

세상의 속도를  
따라잡고 싶다면

# Do it!

IT 분야 베스트셀러!  
전면 개정판

## 지옥에서 온 문서 관리자 깃&깃허브 입문

30만 구독자의 선택! 생활코딩의 영상 중  
꼭 필요한 내용만 모아 이 한 권에 엮었다!

Microsoft MVP 고경희 | 생활코딩 운영자 이고잉 지음

이지스퍼블리싱

  
자주 쓰는  
최신 기능  
총망라!

  
동영상  
강의 제공!

  
VS Code로  
깃&깃허브  
활용까지!

01

## 깃 시작하기

01-1 지옥에서 온 문서 관리자, 깃

01-2 윈도우에서 깃 설치하기

01-3 리눅스 명령 연습하기

01-4 빔 편집기에서 텍스트 문서 만들기

참고 영상: GIT 1  
<http://bit.ly/do-it-git1>



## 깃으로 무엇을 할 수 있을까?

### 버전 관리

- 문서를 수정할 때마다 언제 수정했는지, 어떤 것을 변경했는지 등을 구체적으로 기록하는 **버전 관리 시스템이 바로 깃(Git)**

### 3가지 기능은 순서대로 배워야 합니다

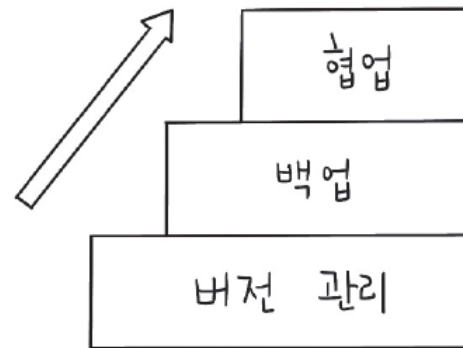
버전 관리를 이해하지 못하면 백업을 이해할 수 없고,  
백업을 이해하지 못하면 협업을 이해할 수 없음

### 백업

- 백업은 현재 컴퓨터에 있는 자료를 다른 컴퓨터에 복제하는 것
- 깃 파일을 위한 백업 공간을 제공하는 인터넷 서비스를 **원격 저장소** 또는 **온라인 저장소**라고 함
- 서비스 가운데 가장 많이 쓰이는 것이 바로 **깃허브(GitHub)**

### 협업

- 깃허브와 같은 온라인 서비스를 사용하면 여러 사람이 파일을 편하게 주고받으면서 일할 수 있음
- 누가 어느 부분을 어떻게 수정했는지 기록으로 남기 때문에 나중에 오류가 생겼을 때도 파악하기 쉬움
- 협업 과정에서 일어날 수 있는 여러 문제를 중간에서 정리해 주는 기능도 함



## 깃 프로그램의 종류

### 깃허브 데스크톱

- 깃 온라인 저장소 서비스인 깃허브에서 제공하는 프로그램(<https://desktop.github.com>)
- 그래픽 사용자 인터페이스(graphic user interface, GUI)로 구현되어 사용하기 쉽고 누구나 배울 수 있음
- 자주 쓰는 기본적인 기능 위주여서 깃 고급 사용자가 되면 아쉬울 수 있음

### 토터스깃

- 윈도우 탐색기의 빠른 메뉴에 추가되는 윈도우 전용 프로그램(<https://tortoisegit.org/download/>)

### 소스트리

- 깃의 기본 기능부터 고급 기능까지 사용할 수 있는 프로그램(<https://www.sourcetreeapp.com/>)
- 기능이 많아 사용법은 복잡하지만 어느 정도 익숙해지면 깃을 자유롭게 활용할 수 있음

## 커맨드 라인 인터페이스

- 커맨드 라인 인터페이스(command line interface, CLI)는 터미널 창에 직접 명령을 입력해서 깃을 사용하는 방식
- 이 방식은 소스트리나 깃허브 데스크톱 등 그래픽 사용자 인터페이스로 만든 프로그램으로, 리눅스의 기본 명령을 알아야 하고, 깃 명령도 외워야 하기 때문에 깃을 사용하는 것보다 어려움  
하지만 이 방법에 익숙해지면 깃을 훨씬 빠르게 다룰 수 있음
- 반복할 일을 자동화하거나 서버 환경에서 깃을 사용하는 등 다양하게 활용
- 개발자들은 대부분 커맨드 라인 인터페이스(CLI)로 깃을 사용

## 윈도우에 깃 설치하기 - (1)

1. <https://git-scm.com/> 사이트에 접속하면 운영체제에 따라 프로그램을 내려받을 수 있는 화면이 나타남  
화면 오른쪽 아래에서 [Download 2.xx.x for Windows]를 클릭
2. 내려받을 수 있는 파일 목록이 있는 화면으로 이동  
맨 위에 있는 [Click here to download]를 클릭해 파일을 내려받은 다음 파일을 실행
3. 첫 화면에서는 라이선스 정보를 확인  
이어서 화면마다 [Next]를 클릭해 설치할 경로와 구성 요소, 그리고 시작 메뉴에 표시할 메뉴 이름 등은 기본값 그대로 사용
4. 깃에서 사용할 기본 편집기를 선택  
기본값 'Use Vim (the ubiquitous text editor) as Git's default editor'가 선택된 상태로 [Next]를 클릭
5. 기존 깃에서는 사용자 컴퓨터에 저장소를 만들 때 master라는 브랜치 이름을 사용했지만 최신 깃에서는 main 브랜치를 사용  
2개의 옵션 중 'Override the default branch name for new repositories'를 선택하고 [Next]를 클릭
6. 커맨드 라인에서 어떤 방법으로 깃을 사용할지 선택  
기본값 'Git from the commandline and also from 3rd-party software'가 선택된 상태로 [Next]를 클릭  
이어서 보안 서버에 접속하는 방법을 선택  
기본값 'Use bundled OpenSSH'가 선택된 상태로 [Next]를 클릭

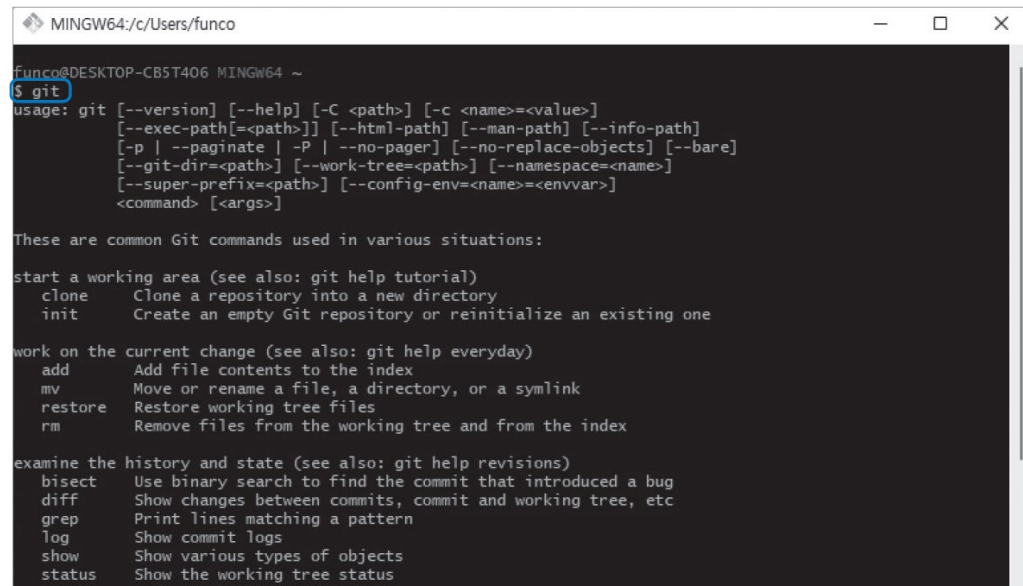
## 윈도우에 깃 설치하기 - (2)

7. HTTPS처럼 보안이 추가된 연결에 어떻게 연결할 것인지 선택  
기본값 'Use the OpenSSL Library'를 선택하고 [Next]를 클릭  
이어서 텍스트 파일에서 줄 끝부분을 어떻게 처리할 것인지 선택  
기본값 'Checkout Windows-style, commit Unix-style line endings'가 선택된 상태로 [Next]를 클릭
8. 터미널 에뮬레이터를 선택  
기본값 'Use MinTTY'가 선택된 상태로 [Next]를 클릭  
깃의 pull 명령을 어떻게 처리할 것인지 선택  
기본값 'Default'가 선택된 상태에서 [Next]를 클릭
9. 이어서 기본값 'Git Credential Manager'가 선택된 상태로 [Next]를 클릭  
파일 시스템을 캐싱하도록 설정하면 버전 관리를 좀 더 빠르게 실행할 수 있음  
이어서 나오는 화면에서는 'Enable file system caching'이 선택된 상태로 [Next]를 클릭
10. 제시된 옵션을 시험 삼아 사용해 볼 것인지 묻는데, 여기에서는 아무것도 선택하지 말고 [Install]을 클릭해 설치를 시작  
[Finish]를 클릭해 깃 설치를 끝냄

## 윈도우에서 깃 실행해 보기

1. 윈도우 작업 표시줄의 검색 창에 'git'이라고 입력한 후 검색 결과 중에서 [Git Bash]를 선택
2. 배시 창이 열리면 'git'이라고 입력한 후 {{ Enter }}를 누름  
깃 명령에서 사용할 수 있는 여러 옵션이 표시된다면 깃이 제대로 설치된 것

```
$ git
```



```
Funco@DESKTOP-CB5T406 MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:



start a working area (see also: git help tutorial)
  clone Clone a repository into a new directory
  init   Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add    Add file contents to the index
  mv     Move or rename a file, a directory, or a symlink
  restore Restore working tree files
  rm     Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect Use binary search to find the commit that introduced a bug
  diff   Show changes between commits, commit and working tree, etc
  grep   Print lines matching a pattern
  log    Show commit logs
  show   Show various types of objects
  status Show the working tree status
```

## 맥에 깃 설치하기 - (1)

### 홈브류 설치하기

1. <https://brew.sh/>를 입력해 홈브류 사이트로 이동한 다음 언어를 '한국어'로 선택  
'Homebrew 설치하기' 아래에 있는 설치 명령을 복사  
오른쪽에 있는 를 클릭하면 명령을 간단히 복사할 수 있음
2. 맥에서 터미널을 열고 {{ Command + V }}를 눌러 복사한 명령을 붙여 넣고 {{ Enter }}를 누름
3. 맥에서 사용하는 비밀번호를 입력하고 나면 홈브류가 설치되기 시작

## 맥에 깃 설치하기 - (2)

### 홈브류에서 깃 설치하기

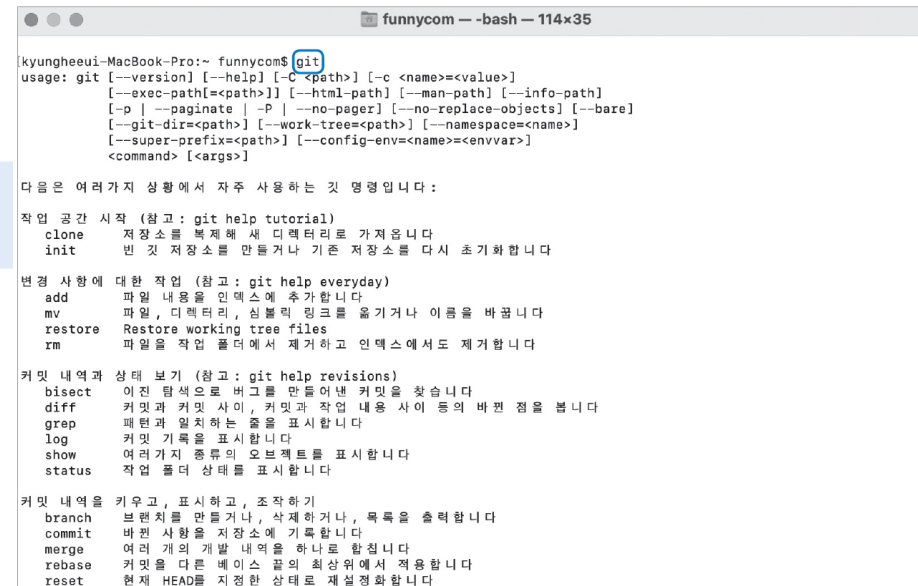
1. 맥 터미널에서 다음과 같이 입력한 후 {{ Enter }}를 누름

```
$ brew install git
```

2. 따로 설정하지 않아도 깃이 설치되며 설치가 끝나면 \$ 표시가 나타남
3. 깃 설치가 끝났다면 \$ 옆에 'git'이라고 입력

깃과 관련된 명령이 나타난다면 깃 설치의 일단 성공한 것

```
$ git
```



```
kyungheui-MacBook-Pro:~ funnycom$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

다음은 여러가지 상황에서 자주 사용하는 깃 명령입니다:

작업 공간 시작 (참고: git help tutorial)
clone  저장소를 복제해 새 디렉터리로 가져옵니다
init   빈 깃 저장소를 만들거나 기존 저장소를 다시 초기화합니다

변경 사항에 대한 작업 (참고: git help everyday)
add    파일 내용을 인덱스에 추가합니다
mv     파일, 디렉터리, 심볼릭 링크를 옮기거나 이름을 바꿉니다
restore Restore working tree files
rm     파일을 작업 풀더에서 제거하고 인덱스에서도 제거합니다

커밋 내역과 상태 보기 (참고: git help revisions)
bisect 이전 탐색으로 버그를 만들어낸 커밋을 찾습니다
diff   커밋과 커밋 사이, 커밋과 작업 내용 사이 등의 바뀐 점을 봅니다
grep   패턴과 일치하는 줄을 표시합니다
log    커밋 기록을 표시합니다
show   여러가지 종류의 오브젝트를 표시합니다
status 작업 풀더 상태를 표시합니다

커밋 내역을 키우고, 표시하고, 조작하기
branch 브랜치를 만들거나, 삭제하거나, 목록을 출력합니다
commit 바뀐 사항을 저장소에 기록합니다
merge  여러 개의 개발 내역을 하나로 합칩니다
rebase 커밋을 다른 베이스 끝의 최상위에서 적용합니다
reset  현재 HEAD를 지정한 상태로 재설정화합니다
```

## 깃 환경 설정하기

- 깃을 사용하기 전에 먼저 사용자 정보를 입력해야 함  
깃은 버전을 저장할 때마다 그 버전을 만든 사용자 정보도 함께 저장하기 때문  
이를 통해 어떤 버전을 누가 언제 만들었는지 쉽게 파악할 수 있음
- 여기서부터는 운영체제와 상관없이 리눅스 방식의 명령을 사용
- 사용하는 운영체제가 윈도우라면 깃 배시를, 맥이라면 터미널 창을 열고 터미널 창에  
다음과 같이 입력한 뒤, 사용자의 이름과 이메일 주소를 저장

```
$ git config --global user.name "이름"  
$ git config --global user.email "이메일 주소"
```

- 깃에서 사용자 정보를 설정하려면 git config 명령을 사용
- --global 옵션을 추가하면 현재 컴퓨터에 있는 모든 저장소에서 같은 사용자 정보를 사용

## 현재 디렉터리 살펴보기 - (1)

1. 깃 배시를 실행한 후 커서 윗줄을 보면 맨 끝에 물결 표시(~)가 있음  
현재 위치가 홈 디렉터리(home directory)라는 의미

2. pwd 명령을 입력하고 {{ Enter }}를 누름  
현재 위치의 경로가 나타남

```
$ pwd
```

3. 현재 디렉터리에 어떤 파일이나 디렉터리가 있는지 확인할 때는 ls 명령을 사용  
ls 명령만 입력하고 {{ Enter }}를 누르면 디렉터리와 파일 이름이 나타남  
이름 뒤에 슬래시(/)가 붙어 있는 것은 디렉터리

```
$ ls
```

## 현재 디렉터리 살펴보기 - (2)

4. ls 명령 뒤에 -l 옵션을 붙이면 파일이나 디렉터리의 상세 정보까지 표시할 수 있음

```
$ ls -l
```

### ls 명령 옵션 모음

- ls 명령을 사용할 때 옵션을 추가하면 파일과 디렉터를 다양한 형식으로 표시할 수 있음
- ls 명령 다음에 붙임표(-)을 붙이고 옵션을 나타내는 문자를 작성
- 이때 -al처럼 옵션을 2개 이상 사용할 수 있음

옵션	설명
-a	숨긴 파일이나 디렉터리도 함께 표시합니다.
-l	파일이나 디렉터리의 상세 정보를 함께 표시합니다.
-r	파일의 정렬 순서를 거꾸로 표시합니다.
-t	파일 작성 시간순으로 (내림차순) 표시합니다.

## 터미널 창 지우기

- 터미널에서 여러 소스를 입력하다 보면 화면이 가득 차서 결과를 쉽게 확인하기 어려울 때가 있음  
이럴 때 clear 명령을 사용하면 터미널 화면을 깨끗하게 비울 수 있음

```
$ clear
```

## 터미널 창에서 디렉터리 이동하기 - (1)

1. 터미널 창에서 디렉터리 사이를 이동할 때는 'cd'라는 명령을 사용  
현재 위치에서 상위 디렉터리로 이동하려면 다음과 같이 cd 명령 다음에 한 칸 띄고 마침표 2개를 입력

```
$ cd ..
```

2. cd 명령을 실행한 후 \$ 기호 위에 표시된 경로를 확인  
끝부분에 c/Users라고 나타나는데 이는 c/Users/사용자 아이디에서 한 단계 위로 올라간 경로
3. 이번에는 c 드라이브의 루트 폴더, 즉 c:\까지 이동하고 ls 명령을 사용해서 그 안의 내용을 확인

```
$ cd ..  
$ ls
```

4. 하위 디렉터리로 이동할 때는 cd 명령 다음에 이동할 하위 디렉터리 이름을 입력  
현재 c/ 디렉터리에 있는데 c/Users 디렉터리로 가려면 다음과 같이 명령을 입력

```
$ cd Users
```

## 터미널 창에서 디렉터리 이동하기 - (2)

5. 처음에 출발했던 디렉터리, 즉 홈 디렉터리로 돌아가려면 다음과 같이 입력

```
$ cd ~
```

## 리눅스에서 디렉터리를 나타내는 기호

- 리눅스에서는 현재 위치나 파일 경로를 나타낼 때 몇 가지 약속된 기호를 사용

기호	설명
~	현재 접속 중인 사용자 디렉터를 가리킵니다. 'c/Users/사용자 아이디'가 사용자 디렉터리입니다. 터미널 창에서 \$ 기호 뒷줄에 있는 ~가 바로 사용자 디렉터를 가리킵니다. 사용자 아이디는 5글자까지만 나타납니다.
.	현재 사용자가 작업 중인 디렉터리입니다.
..	현재 디렉터리의 상위 디렉터리입니다.

## 터미널 창에서 디렉터리 만들기 및 삭제하기 – (1)

1. 터미널 창에서 현재 디렉터리 안에 하위 디렉터리를 만들 때는 'mkdir'이라는 명령을 사용  
홈 디렉터리 안에 있는 Documents 디렉터리에 test라는 하위 디렉터를 만든다면 다음과 같이 작성

```
$ cd Documents  
$ mkdir test
```

2. mkdir 명령을 실행하고 하위 디렉터리가 만들어져도 화면에는 아무것도 나타나지 않음  
test 디렉터리가 제대로 만들어졌는지 확인하려면 ls 명령을 사용하면 화면에 'test/'라고 표시

```
$ ls
```

### 터미널 창에서 디렉터리 만들기 및 삭제하기 - (2)

3. 디렉터리를 삭제할 때는 'rm'이라는 명령을 사용

단, 명심할 것은 삭제할 디렉터리의 상위 디렉터리에서 해야 함


Documents 디렉터리 안에 있는 test 디렉터를 삭제하려면 Document 디렉터리에서 rm 명령을 사용

이때 -r 옵션을 붙이면 디렉터리 안에 있는 하위 디렉터리와 파일을 함께 삭제할 수 있음

```
$ rm -r test
```

4. test 디렉터를 삭제한 후 ls 명령을 실행해 보면 test 디렉터리가 삭제되어 있음

### 터미널 종료하기

- 터미널 창을 닫을 때 창의 오른쪽 위에 있는 를 클릭해도 되지만, 터미널 창에 'exit' 명령을 입력해서 종료할 수도 있음

```
$ exit
```

## vim이란

- 터미널 화면에서 키보드만 이용해서 텍스트 문서를 만들 수 있다면 시간도 단축되고, 키보드와 마우스를 분주하게 오가지 않아도 됨  
그래서 자주 사용하는 텍스트 편집기가 바로 vim(Vim)!
- vim 편집기는 터미널 창에서 키보드에서 입력하는 것만으로도 작성할 수 있다는 점에서 윈도우에서 사용하던 일반 편집기와 다름

## 빔에서 문서 작성하고 저장하기 - (1)

1. 깃 배시 프로그램을 실행해서 터미널 창을 열면 홈 디렉터리부터 시작

Documents 디렉터리로 이동한 후 test 디렉터를 만들고 test 디렉터리로 이동

```
$ cd Documents  
$ mkdir test  
$ cd test
```

2. 현재 디렉터리인 test 디렉터리에 test.txt 파일을 만들기 위해 다음과 같이 vim 명령을 입력

vim 명령 다음에 파일 이름을 입력했을 때, 그 이름과 같은 파일이 없다면 그 이름으로 새로운 텍스트 문서를 만듦

```
$ vim test.txt
```

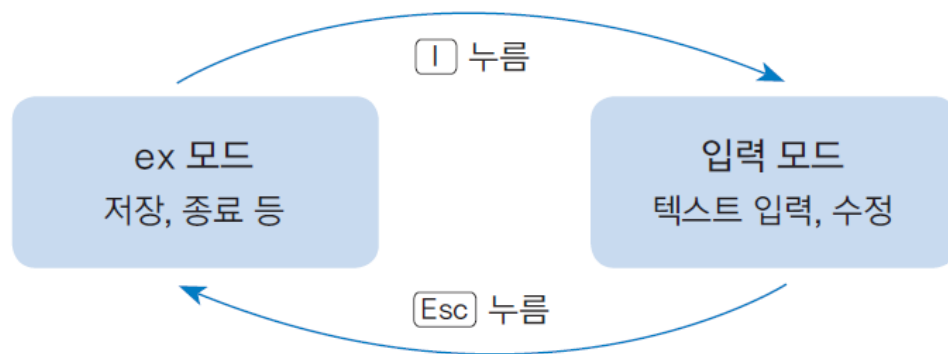
### vim에서 문서 작성하고 저장하기 - (2)

3. 명령을 입력했을 때 다음과 같이 화면이 바뀌면 vim이 잘 실행된 것  
화면 왼쪽 위에는 커서가 깜박이고 왼쪽 아래에는 현재 열려 있는 파일 이름이 표시됨



### vim에서 문서 작성하고 저장하기 - (3)

4. 새로 만든 파일에 아무 내용이나 입력해 보면 제대로 입력되지 않음  
vim에는 문서를 작성하는 '입력 모드'와 문서를 저장하는 'ex 모드'가 있음  
vim은 처음에 'ex모드'로 열리므로 어떤 키를 눌러도 반응이 없는 것



### vim에서 문서 작성하고 저장하기 - (4)

5. vim 편집기에서 텍스트를 입력하려면 ex 모드 상태에서 `{{ I }}` 또는 `{{ A }}`를 눌러서 입력 모드 상태로 바꿔야 함  
입력 모드 상태가 되면 화면 맨 아래 'INSERT' 또는 '끼워넣기'라는 단어가 뜨는데, 이때부터 텍스트를 입력할 수 있음  
test.txt 파일에 간단하게 텍스트를 입력
6. 텍스트를 입력하고 나서 파일을 저장하려면 다시 ex 모드로 돌아가야 함  
입력 모드에서 ex 모드로 돌아가려면 `{{ Esc }}`를 누르고 `:`를 입력  
원래 'INSERT' 또는 '끼워넣기'가 있던 자리에 텍스트를 입력할 수 있음  
그리고 `:` 뒤에 `wq`라고 명령을 입력한 후 `{{ Enter }}`를 누름
7. `{{ Enter }}`를 누르면 작성한 파일이 저장되고 편집기가 종료되면서 vim 편집기를 시작했던 터미널 창으로 되돌아감

## vim의 ex 모드 명령

- vim의 ex 모드에서 사용하는 명령은 콜론(:)으로 시작

명령	설명
:w 또는 :write	편집하던 문서를 저장합니다.
:q 또는 :quit	편집기를 종료합니다.
:wq	편집하던 문서를 저장하고 종료합니다.
:q!	편집하던 문서를 저장하지 않고 편집기를 종료합니다. 확장자가 .swp인 임시 파일이 생깁니다.
:wq 파일명	편집하던 문서를 지정한 파일 이름으로 저장합니다.

## cat 명령으로 텍스트 문서 확인하기

- 터미널 창에서 텍스트 문서의 내용을 간단히 확인할 때는 리눅스의 cat 명령을 사용
- cat 명령 다음에 텍스트 파일 이름을 함께 사용하면 터미널 창에 그 텍스트 파일 내용을 보여줌
- 터미널 창에서 cat 명령 다음에 test.txt를 입력하면 앞에서 작성했던 test.txt 파일의 내용을 확인할 수 있음

```
$ cat test.txt
```

## cat 명령 모음

명령	기능
\$ cat file	file의 내용을 화면에 표시합니다.
\$ cat file(s) > Newfile	file(s)를 차례로 연결해서 새로운 파일인 Newfile을 만듭니다.
\$ cat file1 >> file2	file1의 내용을 file2의 내용 끝에 연결합니다.