Git & GitHub :: 기본 명령어

## Git 작업환경 시작하기 (see also: git help tutorial)

|  |  |  |
| --- | --- | --- |
| 명령어 | 설명 | 예제 코드 |
| clone | 새로운 디렉터리에 Repository를 복제한다. | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1 (master)  $ git clone https://github.com/vuejs/vue.git  Cloning into 'vue'...  remote: Enumerating objects: 61600, done.  remote: Total 61600 (delta 0), reused 0 (delta 0), pack-reused 61600  Receiving objects: 100% (61600/61600), 30.83 MiB | 19.61 MiB/s, done.  Resolving deltas: 100% (43107/43107), done.  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1 (master)  $ ls -al  total 16  drwxr-xr-x 1 jin 197121 0 8월 30 18:56 ./  drwxr-xr-x 1 jin 197121 0 8월 30 18:17 ../  drwxr-xr-x 1 jin 197121 0 8월 30 18:19 .git/  drwxr-xr-x 1 jin 197121 0 8월 30 18:56 vue/ |
| init | 새로운 빈 Repository를 생성하고, 기존 존재하던 것들을 초기화한다. | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub  $ mkdir test1  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub  $ cd test1  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1  $ git init  Initialized empty Git repository in C:/Users/jin/Documents/Github/test1/.git/  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1 (master)  $ ls -al  total 8  drwxr-xr-x 1 jin 197121 0 8월 30 18:19 ./  drwxr-xr-x 1 jin 197121 0 8월 30 18:17 ../  drwxr-xr-x 1 jin 197121 0 8월 30 18:19 .git/ |

## 현재 변경작업 (see also: git help everyday)

|  |  |  |
| --- | --- | --- |
| 명령어 | 설명 | 예제 코드 |
| add | 해당 파일을 Staged 상태로 변경한다. | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue/ryuyj (main)  $ git add test.txt  warning: LF will be replaced by CRLF in ryuyj/test.txt.  The file will have its original line endings in your working directory  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue/ryuyj (main)  $ git status  On branch main  Your branch is up to date with 'origin/main'.  Changes to be committed:  (use "git restore --staged <file>..." to unstage)  new file: test.txt |
| mv | 파일 이동 또는 파일 명 변경 또는 심볼릭 링크의 이름변경 (Move or rename a file, a directory, or a symlink) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue (main)  $ mv test.txt ./ryuyj/test.txt  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue (main)  $ cd ryuyj  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue/ryuyj (main)  $ ls -al  total 9  drwxr-xr-x 1 jin 197121 0 8월 30 20:32 ./  drwxr-xr-x 1 jin 197121 0 8월 30 20:32 ../  -rw-r--r-- 1 jin 197121 21 8월 30 20:30 test.txt |
| rm | 작업 트리(Working tree)에서 인덱스 파일을 제거(Remove files from the working tree and from the index) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git rm test.txt  rm 'test.txt' |
| mkdir | 현재 위치에서 새로은 디렉터리를 생성한다. | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue (main)  $ mkdir ryuyj |
| vim | vi 호환 텍스트 편집기 | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue (main)  $ vim test.txt  입력을 위해서는 영문 i를 입력 후 텍스트 입력  esc 누르고 :wq 입력후 엔터를 치면 저장된다. |
| cat | cat 명령을 사용하면, 명령 뒤에 파일명을 적는 것만으로 파일의 모든 내용을 표준 출력(standard output)에 출력할 수 있습니다. | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue (main)  $ cat test.txt  test file  test text |

## 작업이력과 현재 상태를 검사 (see also: git help revisions)

|  |  |  |
| --- | --- | --- |
| 명령어 | 설명 | 예제 코드 |
| log | Commit 로그를 보이게한다. (Show commit logs) | $ git log  commit d3a4ddf31f909776715ccd243e162df4af00b7f2 (HEAD -> main)  Author: Jin <ryuyj76@naver.com>  Date: Tue Aug 30 20:41:18 2022 +0900  test |
| show | 다양한 종류의 객체를 보이게 한다.(Show various types of objects) | $ git show  commit d3a4ddf31f909776715ccd243e162df4af00b7f2 (HEAD -> main)  Author: Jin <ryuyj76@naver.com>  Date: Tue Aug 30 20:41:18 2022 +0900  test  **diff --git a/ryuyj/test.txt b/ryuyj/test.txt**  **new file mode 100644**  **index 00000000..00461e36**  **--- /dev/null**  **+++ b/ryuyj/test.txt**  @@ -0,0 +1,3 @@  +test file  +  +test text |
| status | 작업트리(Working tree)를 보이게 한다.(Show the working tree status) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1 (master)  $ git status  On branch master  No commits yet  Untracked files:  (use "git add <file>..." to include in what will be committed)  vue/  nothing added to commit but untracked files present (use "git add" to track) |

## 성장, 표시, 당신의 최근 변경 이력

|  |  |  |
| --- | --- | --- |
| 명령어 | 설명 | 예제 코드 |
| branch | 브런치를 생성하거나 목록을 만든다.  (List, create, or delete branches)  **git branch [-l]**  로컬 branch 정보를 보여줍니다. (-l 옵션은 생략가능)  **git branch -v**  로컬 branch의 정보를 마지막 커밋 내역과 함께 보여줍니다.  **git branch -r**  리모트 저장소의 branch 정보를 보여줍니다.  **git branch -a**  로컬/리모트 저장소의 모든 branch 정보를 보여줍니다.  **git branch (이름)**  로컬에 새로운 branch를 생성합니다.  **git branch -d (branch 이름)**  branch를 삭제합니다. 아직 merge하지 않은 커밋을 담고 있는 경우 삭제되지 않습니다.(강제종료 옵션 -D으로만 삭제 가능)  **git branch -m (변경할 branch이름) (변경될 branch이름)**  A 브랜치를 B 브랜치로 변경합니다. | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch  \* jin  master  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch -v  \* jin f05595f hello!!  master f05595f hello!!  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch -r  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch -a  \* jin  master  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch new\_worker  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch -a  \* jin  master  new\_worker  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch -d new\_worker  Deleted branch new\_worker (was f05595f).  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch -a  \* jin  master  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch new\_worker  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch -m new\_worker k\_worker  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git branch -a  \* jin  k\_worker  master |
| checkout | Switch branches or restore working tree files | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git checkout jin  Switched to branch 'jin'  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ |
| commit | 저장소(Repository)에 변경사항을 기록한다.(Record changes to the repository) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test1/vue/ryuyj (main)  $ git commit . -m 'test'  warning: LF will be replaced by CRLF in ryuyj/test.txt.  The file will have its original line endings in your working directory  [main d3a4ddf3] test  1 file changed, 3 insertions(+)  create mode 100644 ryuyj/test.txt |
| diff | 두 개 이상의 변경사항 (Commit과 작업트리의 Commit등)을 표시 변경 (Show changes between commits, commit and working tree, etc) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git diff master..jin  **diff --git a/test.txt b/test.txt**  **index 522c13b..7f52f14 100644**  **--- a/test.txt**  **+++ b/test.txt**  @@ -1 +1 @@  -Hello~ test !!!!!!!  +test text - 1 |
| merge | 두 개 이상의 개발 과정을 병합 (Join two or more development histories together) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (jin)  $ git checkout master  Switched to branch 'master'  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ ls -al  total 13  drwxr-xr-x 1 jin 197121 0 8월 30 21:14 ./  drwxr-xr-x 1 jin 197121 0 8월 30 18:17 ../  drwxr-xr-x 1 jin 197121 0 8월 30 21:14 .git/  -rw-r--r-- 1 jin 197121 14 8월 9 18:01 test.txt  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git merge jin  Updating f05595f..8c9455d  Fast-forward  test.text | 1 +  1 file changed, 1 insertion(+)  create mode 100644 test.text  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ ls -al  total 14  drwxr-xr-x 1 jin 197121 0 8월 30 21:16 ./  drwxr-xr-x 1 jin 197121 0 8월 30 18:17 ../  drwxr-xr-x 1 jin 197121 0 8월 30 21:16 .git/  -rw-r--r-- 1 jin 197121 7 8월 30 21:16 test.text  -rw-r--r-- 1 jin 197121 14 8월 9 18:01 test.txt |

## 협업 관련 (see also: git help workflows)

|  |  |  |
| --- | --- | --- |
| 명령어 | 설명 | 예제 코드 |
| fetch | 다른 저장소에서 개체를 다운로드하고 바로 병합하지는 않는 상태(Download objects and refs from another repository) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git fetch https://github.com/ryuyj/test.git  fatal: Couldn't find remote ref HEAD |
| pull | 원격 저장소(Repository)에서 로컬저장소로 가져와 로컬지점과 통합한다.)Fetch from and integrate with another repository or a local branch) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git pull  There is no tracking information for the current branch.  Please specify which branch you want to merge with.  See git-pull(1) for details.  git pull <remote> <branch>  If you wish to set tracking information for this branch you can do so with:  git branch --set-upstream-to=origin/<branch> master |
| push | 관련 객체를 가지고 원격저장소에 업데이트 위해 보낸다. 바로업데이트는 되지않는다.(Update remote refs along with associated objects) | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git push  fatal: The current branch master has no upstream branch.  To push the current branch and set the remote as upstream, use  git push --set-upstream origin master  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git push --set-upstream origin master  Enumerating objects: 15, done.  Counting objects: 100% (15/15), done.  Delta compression using up to 12 threads  Compressing objects: 100% (9/9), done.  Writing objects: 100% (15/15), 1.16 KiB | 297.00 KiB/s, done.  Total 15 (delta 0), reused 0 (delta 0), pack-reused 0  To https://github.com/ryuyj/test.git  \* [new branch] master -> master  Branch 'master' set up to track remote branch 'master' from 'origin'. |
| remote | 원격 저장소(remote repository)를 연결 | jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git remote remove origin  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git remote -v  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git remote add origin https://github.com/ryuyj/test.git  jin@DESKTOP-IQ7KA1Q MINGW64 ~/Documents/GitHub/test (master)  $ git remote -v  origin https://github.com/ryuyj/test.git (fetch)  origin https://github.com/ryuyj/test.git (push) |

[공통] 마크다운 markdown 작성법

======================

# 1. 마크다운에 관하여

## 1.1. 마크다운이란?

[\*\*Markdown\*\*](http://whatismarkdown.com/)은 텍스트 기반의 마크업언어로 2004년 존그루버에 의해 만들어졌으며 쉽게 쓰고 읽을 수 있으며 HTML로 변환이 가능하다. 특수기호와 문자를 이용한 매우 간단한 구조의 문법을 사용하여 웹에서도 보다 빠르게 컨텐츠를 작성하고 보다 직관적으로 인식할 수 있다.

마크다운이 최근 각광받기 시작한 이유는 깃헙([https://github.com](https://github.com)) 덕분이다. 깃헙의 저장소Repository에 관한 정보를 기록하는 README.md는 깃헙을 사용하는 사람이라면 누구나 가장 먼저 접하게 되는 마크다운 문서였다. 마크다운을 통해서 설치방법, 소스코드 설명, 이슈 등을 간단하게 기록하고 가독성을 높일 수 있다는 강점이 부각되면서 점점 여러 곳으로 퍼져가게 된다.

## 1.2. 마크다운의 장-단점

### 1.2.1. 장점

1. 간결하다.

2. 별도의 도구없이 작성가능하다.

3. 다양한 형태로 변환이 가능하다.

4. 텍스트(Text)로 저장되기 때문에 용량이 적어 보관이 용이하다.

5. 텍스트파일이기 때문에 버전관리시스템을 이용하여 변경이력을 관리할 수 있다.

6. 지원하는 프로그램과 플랫폼이 다양하다.

### 1.2.2. 단점

1. 표준이 없다.

2. 표준이 없기 때문에 도구에 따라서 변환방식이나 생성물이 다르다.

3. 모든 HTML 마크업을 대신하지 못한다.

\*\*\*\*

# 2. 마크다운 사용법(문법)

## 2.1. 헤더Headers

\* 큰제목: 문서 제목

```

This is an H1

=============

```

This is an H1

=============

\* 작은제목: 문서 부제목

```

This is an H2

-------------

```

This is an H2

-------------

\* 글머리: 1~6까지만 지원

```

# This is a H1

## This is a H2

### This is a H3

#### This is a H4

##### This is a H5

###### This is a H6

```

# This is a H1

## This is a H2

### This is a H3

#### This is a H4

##### This is a H5

###### This is a H6

####### This is a H7(지원하지 않음)

## 2.2. BlockQuote

이메일에서 사용하는 ```>``` 블럭인용문자를 이용한다.

```

> This is a first blockqute.

> > This is a second blockqute.

> > > This is a third blockqute.

```

> This is a first blockqute.

> > This is a second blockqute.

> > > This is a third blockqute.

이 안에서는 다른 마크다운 요소를 포함할 수 있다.

> ### This is a H3

> \* List

> ```

> code

> ```

## 2.3. 목록

### ● 순서있는 목록(번호)

순서있는 목록은 숫자와 점을 사용한다.

```

1. 첫번째

2. 두번째

3. 세번째

```

1. 첫번째

2. 두번째

3. 세번째

\*\*현재까지는 어떤 번호를 입력해도 순서는 내림차순으로 정의된다.\*\*

```

1. 첫번째

3. 세번째

2. 두번째

```

1. 첫번째

3. 세번째

2. 두번째

딱히 개선될 것 같지는 않다. 존 그루버가 신경안쓰고 있다고...

### ● 순서없는 목록(글머리 기호: `\*`, `+`, `-` 지원)

```

\* 빨강

\* 녹색

\* 파랑

+ 빨강

+ 녹색

+ 파랑

- 빨강

- 녹색

- 파랑

```

\* 빨강

\* 녹색

\* 파랑

+ 빨강

+ 녹색

+ 파랑

- 빨강

- 녹색

- 파랑

혼합해서 사용하는 것도 가능하다(내가 선호하는 방식)

```

\* 1단계

- 2단계

+ 3단계

+ 4단계

```

\* 1단계

- 2단계

+ 3단계

+ 4단계

## 2.4. 코드

4개의 공백 또는 하나의 탭으로 들여쓰기를 만나면 변환되기 시작하여 들여쓰지 않은 행을 만날때까지 변환이 계속된다.

### 2.4.1. 들여쓰기

```

This is a normal paragraph:

This is a code block.

end code block.

```

실제로 적용해보면,

적용예:

\*\*\*\*\*

This is a normal paragraph:

This is a code block.

end code block.

\*\*\*\*\*

> 한줄 띄어쓰지 않으면 인식이 제대로 안되는 문제가 발생합니다.

```

This is a normal paragraph:

This is a code block.

end code block.

```

적용예:

\*\*\*\*\*

This is a normal paragraph:

This is a code block.

end code block.

\*\*\*\*\*

### 2.4.1. 코드블럭

코드블럭은 다음과 같이 2가지 방식을 사용할 수 있습니다:

\* `<pre><code>{code}</code></pre>` 이용방식

```

<pre>

<code>

public class BootSpringBootApplication {

public static void main(String[] args) {

System.out.println("Hello, Honeymon");

}

}

</code>

</pre>

```

<pre>

<code>

public class BootSpringBootApplication {

public static void main(String[] args) {

System.out.println("Hello, Honeymon");

}

}

</code>

</pre>

\* 코드블럭코드("\```") 을 이용하는 방법

<pre>

<code>

```

public class BootSpringBootApplication {

public static void main(String[] args) {

System.out.println("Hello, Honeymon");

}

}

```

</code>

</pre>

```

public class BootSpringBootApplication {

public static void main(String[] args) {

System.out.println("Hello, Honeymon");

}

}

```

\*\*깃헙\*\*에서는 코드블럭코드("\```") 시작점에 사용하는 언어를 선언하여 [문법강조(Syntax highlighting)](https://docs.github.com/en/github/writing-on-github/creating-and-highlighting-code-blocks#syntax-highlighting)이 가능하다.

<pre>

<code>

```java

public class BootSpringBootApplication {

public static void main(String[] args) {

System.out.println("Hello, Honeymon");

}

}

```

</code>

</pre>

```java

public class BootSpringBootApplication {

public static void main(String[] args) {

System.out.println("Hello, Honeymon");

}

}

```

## 2.5. 수평선 ```<hr/>```

아래 줄은 모두 수평선을 만든다. 마크다운 문서를 미리보기로 출력할 때 \*페이지 나누기\* 용도로 많이 사용한다.

```

\* \* \*

\*\*\*

\*\*\*\*\*

- - -

---------------------------------------

```

\* 적용예

\* \* \*

\*\*\*

\*\*\*\*\*

- - -

---------------------------------------

## 2.6. 링크

\* 참조링크

```

[link keyword][id]

[id]: URL "Optional Title here"

// code

Link: [Google][googlelink]

[googlelink]: https://google.com "Go google"

```

Link: [Google][googlelink]

[googlelink]: https://google.com "Go google"

\* 외부링크

```

사용문법: [Title](link)

적용예: [Google](https://google.com, "google link")

```

Link: [Google](https://google.com, "google link")

\* 자동연결

```

일반적인 URL 혹은 이메일주소인 경우 적절한 형식으로 링크를 형성한다.

\* 외부링크: <http://example.com/>

\* 이메일링크: <address@example.com>

```

\* 외부링크: <http://example.com/>

\* 이메일링크: <address@example.com>

## 2.7. 강조

```

\*single asterisks\*

\_single underscores\_

\*\*double asterisks\*\*

\_\_double underscores\_\_

~~cancelline~~

```

\* \*single asterisks\*

\* \_single underscores\_

\* \*\*double asterisks\*\*

\* \_\_double underscores\_\_

\* ~~cancelline~~

> ```문장 중간에 사용할 경우에는 \*\*띄어쓰기\*\* 를 사용하는 것이 좋다.```

> 문장 중간에 사용할 경우에는 띄어쓰기를 사용하는 것이 좋다.

## 2.8. 이미지

```

![Alt text](/path/to/img.jpg)

![Alt text](/path/to/img.jpg "Optional title")

```

![석촌호수 러버덕](http://cfile6.uf.tistory.com/image/2426E646543C9B4532C7B0)

![석촌호수 러버덕](http://cfile6.uf.tistory.com/image/2426E646543C9B4532C7B0 "RubberDuck")

사이즈 조절 기능은 없기 때문에 ```<img width="" height=""></img>```를 이용한다.

예

```

<img src="/path/to/img.jpg" width="450px" height="300px" title="px(픽셀) 크기 설정" alt="RubberDuck"></img><br/>

<img src="/path/to/img.jpg" width="40%" height="30%" title="px(픽셀) 크기 설정" alt="RubberDuck"></img>

```

<img src="http://cfile6.uf.tistory.com/image/2426E646543C9B4532C7B0" width="450px" height="300px" title="px(픽셀) 크기 설정" alt="RubberDuck"></img><br/>

<img src="http://cfile6.uf.tistory.com/image/2426E646543C9B4532C7B0" width="40%" height="30%" title="%(비율) 크기 설정" alt="RubberDuck"></img>

## 2.9. 줄바꿈

3칸 이상 띄어쓰기(` `)를 하면 줄이 바뀐다.

```

\* 줄 바꿈을 하기 위해서는 문장 마지막에서 3칸이상을 띄어쓰기해야 한다.

이렇게

\* 줄 바꿈을 하기 위해서는 문장 마지막에서 3칸이상을 띄어쓰기해야 한다.\_\_\_\\ 띄어쓰기

이렇게

```

\* 줄 바꿈을 하기 위해서는 문장 마지막에서 3칸이상을 띄어쓰기해야 한다. 이렇게

\* 줄 바꿈을 하기 위해서는 문장 마지막에서 3칸이상을 띄어쓰기해야 한다. \

이렇게

\*\*\*\*

# 3. 마크다운 사용기

## 3.1. 위지윅(WSYWIG) 에디터

우리가 흔하게 접하는 웹에서 사용되는 에디터(네이버, 다음, 구글 등)이 대부분 위지윅 에디터에 속하며 기본적으로 HTML을 이용하여 스타일을 적용하여 문장을 꾸미는 형태를 취하게 된다. 그래서 하루패드와 같은 마크다운 에디터의 View 영역의 내용을 복사하여 붙여넣기를 하면 대체적으로 View영역에서 보이는 그대로 복사되는 편이다. 다만, 붙여넣기 이후에 문장들을 수정하려고 할 떄 문제가 되는데, 이는 스타일이 포함된 태그가 수정과정에서 변형되면서 전체적인 영향을 끼치는 탓이다. 티스토리 블로그에서는 쉽지 않고... 워드프레스의 경우에는 마크다운으로 작성된 포스트를 HTML로 변환해주는 기능을 활용하는 것이 좋다.

결론은, \*\*복사해서 붙여넣기하면 가급적이면 본문은 수정하지 않는 것이 좋다.\*\*

## 3.2. 깃헙Github, 비트버킷Bitbucket과 요비Yobi 등

최근 유행하는 협업개발플랫폼의 경우에는 마크다운을 변환하는 컨버터 기능을 기본탑재하고 있기 때문에 마크다운 문법으로 작성한 텍스트를 그대로 복사해서 붙여넣거나 업로드하는 것만으로 마크다운의 적용이 가능하다.

## 3.3. MS워드 적용

View 영역의 항목을 그대로 붙여넣거나 HTML 내보내기 등으로 생성한 파일을 불러오는 형태로 사용가능하다. 적용한 헤더를 워드가 읽어드리면서 목차에 적용하기 때문에 이를 활용하면 목차까지도 손쉽게 적용이 가능해진다.

\*\*\*\*\*

# 4. 정리

마크다운은 기본문법만 알고있다면 일반 텍스트편집기에서도 손쉽게 작성이 가능한 마크업언어다. 현재 다양한 도구와 플랫폼에서 지원하고 있기 때문에 더욱 손쉽게 스타일적용된 문서를 작성할 수 있어 점점 널리 사용되고 있다.

> 마크다운을 이해하고 사용하면서 쉽고 빠르게 스타일문서를 작성해보세요.

저는 Dropbox 프로를 구매해서 집-랩탑-스마트폰이 각각 연동을 시켜서 사용하고 있습니다. 드랍박스에 저장된 마크다운 문서는 Dropbox 웹서비스 상에서 제공하기 때문에 웹상에서 바로 열람할 수도 있어 링크를 걸어서 다른 사람과 공유하는 형식으로 사용하고 있다.

\* 링크 예: [Markdown 설명](https://www.dropbox.com/s/mzp9tq4qtfjdlif/20141021\_markdown\_use\_tip.md?dl=0)

\*\*\*\*\*