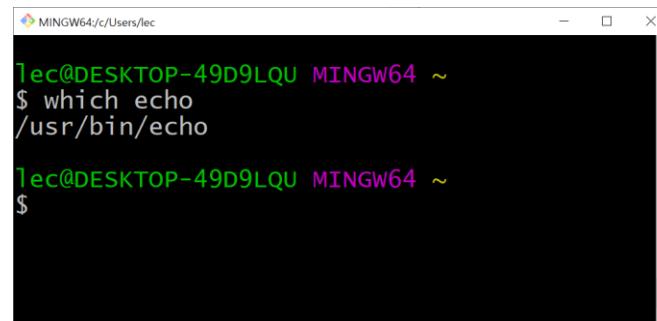
---

- 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 ~
$ |
```



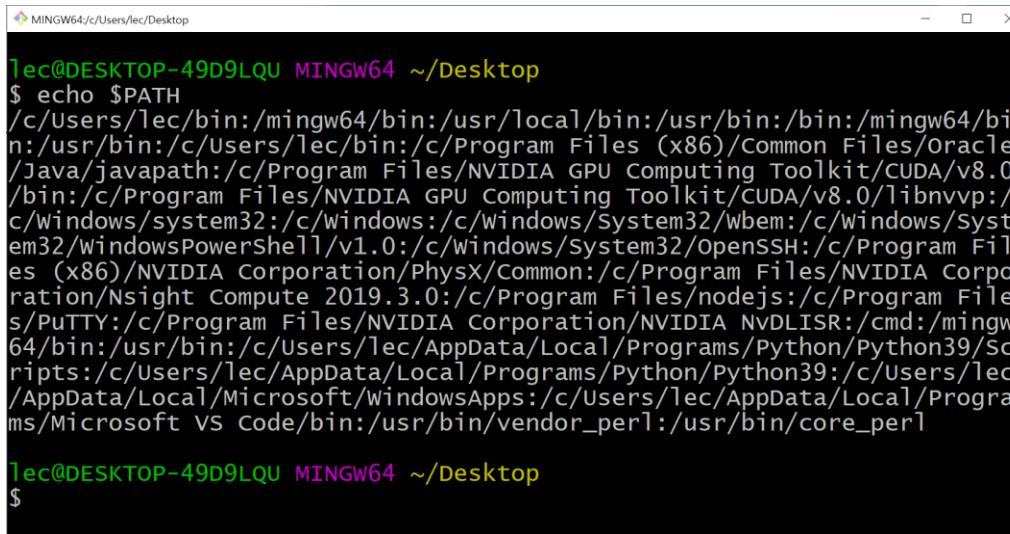
```
MINGW64:/c/Users/lec
lec@DESKTOP-49D9LQU MINGW64 ~
$ which echo
/usr/bin/echo

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

# \$PATH

---

- 셀에서 명령을 실행하면, 디렉토리를 관리하는 \$PATH 환경 변수에 접근하여 :로 분리된 디렉토리 목록을 검색하여 해당하는 프로그램을 실행



```
MINGW64:c/Users/lec/Desktop
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop
$ echo $PATH
/c/Users/lec/bin:/mingw64/bin:/usr/local/bin:/usr/bin:/bin:/mingw64/bin:/usr/bin:/c/Users/lec/bin:/c/Program Files (x86)/Common Files/oracle/Java/javapath:/c/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v8.0/bin:/c/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v8.0/libnvvp:/c/windows/system32:/c/windows:/c/windows/System32/wbem:/c/windows/System32/windowsPowerShell/v1.0:/c/windows/System32/OpenSSH:/c/Program Files (x86)/NVIDIA Corporation/PhysX/Common:/c/Program Files/NVIDIA Corporation/Nsight Compute 2019.3.0:/c/Program Files/nodejs:/c/Program Files/PuTTY:/c/Program Files/NVIDIA Corporation/NVIDIA NvDLISR:/cmd:/mingw64/bin:/usr/bin:/c/Users/lec/AppData/Local/Programs/Python/Python39/Scripts:/c/Users/lec/AppData/Local/Programs/Python/Python39:/c/users/lec/AppData/Local/Microsoft/WindowsApps:/c/Users/lec/AppData/Local/Programs/Microsoft VS Code/bin:/usr/bin/vendor_perl:/usr/bin/core_perl

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

# 디렉토리 관리

---

- pwd (print working directory) : 현재 디렉토리 출력
- cd (change directory) : 디렉토리 이동
  - . : 현재 디렉토리, .. : 상위 디렉토리
- ls (list) : 목록
- mkdir (make directory) : 디렉토리 생성
- touch : 파일의 날짜와 시간을 수정(0바이트 빈파일 생성)

# Git 기초 흐름

---

# Git

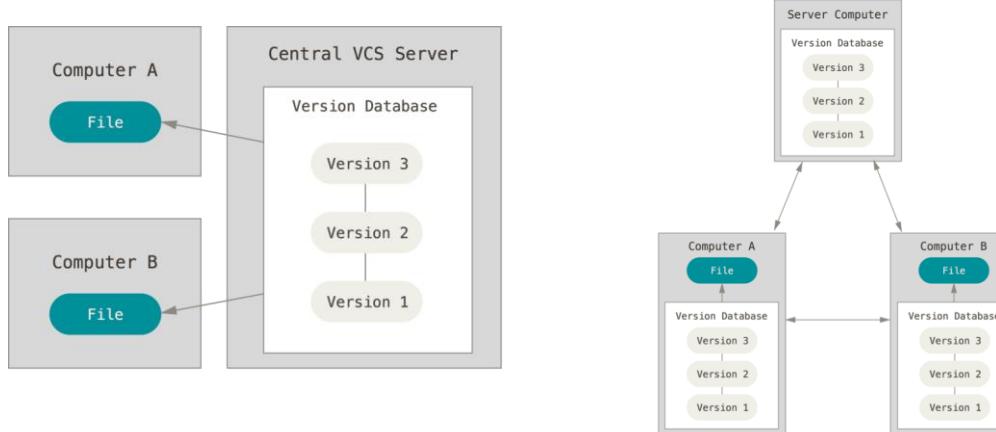
---

- 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



# 분산버전관리시스템(DVCS)

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



# 기본 흐름

---

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



작업(수정)한 파일

Working directory

커밋할 목록

INDEX  
(staging area)

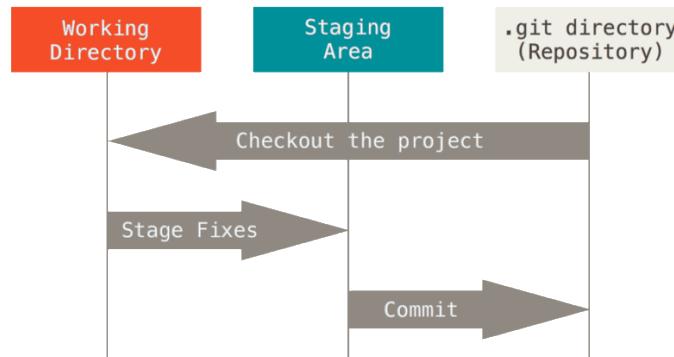
버전

HEAD

# 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>

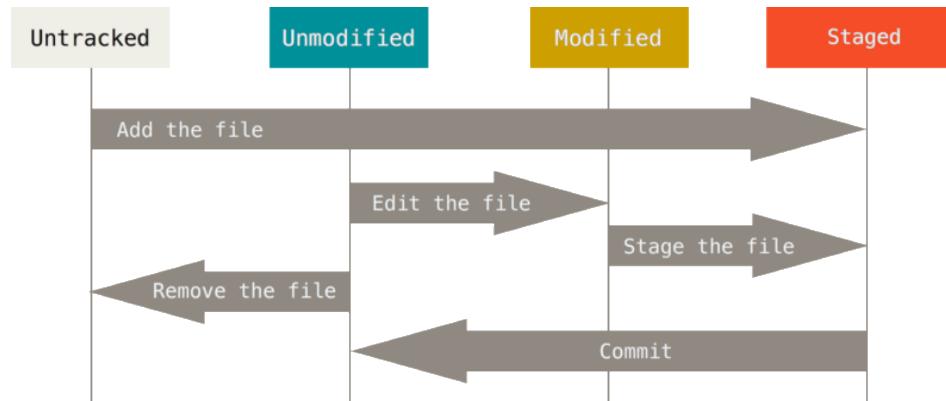
# 파일 라이프사이클

---

- Working directory의 모든 파일은 특정 상태를 가지며, git 명령어를 통해 변경
  - 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

---



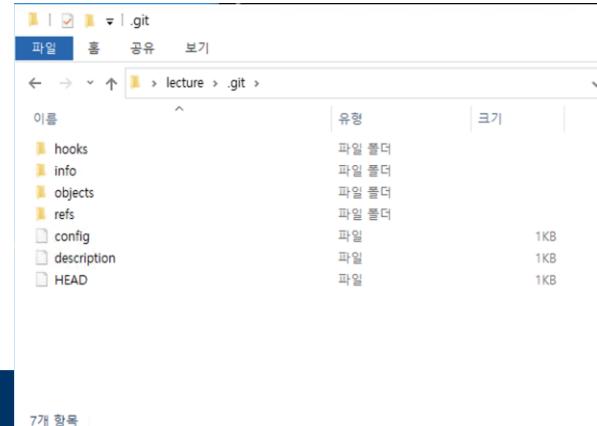
<https://git-scm.com/book/ko/v2/Git의-기초-수정하고-저장소에-저장하기>

# 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/
```



# 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

---

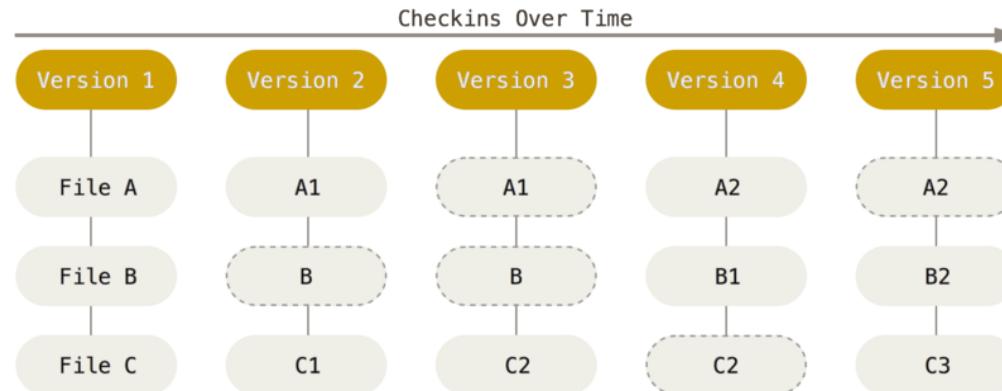
```
$ git commit -m '〈커밋메시지〉'
```

- 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

- 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



# 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)
$ |
```

# 기본 명령어 – status

\$ git log

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

\$ git log -1

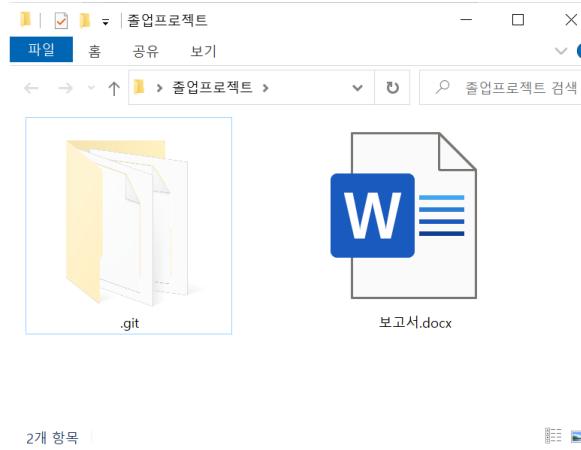
\$ git log --oneline

\$ git log -2 --oneline

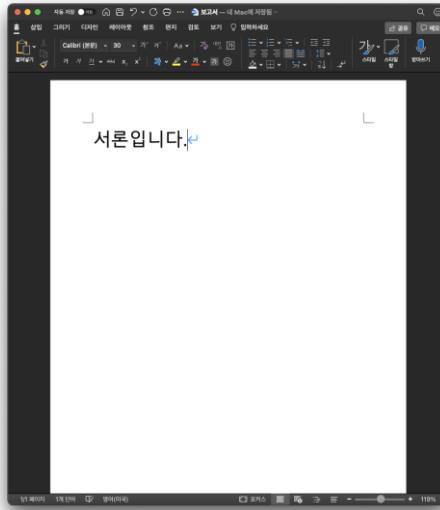
```
MINGW64:/c/Users/takhe/Desktop/chromium
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/chromium (master)
$ git log -5 --oneline
0743f396477c (HEAD -> master, origin/master, origin/main) Merge from 923a131e2a89 to 349dde1f1c62 (3 revisions)
80dae1472404 Chromium Updater system level install shou-
ions
8f27a5b8199d [ios] Notify SyncedSessionsObserver of ch-
7dc9ecc09d3 Add null check for embedding app's package
df0d55429722 ChromeVox reads the opening of tab strip

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

# 버전 관리 시나리오

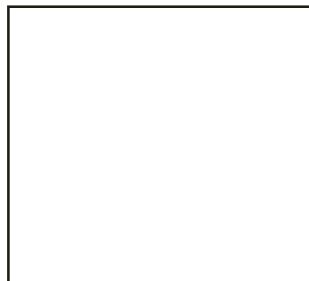


# 서론을 작성하고,

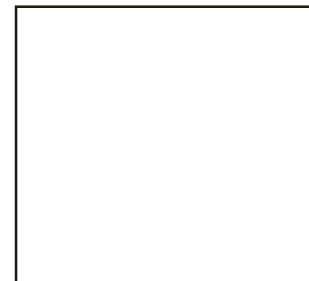




Working directory

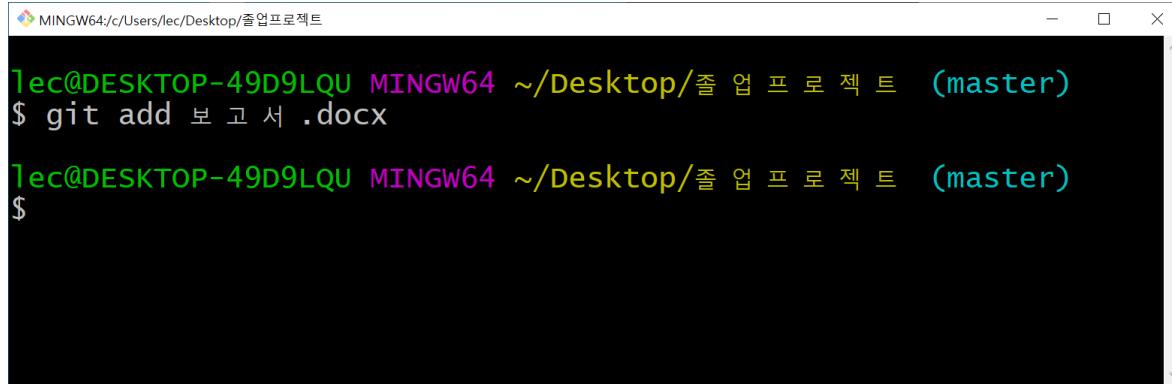


Staging area



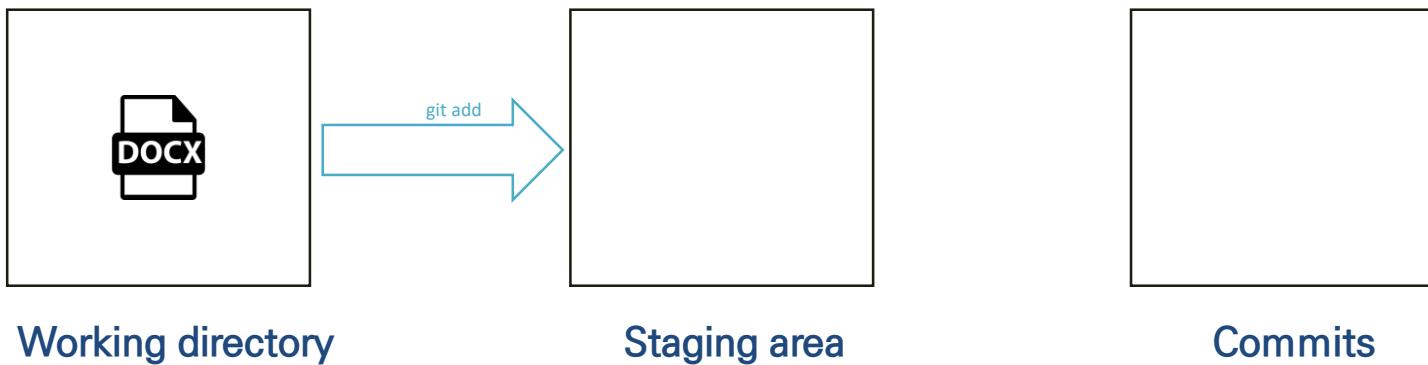
Commits

보고서.docx 변경사항 추적

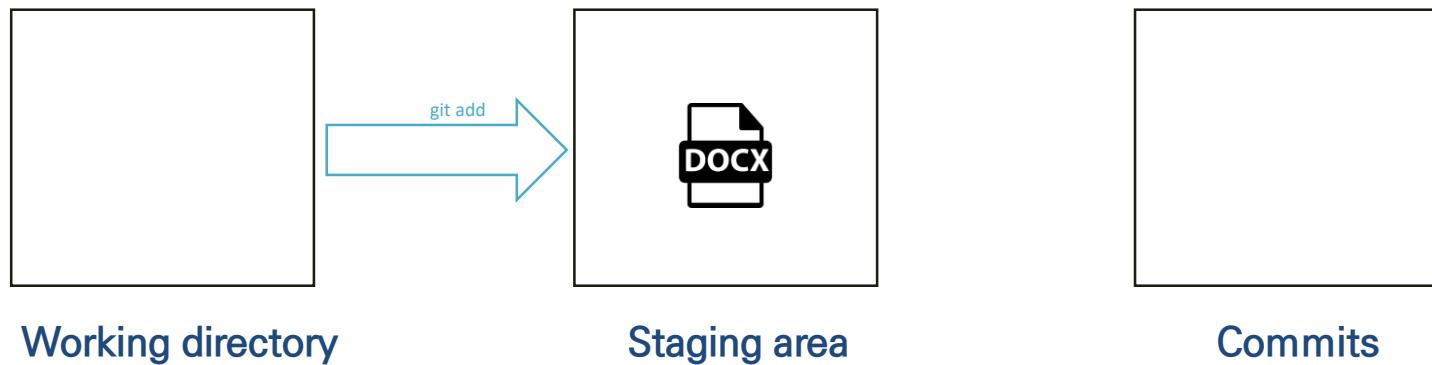


A screenshot of a terminal window titled "MINGW64:c/Users/lec/Desktop/졸업프로젝트". The window shows a command-line interface with the following text:

```
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop/졸업프로젝트 (master)
$ git add 보고서.docx
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop/졸업프로젝트 (master)
$
```



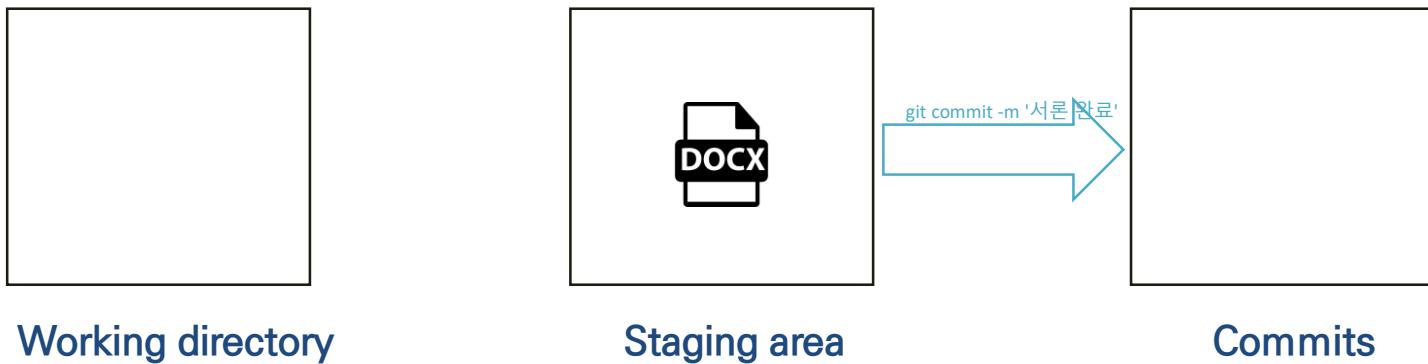
\$ git add 보고서.docx



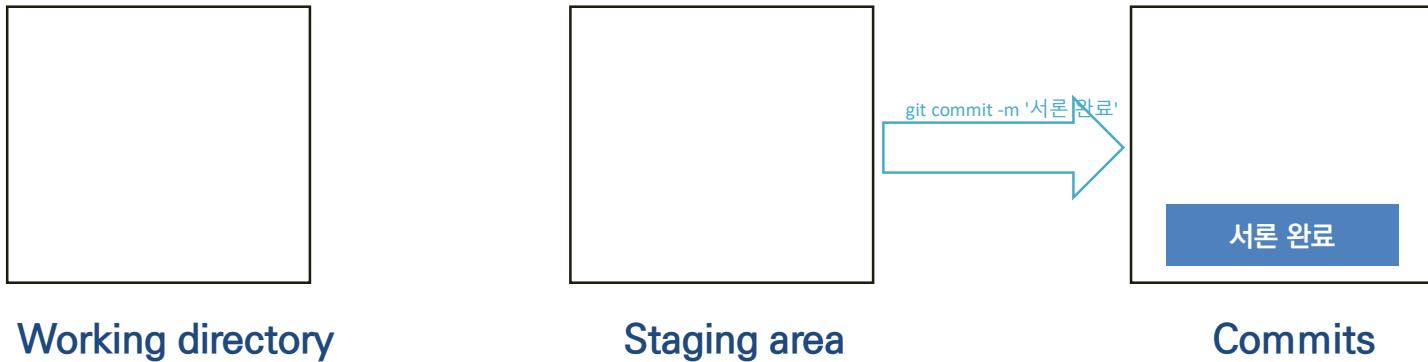
변경사항 목록에 쌓기

```
MINGW64:/c/Users/lec/Desktop/졸업프로젝트
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop/졸업프로젝트 (master)
$ git commit -m '서론 완료'
[master 45a5220] 서론 완료
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 adfsadf.txt

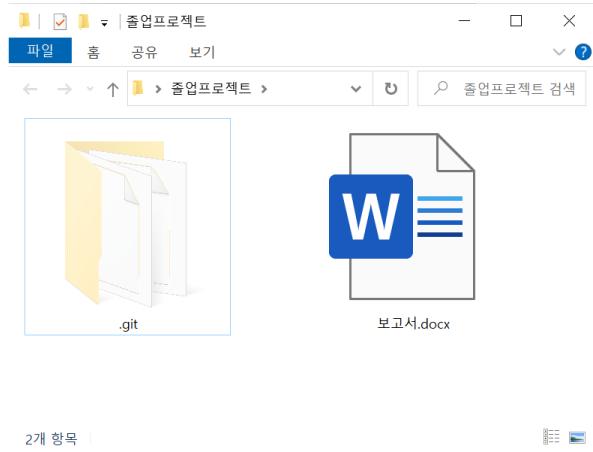
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop/졸업프로젝트 (master)
$ |
```



**\$ git commit -m '메시지'**

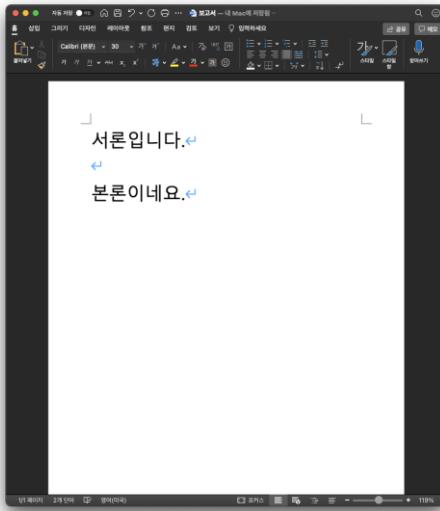


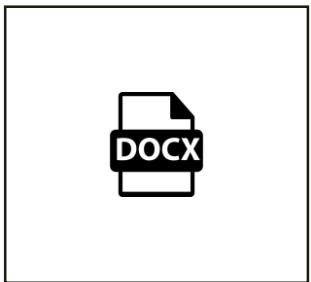
## 버전 기록



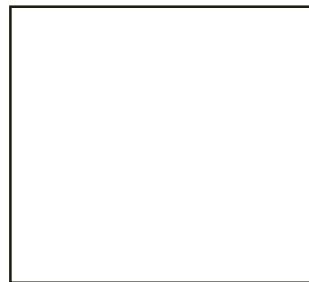
# 현재 폴더 상태

# 본론 작성 후에,





Working directory

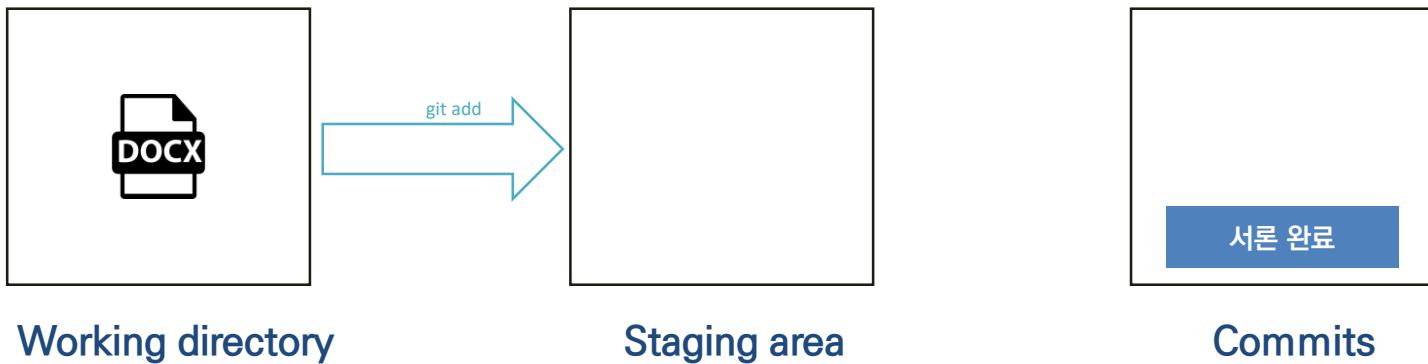


Staging area

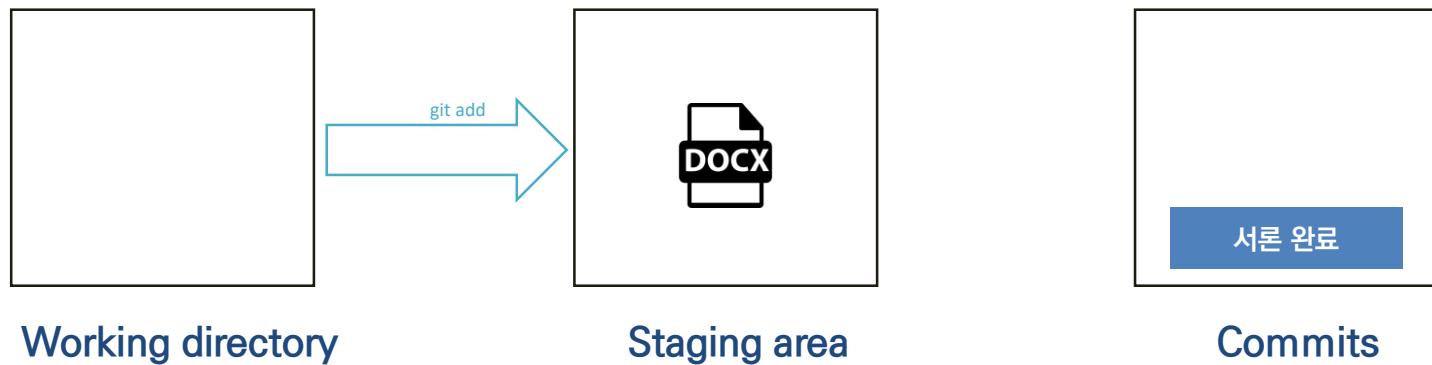


Commits

보고서.docx 변경사항 추적



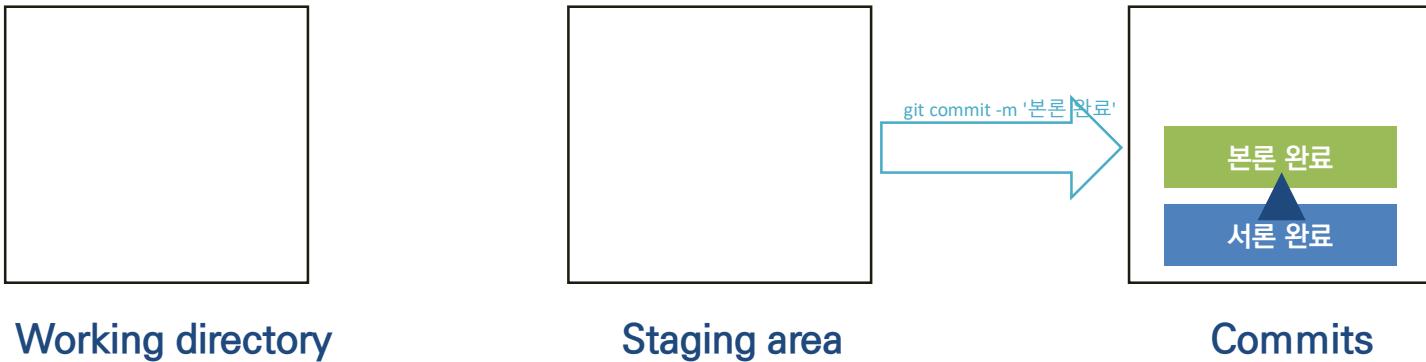
\$ git add 보고서.docx



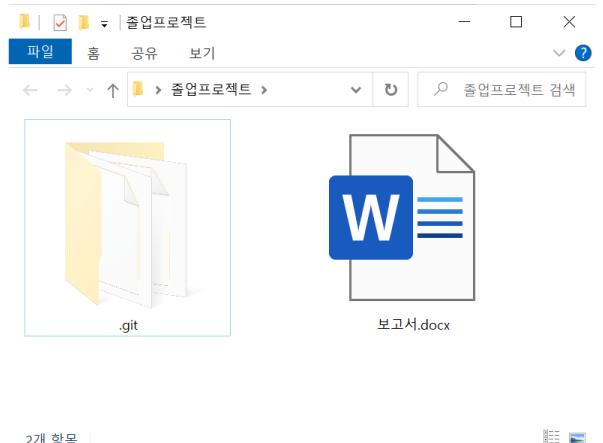
변경사항 목록에 쌓기



**\$ git commit -m '메시지'**



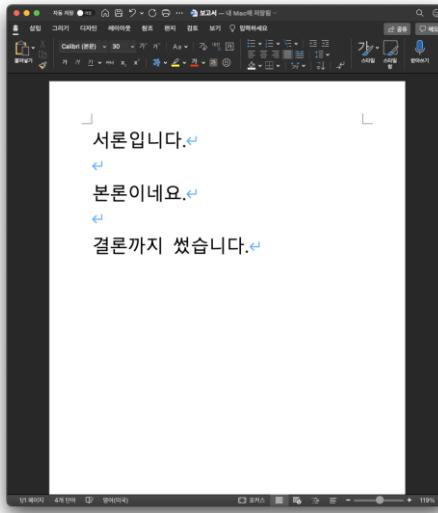
## 버전 기록

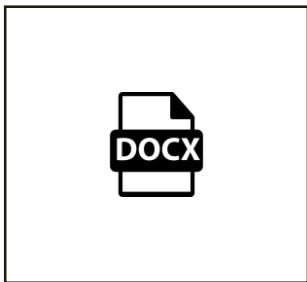


# 현재 폴더 상태

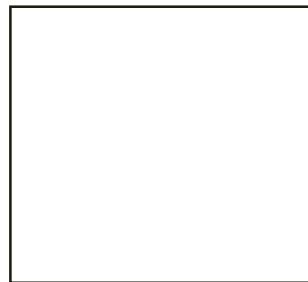
---

# 결론 작성 후에,

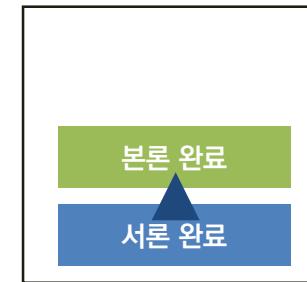




Working directory

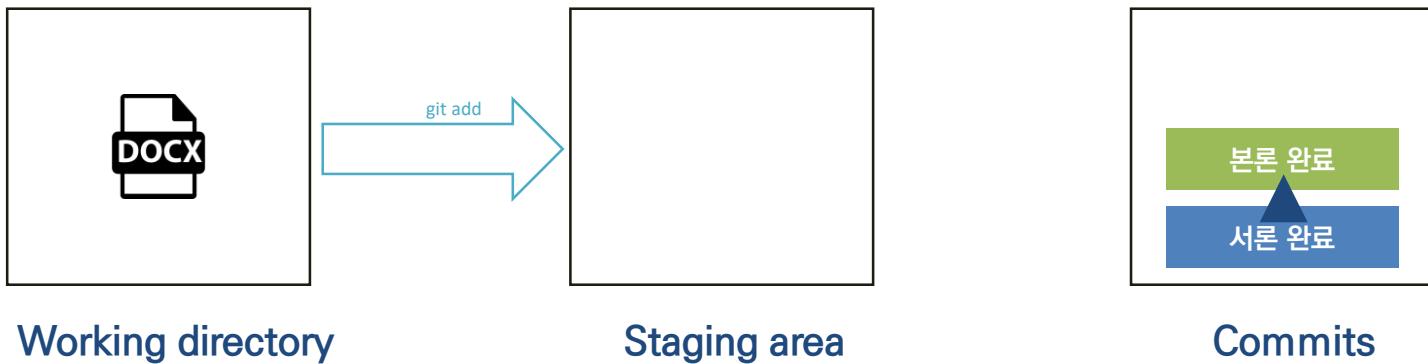


Staging area

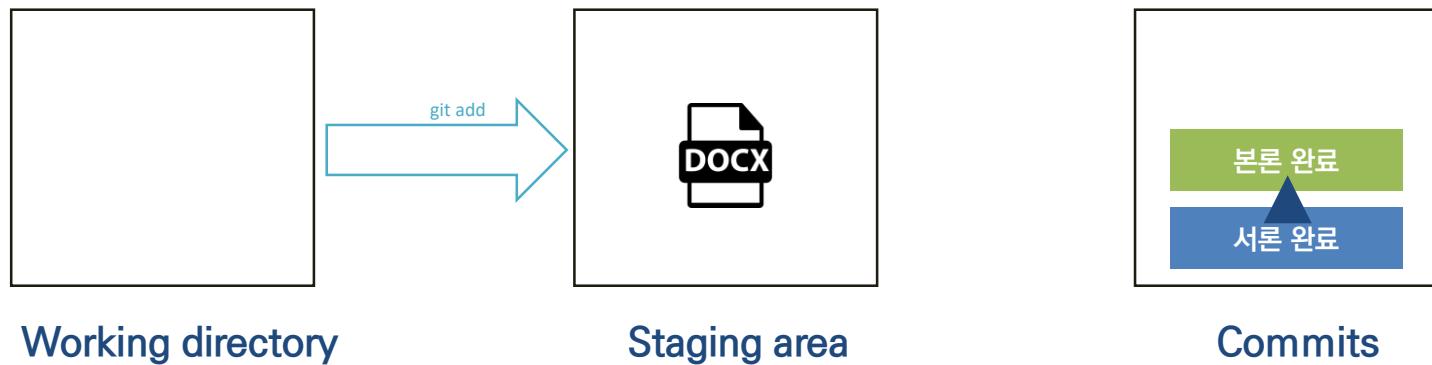


Commits

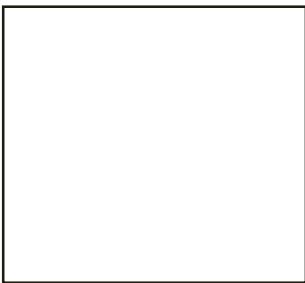
보고서.docx 변경사항 추적



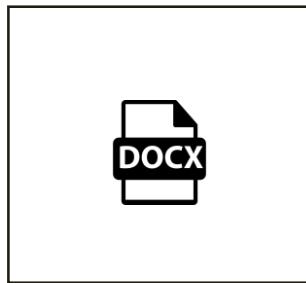
\$ git add 보고서.docx



변경사항 목록에 쌓기

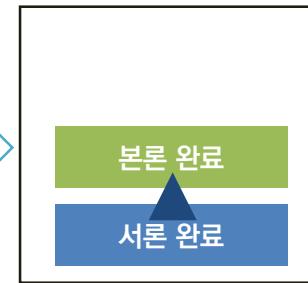


Working directory



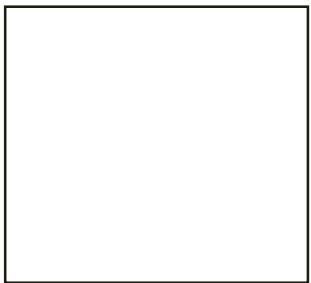
Staging area

git commit -m '끝!!!'

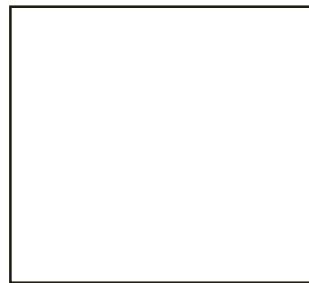


Commits

\$ git commit -m '메시지'



Working directory



Staging area

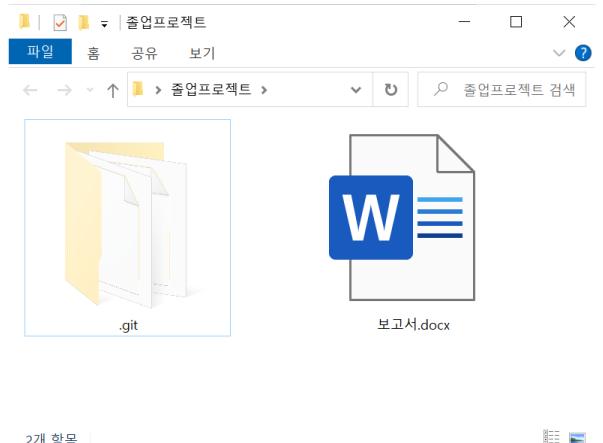
git commit -m '끝!!!'

A light blue arrow points from the Working directory box to the Staging area box, indicating the flow of changes.



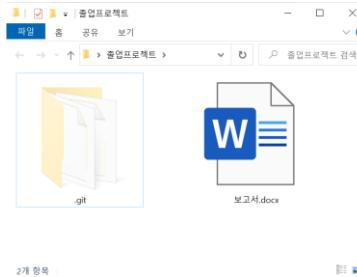
Commits

## 버전 기록

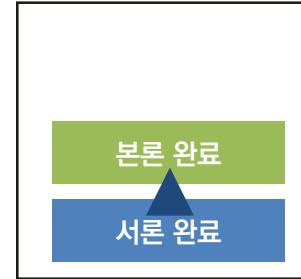
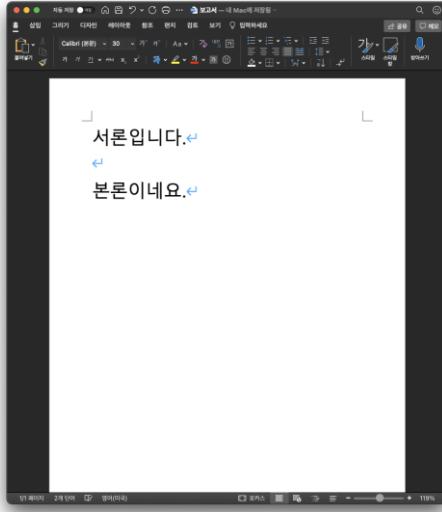


# 현재 폴더 상태

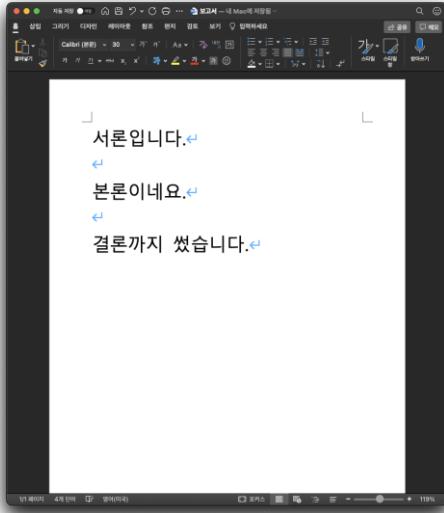
# 파일을 복제하는 것이 아닌 버전을 기록하는 것



본론 완료 버전으로 돌아가서 파일을 열면,



다시 끝!!! 버전으로 돌아가서 파일을 열면,



---

**커밋으로 버전별로  
스냅샷처럼 기록**

---

**add를 사용하는 이유?  
조모임 작업 중 버전 관리를 한다면,**

## Working directory

## Staging area

## Commits



자료조사  
\_석영.hwp



보고서.hwp      발표자료.pptx



발표자료\_표  
지.png



발표자료\_배  
경.png

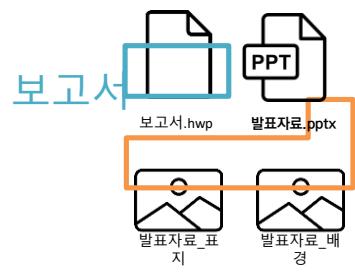
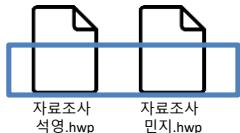
버전을 어떻게 나눠 볼 수 있을까?

## Working directory

## Staging area

## Commits

자료조사



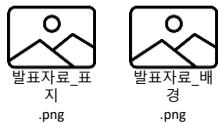
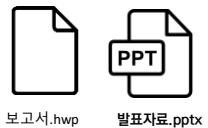
발표자료

버전을 어떻게 나눠 볼 수 있을까?

---

# 자료조사 버전 만들기

## Working directory

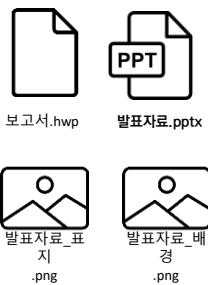


## Staging area

## Commits

1) 자료 조사 파일 add

## Working directory



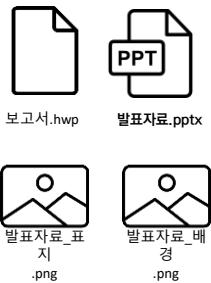
## Staging area



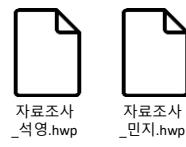
## Commits

1) 자료 조사 파일 add 결과

## Working directory



## Staging area



## Commits

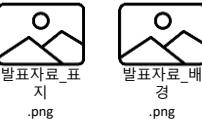
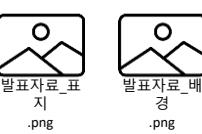
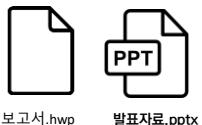


## 2) 버전 기록(커밋)

Working directory

Staging area

Commits



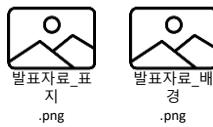
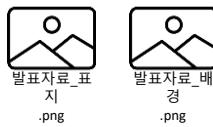
자료 조사  
(ef123a1)

2) 버전 기록(커밋) 결과

---

# 보고서 버전 만들기

## Working directory



## Staging area



git add

## Commits

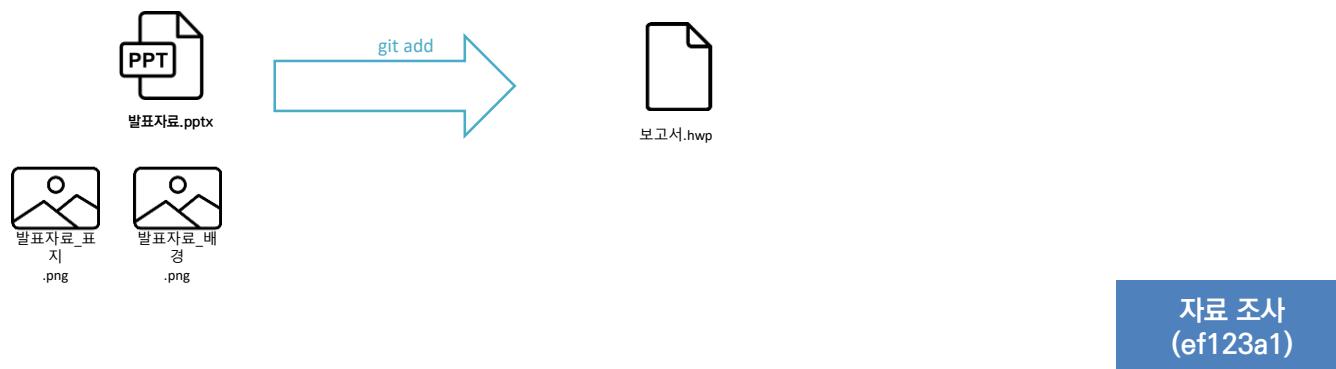
자료 조사  
(ef123a1)

1) 보고서 파일 add

Working directory

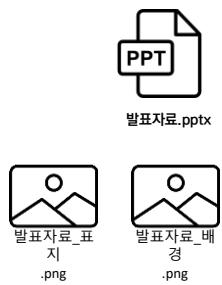
Staging area

Commits



1) 보고서 파일 add 결과

## Working directory



## Staging area



`git commit -m '보고서'`

## Commits

자료 조사  
(ef123a1)

## 2) 버전 기록(커밋)

Working directory

Staging area

Commits



## 2) 버전 기록(커밋) 결과

---

# 발표자료 버전 만들기

Working directory

Staging area

Commits

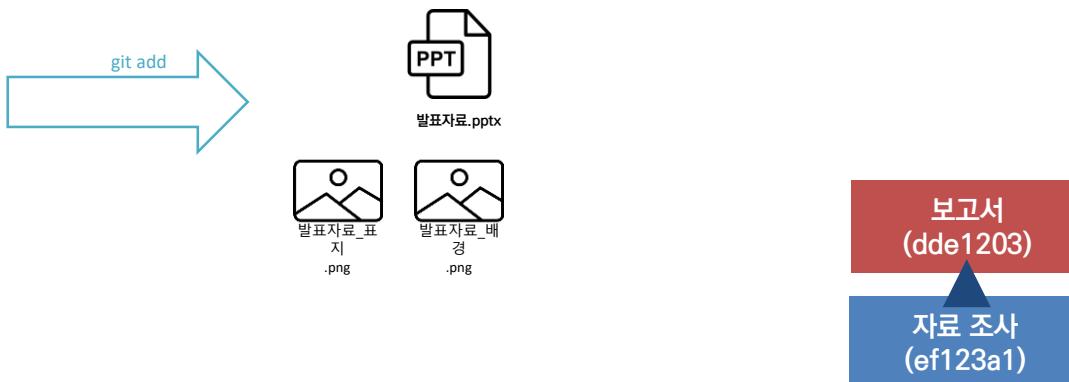


# 1) 발표자료 파일 add

Working directory

Staging area

Commits

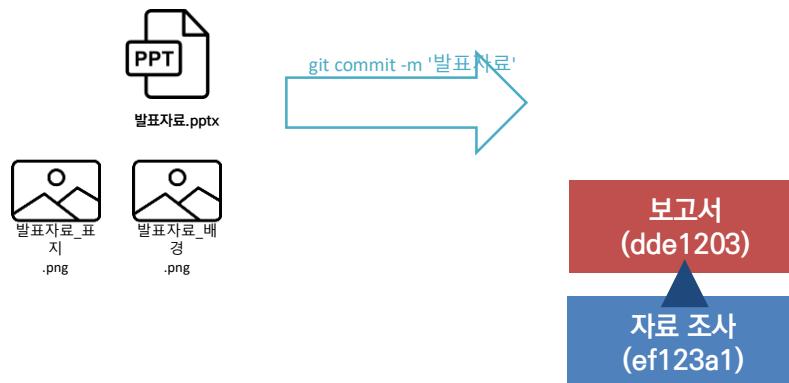


1) 발표자료 파일 add 결과

Working directory

Staging area

Commits

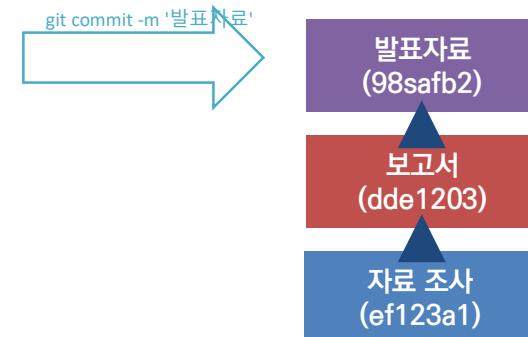


## 2) 버전 기록(커밋)

Working directory

Staging area

Commits



## 2) 버전 기록(커밋) 결과

---

staging area를 통해  
원하는 파일으로만 버전을 나눠서 기록

---

PPT 자료를 수정한 뒤..



발표자료.pptx

## Working directory

## Staging area

## Commits

modified



발표자료.pptx



발표 자료 수정 이력 추적

Working directory

Staging area

Commits

modified



발표자료.pptx

git add



발표자료  
(98safb2)

보고서  
(dde1203)

자료 조사  
(ef123a1)

1) 수정 내역 add

Working directory

Staging area

Commits



1) 수정 내역 add 결과

Working directory

Staging area

Commits

modified



발표자료.pptx



발표자료  
(98safb2)

보고서  
(dde1203)

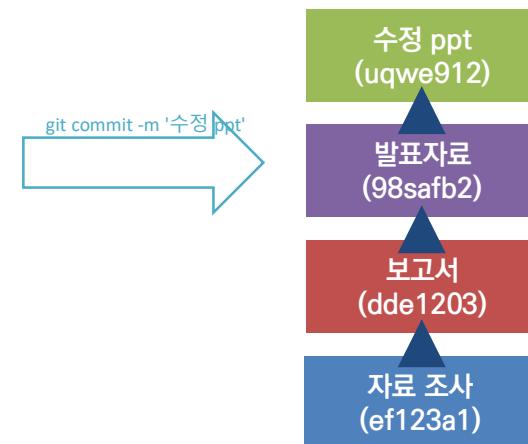
자료 조사  
(ef123a1)

## 2) 버전 기록(커밋)

Working directory

Staging area

Commits



## 2) 버전 기록(커밋) 결과

# Git 활용하기

---

- Git bash 설치

<https://gitforwindows.org/>



# Git 설정 파일 (config)

---

- 사용자 정보 (commit author) : 커밋을 하기 위해 반드시 필요
  - git config —global user.name “*username*”
    - *Github*에서 설정한 *username*으로 설정
  - git config —global user.email “*my@email.com*”
    - *Github*에서 설정한 *email*로 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 (15분)

---

- 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 당연하게도 변경사항이 없어 버전이 기록되지 않습니다.

# (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 (40분)

---

- 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

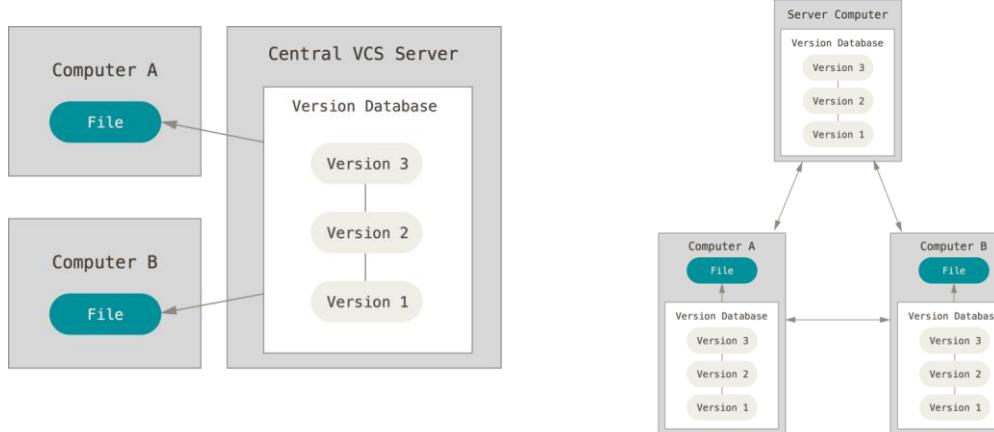
# 원격저장소 활용하기

---

TIL

# 분산버전관리시스템(DVCS)

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



# 원격저장소 설정 명령어

---

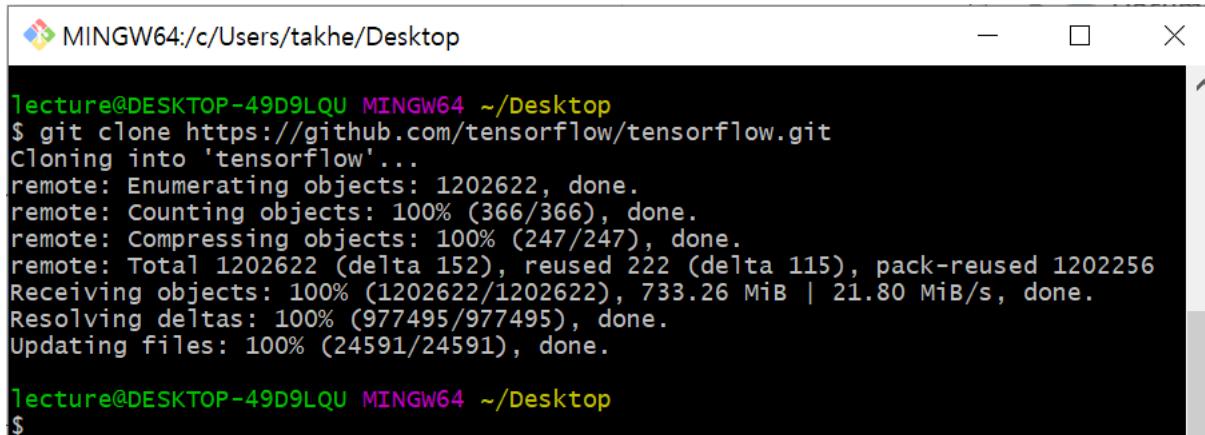
- remote 확인  
  \$ git remote -v
- remote 추가  
  \$ git remote add <이름> <url>
- remote 삭제  
  \$ git remote rm <이름>
- remote url 변경  
  \$ git remote set-url <이름> <url>
- remote 이름 변경  
  \$ git remote rename <이름> <새이름>

# 원격저장소 활용 명령어

---

**\$ git clone <원격저장소주소>**

- 원격 저장소를 복제하여 가져옴



The screenshot shows a terminal window titled 'MINGW64:/c/Users/takhe/Desktop'. The command \$ git clone https://github.com/tensorflow/tensorflow.git is run, followed by its execution output. The output details the cloning process, including object enumeration, counting, compressing, receiving objects, resolving deltas, and updating files.

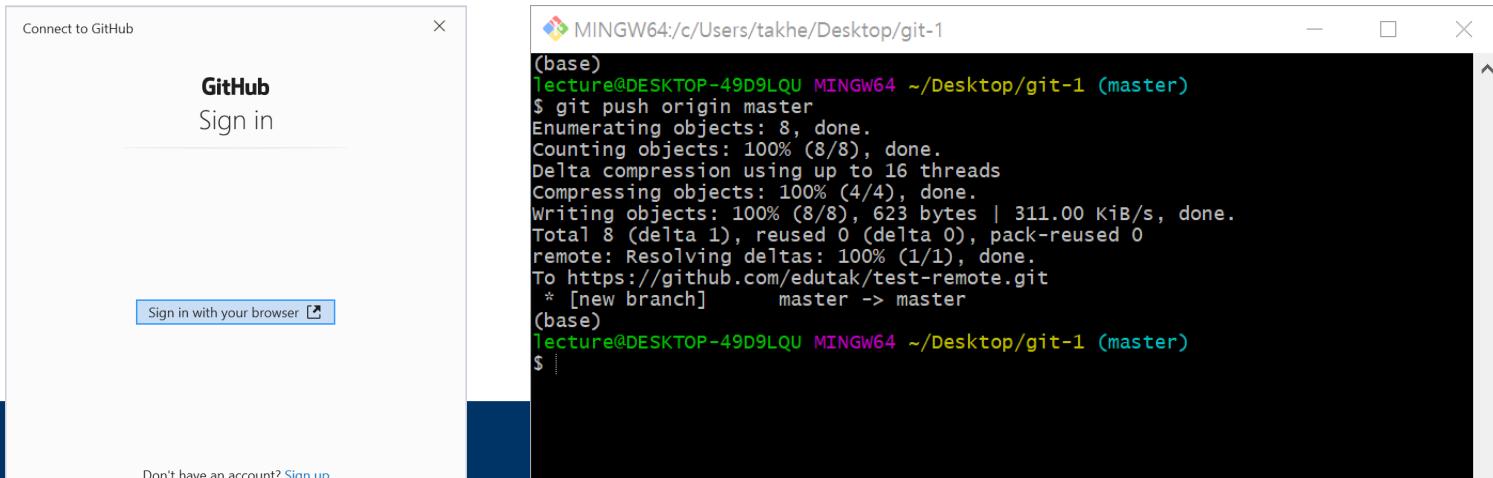
```
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop
$ git clone https://github.com/tensorflow/tensorflow.git
Cloning into 'tensorflow'...
remote: Enumerating objects: 1202622, done.
remote: Counting objects: 100% (366/366), done.
remote: Compressing objects: 100% (247/247), done.
remote: Total 1202622 (delta 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 활용 명령어

\$ git push <원격저장소이름> <브랜치이름>

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



The image shows two windows side-by-side. On the left is a 'Connect to GitHub' dialog box from GitHub's website, prompting for a 'Sign in'. On the right is a terminal window titled 'MINGW64:/c/Users/takhe/Desktop/git-1' showing the output of a 'git push' command. The terminal output is as follows:

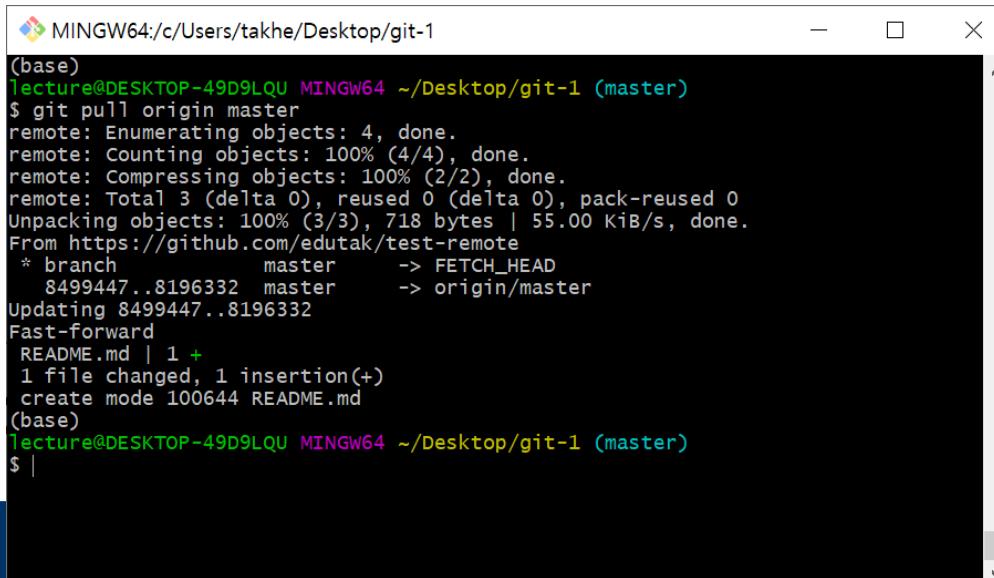
```
(base)
Tecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git push origin master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (8/8), 623 bytes | 311.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/edutak/test-remote.git
 * [new branch]      master -> master
(base)
Tecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ |
```

# 원격저장소 활용 명령어

---

**\$ 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)
$ |
```

# GitHub 실습

---

# GitHub 실습 1. 원격저장소에 공유하기

---

- 1) git 실습 1에서 만든 test 저장소를 준비
  - 없는 경우 test 폴더를 만들고 커밋을 발생 시켜주세요.
- 2) GitHub에서 원격저장소 test를 만들고
- 3) 로컬 저장소에서 원격저장소 설정을 하고, push 하기

# GitHub 실습 2. TIL (Today I Learned)

---

- 1) TIL 폴더를 만들고, 저장소로 설정
- 2) README.md 파일 만들고, add, commit
- 3) 원격 저장소 만들고,
- 4) push (remote 설정, git push origin master)
- 반복)
  - Markdown.md → 기존에 정리한 것 폴더로 이동하고 푸시
  - Git.md → 마크다운으로 정리하고 커밋하고 푸시
  - 지난 수업 내용 정리하고 커밋하고 푸시!

# GitHub 실습 3. 알고리즘 문제 풀이

---

- 1) 알고리즘 문제 풀이를 위한 폴더를 만들어봅시다.
  - 아래의 문제를 본인의 언어를 통해서 풀며 작성해봅시다.
  - <https://www.acmicpc.net/problem/2557>
  - <https://www.acmicpc.net/problem/1000>
  - <https://programmers.co.kr/learn/courses/30/lessons/12932>
- 2) README 등을 활용하여 원격저장소를 만들어 관리해봅시다.

# GitHub 실습 4. GitHub Profile README

---

- 1) 바탕화면에 ‘프로필’ 폴더를 만들어주세요.
- 2) README.md 파일을 만들고 자기 소개를 작성해주세요.
- 3) 원격저장소에 본인의 username으로 저장소를 만들어주세요
- 4) 해당 저장소에 push하고 본인의 프로필 URL로 들어가주세요.
  - <https://github.com/username>
- 5) 수정한 후 다시 push!

# GitHub 실습 5. clone 하기

---

- 1) **오픈소스로 공개된 원격 저장소를 복제(clone)**
  - git bash에서 경로를 주의해서 보세요.
  - 복제하면 원격 저장소 이름의 폴더가 생성됩니다.
- 2) 복제한 저장소의 log를 살펴보세요.

# GitHub 실습 6. 충돌 확인 및 해결하기

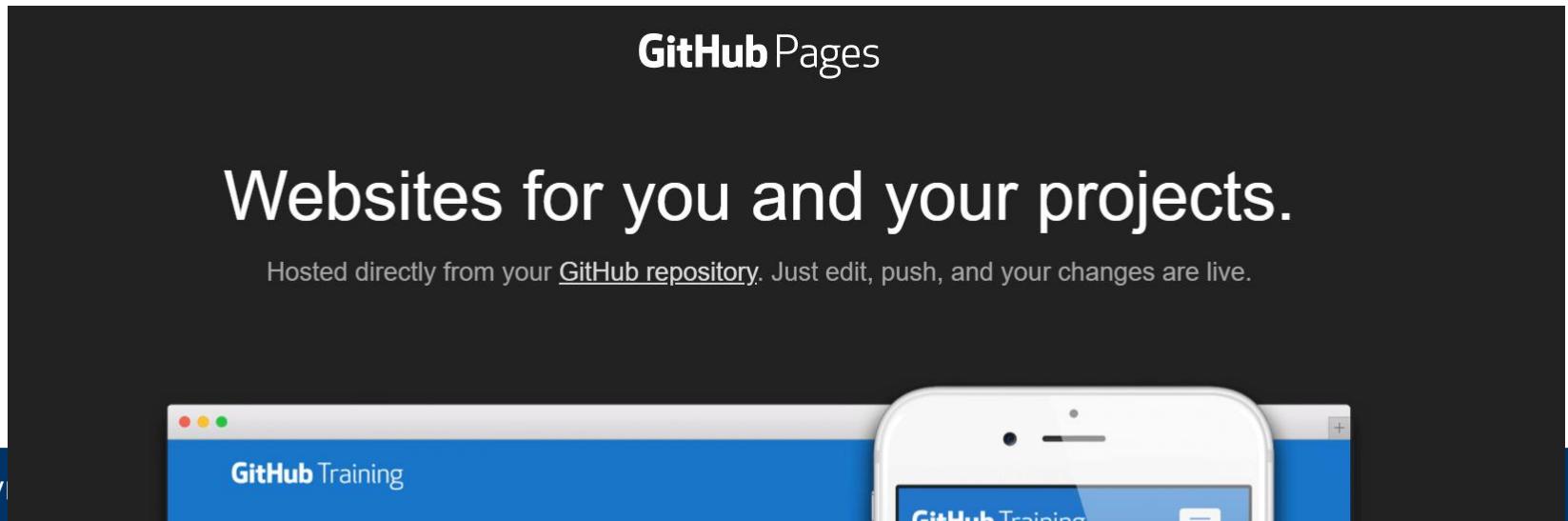
---

- 1) 본인의 프로필 URL로 들어가주세요.
  - <https://github.com/username>
- 2) GitHub에서 직접 수정 해보세요.
- 3) 로컬 저장소 ‘프로필’ 폴더에서 다르게 수정한 이후에 push 해보세요.
- 4) 오류 메시지를 확인하고, 이에 대한 해결 방법을 고민 해보세요.

# GitHub Pages

---

- GitHub에서는 원격 저장소를 호스팅하여 무료로 웹 사이트로 활용할 수 있도록 제공
  - <https://pages.github.com/>
- 개인 블로그, 회사 기술 블로그, 랜딩 페이지, 공식 문서 등 다양하게 활용



# GitHub 실습 7. GitHub Pages 실습

---

- 1) ‘포트폴리오’ 폴더를 만들고 index.html을 만들고 내용 작성하기
  - HTML 코드를 작성하지 않고 글만 작성해도 됩니다.
  - 선택) <https://startbootstrap.com> 에서 템플릿을 받아서 활용하기
- 2) 버전 남기기
- 3) 원격저장소 이름을 *username.github.io*로 만들기
  - *username*은 반드시 본인이 GitHub 가입할 때 작성한 것으로 설정해주세요.
- 4) 해당 원격저장소에 push하고 확인하기
  - <https://username.github.io>
- 5) 일부 내용을 수정하고 버전 남기고 push하기