

# Git & GitHub

◆ 깃으로 버전 관리하기

정수아

# Contents

**01** 깃 저장소 만들기

**02** 버전 만들기

**03** 커밋 내용 확인하기

**04** .gitignore

**05** 단계마다 파일 상태 알아보기

**06** 작업 되돌리기

**07** 특정 커밋으로 되돌리기

**08** 정리하기



01

## 깃 저장소 만들기

# 깃 저장소 만들기

## ❖ hello-git 디렉터리 생성 후 이동

```
$ mkdir hello-git  
$ cd hello-git
```



```
MINGW64:/c/Users/KBS/hello-git  
  
KBS@DESKTOP-88BT23R MINGW64 ~  
$ mkdir hello-git  
  
KBS@DESKTOP-88BT23R MINGW64 ~  
$ cd hello-git
```

# 깃 저장소 만들기

## ❖ 깃 초기화 하기

- 디렉터리를 깃 저장소로 사용할 수 있도록 만들어 줌
- 초기화 후 .git 디렉터리(숨김 폴더)가 생성
  - 깃을 사용하면서 생성하는 버전 정보가 저장될 저장소

```
$ git init
```

A screenshot of a Windows command prompt window titled 'MINGW64:/c/Users/KBS/hello-git'. The prompt shows the user 'KBS@DESKTOP-88BT23R' in the directory '~/hello-git'. The command '\$ git init' has been entered, and the output 'Initialized empty Git repository in C:/Users/KBS/hello-git/.git/' is displayed. The output line is highlighted with a red rectangular box. The prompt then shows '(master)' and '\$' on the next line.

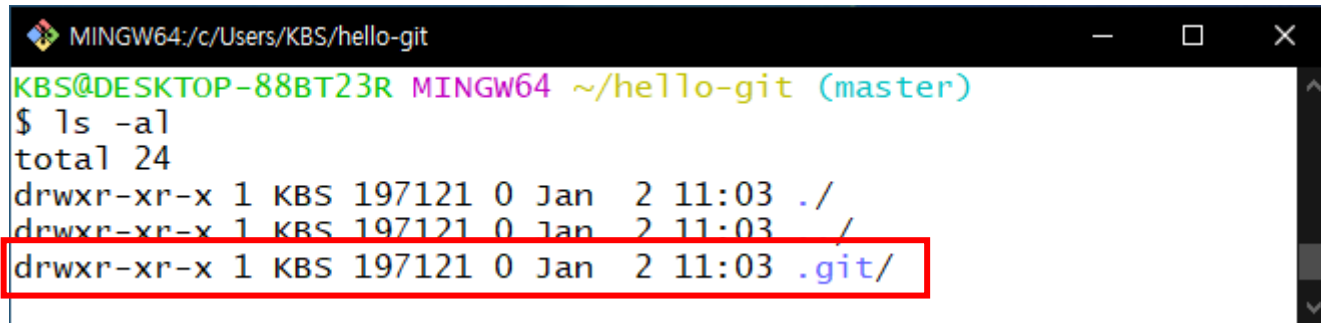
```
MINGW64:/c/Users/KBS/hello-git  
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git  
$ git init  
Initialized empty Git repository in C:/Users/KBS/hello-git/.git/  
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)  
$
```

# 깃 저장소 만들기

## ❖ 깃 초기화 하기

- 디렉터리 내부 확인

```
$ ls -al
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ ls -al
total 24
drwxr-xr-x 1 KBS 197121 0 Jan  2 11:03 ./
drwxr-xr-x 1 KBS 197121 0 Jan  2 11:03 ../
drwxr-xr-x 1 KBS 197121 0 Jan  2 11:03 .git/
```

- .git 디렉터리
  - 깃을 사용하면서 버전이 저장될 저장소



02

## 버전 만들기

# 버전 만들기

## ❖ 버전이란?

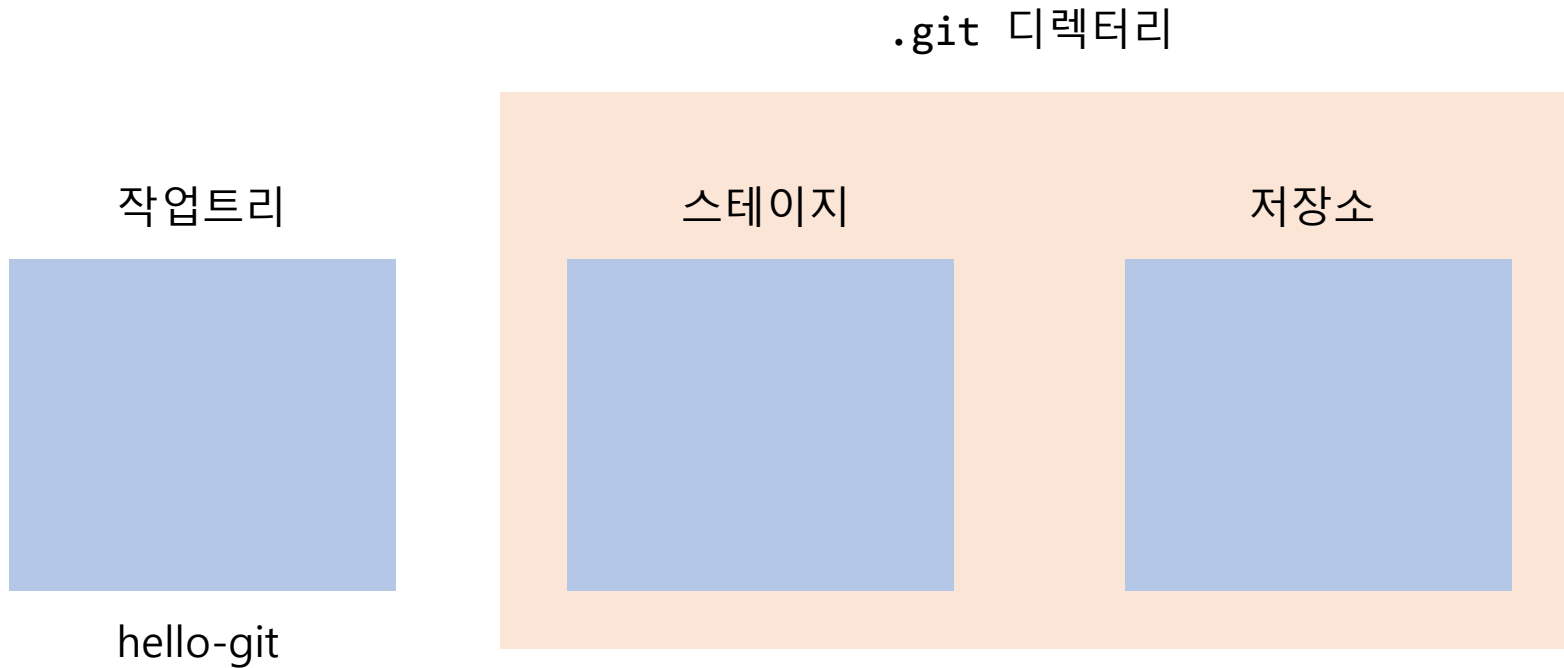
- 깃에서 문서를 수정하고 저장할 때마다 생기는 것
- 특징
  - 버전마다 변경 시점과 변경 내용을 확인할 수 있음
  - 이전 버전으로 되돌아갈 수 있음

깃은 어떻게 파일 이름은 그대로 유지하면서 수정 내역을 기록할까?



# 버전 만들기

## ❖ 스테이지와 커밋



# 버전 만들기

## ❖ 작업 트리(working tree)

- 파일 수정, 저장 등의 작업을 하는 디렉터리
- 작업 디렉터리(working directory)라고도 함
- 우리 눈에 보이는 디렉터리를 말함

## ❖ 스테이지(stage)

- 버전으로 만들 파일이 대기하는 곳
- 눈에 보이지 않음

## ❖ 저장소(repository)

- 스테이지에서 대기하고 있던 파일들을 버전으로 만들어 저장하는 장소
- 눈에 보이지 않음

# 버전 만들기

## ❖ 버전 생성 과정

1) 작업 트리에서 파일을 수정하고 저장



2) 깃으로 버전 관리를 하기 위해 스테이지에 등록



# 버전 만들기

## ❖ 버전 생성 과정

3) 버전을 만들기 위해 '커밋(commit)' 명령을 실행



4) 스테이지에 있던 파일을 저장소에 저장(V1)



# 버전 만들기

## ❖ 버전 생성 과정

5) 작업 트리에서 파일을 수정하고 저장



6) 깃으로 버전 관리를 하기 위해 스테이지에 등록



# 버전 만들기

## ❖ 버전 생성 과정

7) 버전을 만들기 위해 '커밋(commit)' 명령을 실행



8) 스테이지에 있던 파일을 저장소에 새로운 버전으로 저장(V2)

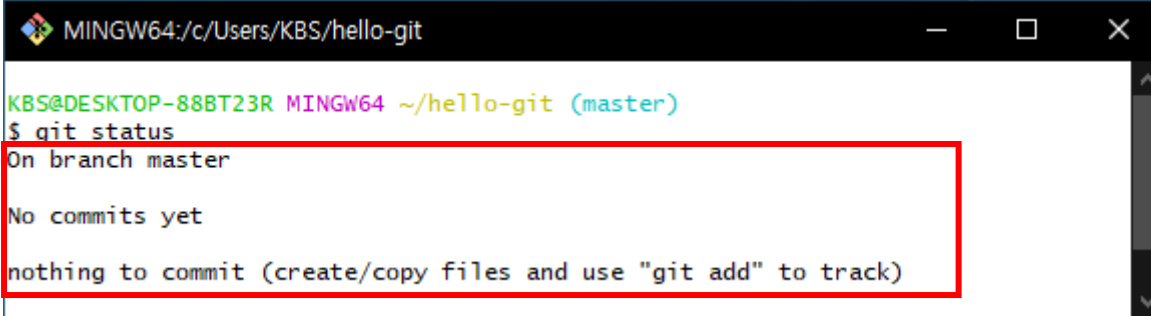


# [실습] 버전 만들기

## ❖ 작업 트리에서 vim으로 문서 수정하기

- hello-git 디렉터리로 이동 후, 깃 상태 확인

```
$ git status
```



A terminal window titled 'MINGW64:/c/Users/KBS/hello-git' showing the output of the 'git status' command. The output is as follows:

```
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
No commits yet
nothing to commit (create/copy files and use "git add" to track)
```

Three numbered annotations are present on the left side of the terminal output:

- ① points to 'On branch master'
- ② points to 'No commits yet'
- ③ points to 'nothing to commit (create/copy files and use "git add" to track)'

- ① 현재 master 브랜치에 있음
- ② 아직 커밋한 파일이 없음
- ③ 현재 커밋할 파일이 없음

# [실습] 버전 만들기

## ❖ 작업 트리에서 vim으로 문서 수정하기

- 새로운 파일(hello.txt) 생성

```
$ vim hello.txt
```

- 입력 모드(A 또는 I)로 변경 후, 내용 입력



```
MINGW64:/c/Users/sophia/hello-git
안녕하세요!
정수아입니다.
~
~
~
~
~
~
hello.txt[+] [unix] (08:59 01/01/1970) 2,1 All
-- INSERT --
```

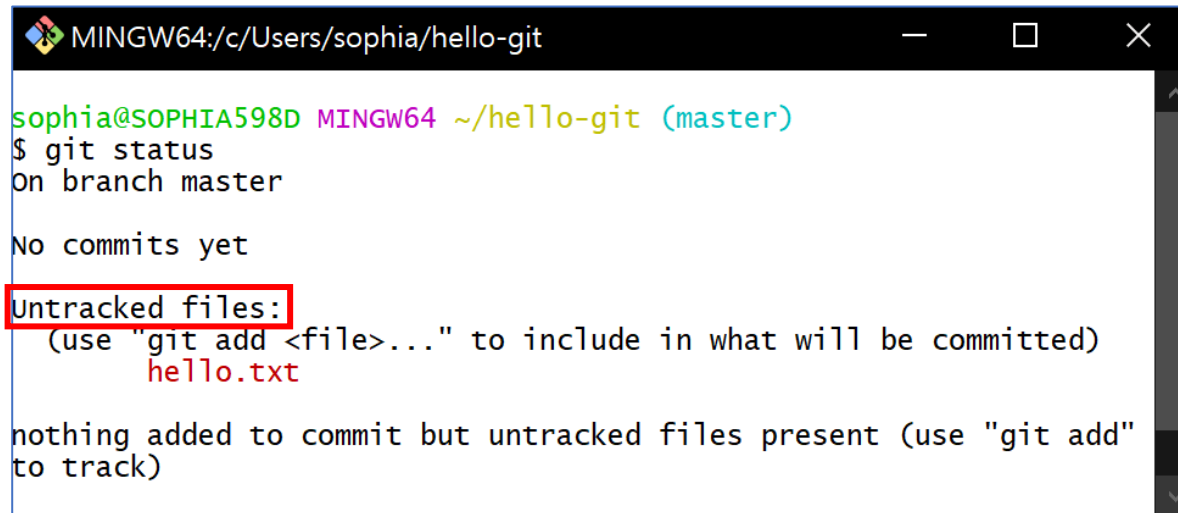


# [실습] 버전 만들기

## ❖ 작업 트리에서 vim으로 문서 수정하기

- ex 모드(Esc)로 변경 후, 문서 저장(:wq)
- 깃 상태 확인

```
$ git status
```



```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git status
on branch master

No commits yet

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

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

# [실습] 버전 만들기

## ❖ Untracked files

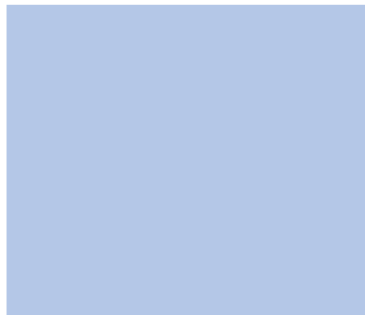
- 깃에서 아직 한번도 관리하지 않은 파일을 의미
- ex) hello.txt

작업트리

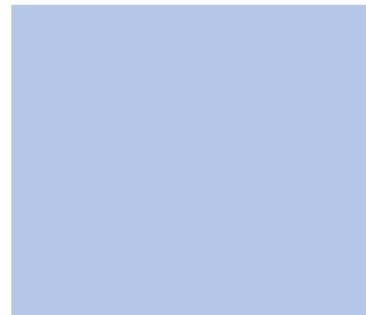


hello.txt

스테이지



저장소



# [실습] 버전 만들기

## ❖ 수정한 파일 스테이징하기



# [실습] 버전 만들기

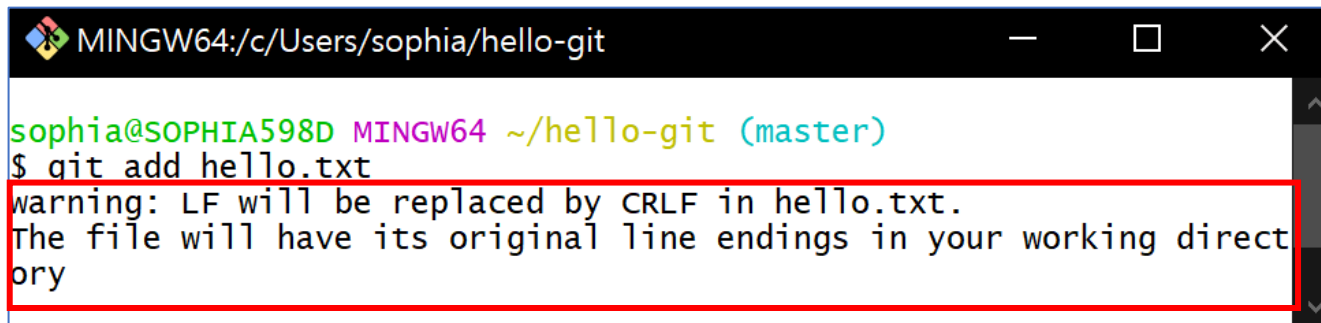
## ❖ 수정한 파일 스테이징하기

- 스테이지에 수정한 파일 추가

```
$ git add 파일명
```

- 예) 스테이지에 hello.txt를 추가

```
$ git add hello.txt
```



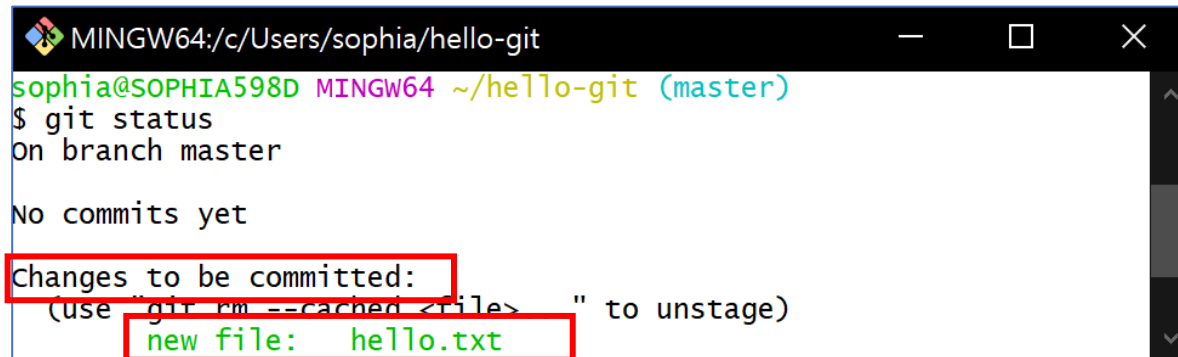
```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git add hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
```

# [실습] 버전 만들기

## ❖ 수정한 파일 스테이징하기

- 깃 상태 확인

```
$ git status
```



```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>" to unstage)
    new file:   hello.txt
```

- 새 파일 hello.txt를 커밋할 예정임을 의미

# [실습] 버전 만들기

## ❖ 스테이지에 올라온 파일 커밋하기



# [실습] 버전 만들기

## ❖ 스테이지에 올라온 파일 커밋하기

- 파일 커밋

```
$ git commit -m "커밋메시지"
```

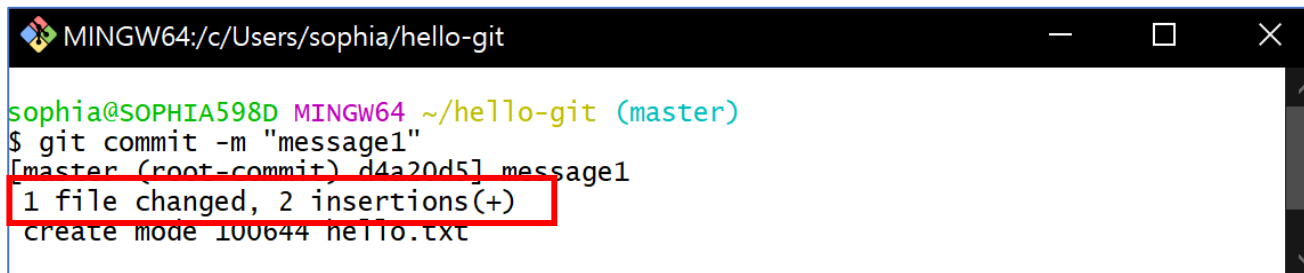
- 커밋 메시지
  - 커밋할 때 해당 버전에 어떤 변경 사항이 있었는지 확인하기 위한 메시지를 기록
  - 영어로 작성하는 것이 좋음

# [실습] 버전 만들기

## ❖ 스테이지에 올라온 파일 커밋하기

- 예) 커밋 메시지에 "message1" 작성

```
$ git commit -m "message1"
```

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/sophia/hello-git'. The prompt is 'sophia@SOPHIA598D MINGW64 ~/hello-git (master)'. The command '\$ git commit -m "message1"' has been entered. The output shows '[master (root-commit) d4a20d5] message1' followed by '1 file changed, 2 insertions(+)' and 'create mode 100644 hello.txt'. The line '1 file changed, 2 insertions(+)' is highlighted with a red rectangle.

```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git commit -m "message1"
[master (root-commit) d4a20d5] message1
1 file changed, 2 insertions(+)
create mode 100644 hello.txt
```

- 파일 1개가 변경되었으며, 파일에 2개의 내용이 추가됨

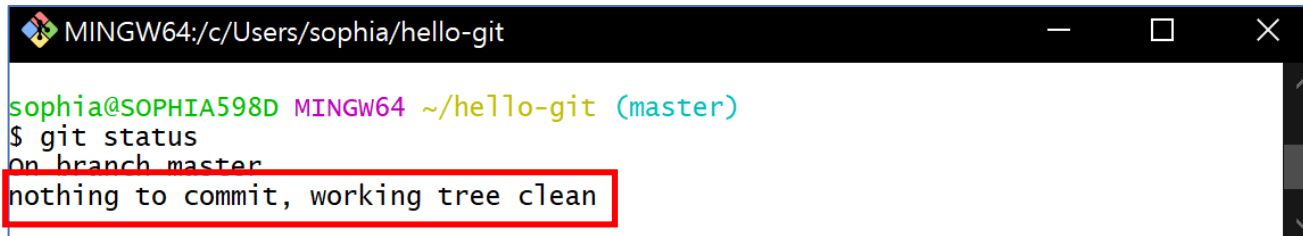


# [실습] 버전 만들기

## ❖ 스테이지에 올라온 파일 커밋하기

- 깃 상태 확인

```
$ git status
```

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/sophia/hello-git'. The prompt is 'sophia@SOPHIA598D MINGW64 ~/hello-git (master)'. The command '\$ git status' has been entered, and the output is 'on branch master' followed by 'nothing to commit, working tree clean' which is highlighted with a red rectangular box.

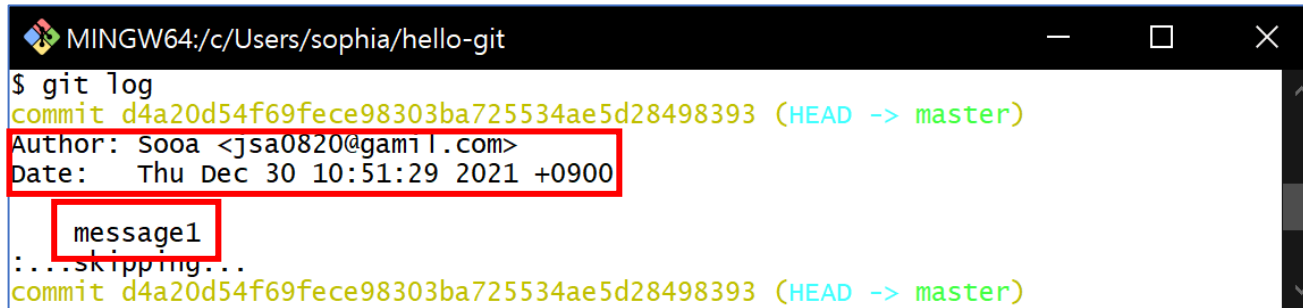
```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git status
on branch master
nothing to commit, working tree clean
```

- nothing to commit
  - 버전으로 만들 파일이 없음
- working tree clean
  - 작업 트리에도 수정사항이 없음

# [실습] 버전 만들기

## ❖ 커밋한 버전 확인하기

```
$ git log
```



```
MINGW64:/c/Users/sophia/hello-git
$ git log
commit d4a20d54f69fece98303ba725534ae5d28498393 (HEAD -> master)
Author: Sooa <jsa0820@gmail.com>
Date: Thu Dec 30 10:51:29 2021 +0900
    message1
...skipping...
commit d4a20d54f69fece98303ba725534ae5d28498393 (HEAD -> master)
```

- Author
  - 커밋을 만든 사람
- Date
  - 커밋 시간 및 커밋 메시지

# [실습] 버전 만들기

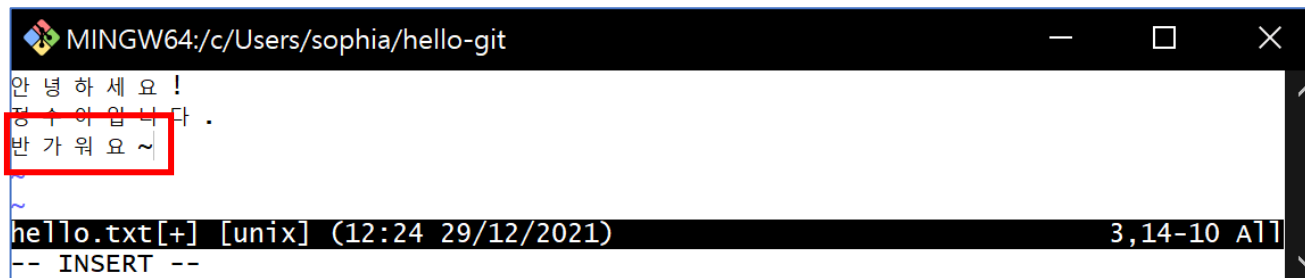
## ❖ 스테이징과 커밋 한번에 처리하기

- 한 번 이상 커밋한 파일을 다시 커밋할 때만 사용 가능

```
$ git commit -am "커밋메시지"
```

- hello.txt 파일 수정

```
$ vim hello.txt
```

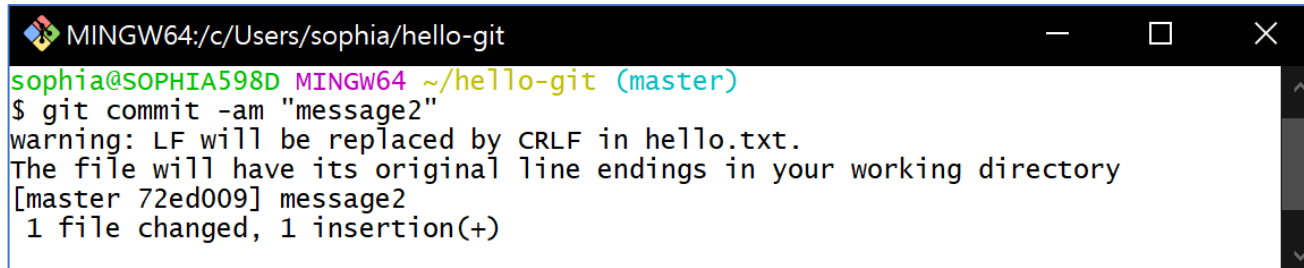


```
MINGW64:/c/Users/sophia/hello-git
안녕하세요!
정수아입니다.
반가워요~
~
hello.txt[+] [unix] (12:24 29/12/2021) 3,14-10 All
-- INSERT --
```

# [실습] 버전 만들기

## ❖ 스테이징과 커밋 한번에 처리하기

```
$ git commit -am "message2"
```

A screenshot of a Windows terminal window with a black title bar and standard window controls. The title bar text is 'MINGW64:/c/Users/sophia/hello-git'. The terminal content shows a user named 'sophia' at a machine 'SOPHIA598D' in a 'MINGW64' environment, located at '~/hello-git' on the 'master' branch. The user enters the command '\$ git commit -am "message2"'. The output shows a warning about line endings (LF vs CRLF) for 'hello.txt', followed by the commit hash '[master 72ed009]', the commit message 'message2', and a summary '1 file changed, 1 insertion(+)'.

# [실습] 버전 만들기

## ❖ 커밋 기록 확인하기

```
$ git log
```

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/sophia/hello-git'. The terminal shows the command 'git log' being executed. The output displays two commit records. The first commit, with hash '72ed009d3469b3a3fc9cea27f590878e7649d84a', is highlighted with a red rectangular box. This commit is labeled '(HEAD -> master)' and was authored by 'Sooa' on 'Mon Jan 3 11:19:37 2022 +0900' with the message 'message2'. The second commit, with hash 'd4a20d54f69fece98303ba725534ae5d28498393', was authored by 'Sooa' on 'Thu Dec 30 10:51:29 2021 +0900' with the message 'message1'.

```
MINGW64:/c/Users/sophia/hello-git
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git log
commit 72ed009d3469b3a3fc9cea27f590878e7649d84a (HEAD -> master)
Author: Sooa <jsa0820@gamil.com>
Date:   Mon Jan 3 11:19:37 2022 +0900

    message2

commit d4a20d54f69fece98303ba725534ae5d28498393
Author: Sooa <jsa0820@gamil.com>
Date:   Thu Dec 30 10:51:29 2021 +0900

    message1
```



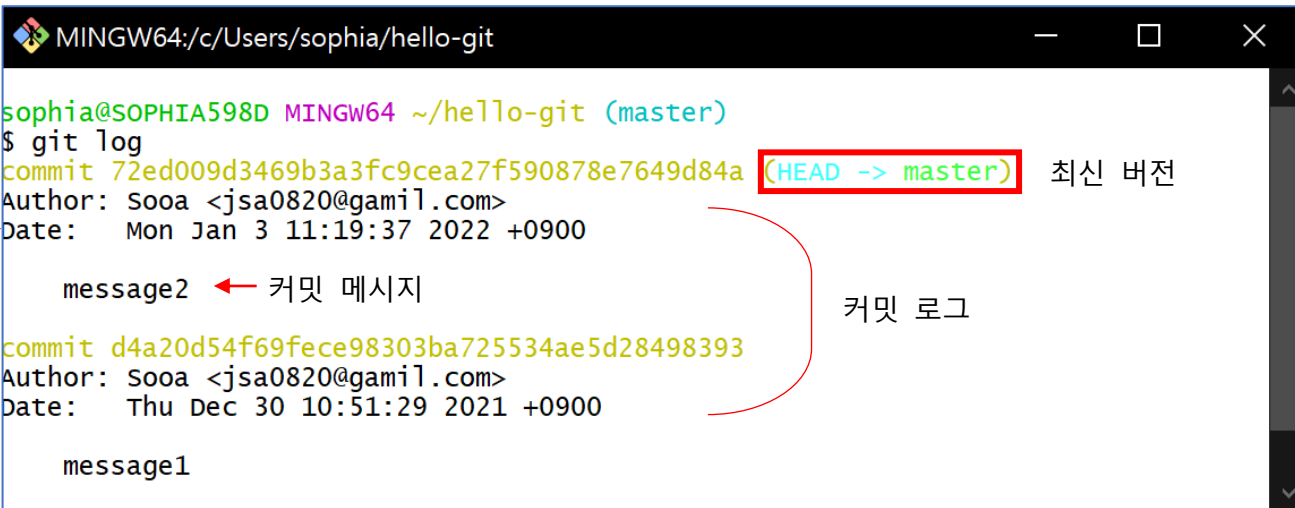
03

## 커밋 내용 확인하기

# 커밋 내용 확인하기

## ❖ 커밋 기록 확인하기

```
$ git log
```



The image shows a Windows terminal window titled "MINGW64:/c/Users/sophia/hello-git". The user is at the prompt "sophia@SOPHIA598D MINGW64 ~/hello-git (master)". They have entered the command "\$ git log". The output shows two commits. The first commit is highlighted with a red box around the text "(HEAD -> master)". To the left of the terminal, there are three lines of Korean text with red arrows pointing to specific parts of the first commit: "커밋 해시" points to the commit hash, "작성자" points to the author, and "버전 생성 날짜" points to the date. To the right of the terminal, the text "최신 버전" is next to the "(HEAD -> master)" text. Below the terminal output, there is a red bracket spanning the width of the commit details, with the text "커밋 로그" to its right. The commit details include the commit hash, author, date, and message.

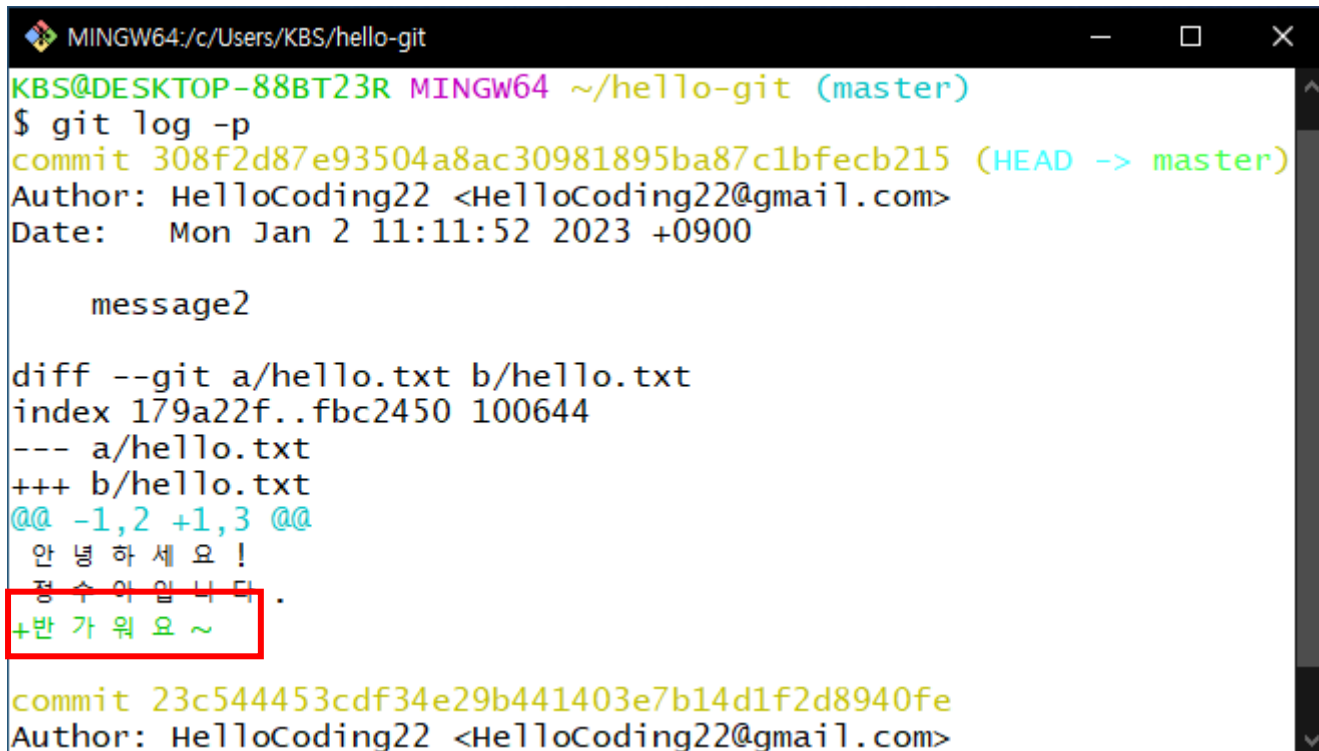
```
sophia@SOPHIA598D MINGW64 ~/hello-git (master)
$ git log
commit 72ed009d3469b3a3fc9cea27f590878e7649d84a (HEAD -> master) 최신 버전
Author: Sooa <jsa0820@gamil.com>
Date: Mon Jan 3 11:19:37 2022 +0900
    message2 ← 커밋 메시지

commit d4a20d54f69fece98303ba725534ae5d28498393
Author: Sooa <jsa0820@gamil.com>
Date: Thu Dec 30 10:51:29 2021 +0900
    message1
```

# 커밋 내용 확인하기

## ❖ 커밋에 포함된 파일의 변경 내용 확인

```
$ git log -p 또는 git log --patch
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log -p
commit 308f2d87e93504a8ac30981895ba87c1bfecb215 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Mon Jan 2 11:11:52 2023 +0900

    message2

diff --git a/hello.txt b/hello.txt
index 179a22f..fbc2450 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,3 @@
안녕하세요!
정수아입니다.
+반가워요~

commit 23c544453cdf34e29b441403e7b14d1f2d8940fe
Author: HelloCoding22 <HelloCoding22@gmail.com>
```

변경 내용



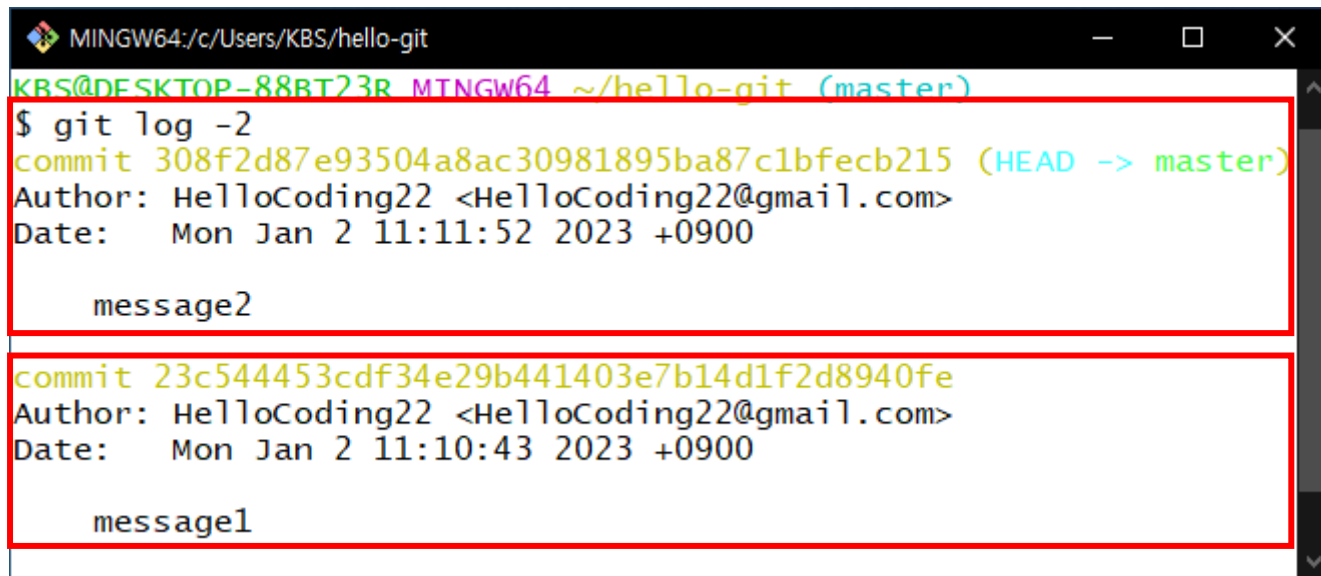
# 커밋 내용 확인하기

## ❖ 최근 몇 개의 커밋을 보여줄지 지정

```
$ git log -숫자
```

- 예) 최근 2개의 커밋을 출력

```
$ git log -2
```

A screenshot of a Windows terminal window titled 'MINGW64: c:/Users/KBS/hello-git'. The prompt is 'KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)'. The command '\$ git log -2' has been executed. The output shows two commits. The first commit, highlighted with a red box, has hash '308f2d87e93504a8ac30981895ba87c1bfecb215', is on the 'HEAD -> master' branch, authored by 'HelloCoding22 <HelloCoding22@gmail.com>' on 'Mon Jan 2 11:11:52 2023 +0900', with the message 'message2'. The second commit, also highlighted with a red box, has hash '23c544453cdf34e29b441403e7b14d1f2d8940fe', is authored by the same person on 'Mon Jan 2 11:10:43 2023 +0900', with the message 'message1'.

```
MINGW64: c:/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log -2
commit 308f2d87e93504a8ac30981895ba87c1bfecb215 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:11:52 2023 +0900

    message2

commit 23c544453cdf34e29b441403e7b14d1f2d8940fe
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:10:43 2023 +0900

    message1
```

# 커밋 내용 확인하기

## ❖ -p 옵션과 -숫자 옵션을 동시에 사용

```
$ git log -p -1
```

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The prompt is 'KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)'. The command '\$ git log -p -1' has been executed. The output shows a commit hash '308f2d87e93504a8ac30981895ba87c1bfecb215' with the message '(HEAD -> master)'. The author is 'HelloCoding22 <HelloCoding22@gmail.com>' and the date is 'Mon Jan 2 11:11:52 2023 +0900'. The commit message is 'message2'. Below this, a diff is shown for 'a/hello.txt' and 'b/hello.txt', with index '179a22f..fbc2450 100644'. The diff shows three lines of text: '안녕하세요!', '정수아입니다.', and '+반가워요~'.

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log -p -1
commit 308f2d87e93504a8ac30981895ba87c1bfecb215 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date:   Mon Jan 2 11:11:52 2023 +0900

    message2

diff --git a/hello.txt b/hello.txt
index 179a22f..fbc2450 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,3 @@
안녕하세요!
정수아입니다.
+반가워요~
```

# 커밋 내용 확인하기

## ❖ 변경 사항 확인하기

```
$ git diff
```

- 예시
  - 작업 트리에 있는 파일과 스테이지에 있는 파일을 비교
  - 스테이지에 있는 파일과 저장소에 있는 최신 커밋을 비교

# [실습] 변경 사항 확인하기

## ❖ hello.txt 파일 수정하기

```
MINGW64:/c/Users/KBS/hello-git
안녕하세요!
정수아입니다.
만가워요~
~
hello.txt [unix] (10:48 04/01/2022) 3,14-10 All
-- INSERT --
```

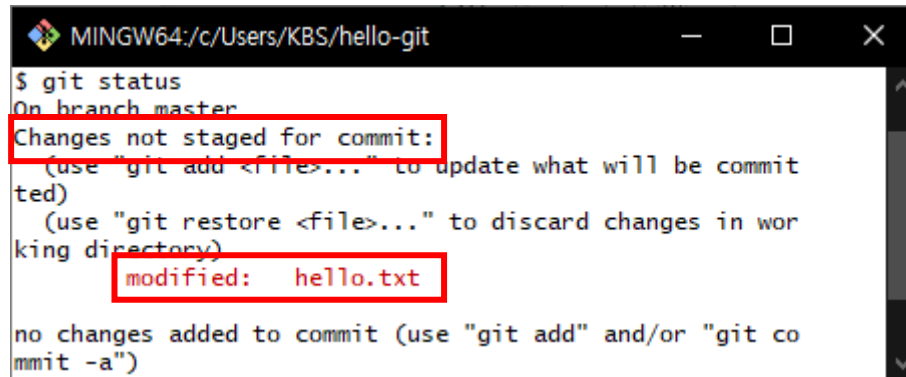
  

```
MINGW64:/c/Users/KBS/hello-git
안녕하세요!
git 수업중입니다.
만가워요~
~
hello.txt[+] [unix] (10:48 04/01/2022) 2,24-18 All
-- INSERT --
```

# [실습] 변경 사항 확인하기

## ❖ 깃 상태 확인하기

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be commit
ted)
  (use "git restore <file>..." to discard changes in wor
king directory)
        modified:   hello.txt

no changes added to commit (use "git add" and/or "git co
mmit -a")
```

- 아직 스테이징 상태가 아님
- hello.txt 파일이 수정되었음

# [실습] 변경 사항 확인하기

## ❖ 변경 사항 확인하기

- 방금 수정한 파일과 저장소에 있는 최신 버전의 파일 비교

```
$ git diff
```

```
MINGW64:/c/Users/KBS/hello-git
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
diff --git a/hello.txt b/hello.txt
index fbc2450..09cfa1c 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,3 +1,3 @@
-안녕하세요!
-정수 아닙니다.
+git 수업 중입니다.
+반가워요~
~
(END)
```

- ① - 최신 버전과 비교 시, 삭제 되었음을 의미
- ② + 최신 버전과 비교 시, 추가 되었음을 의미



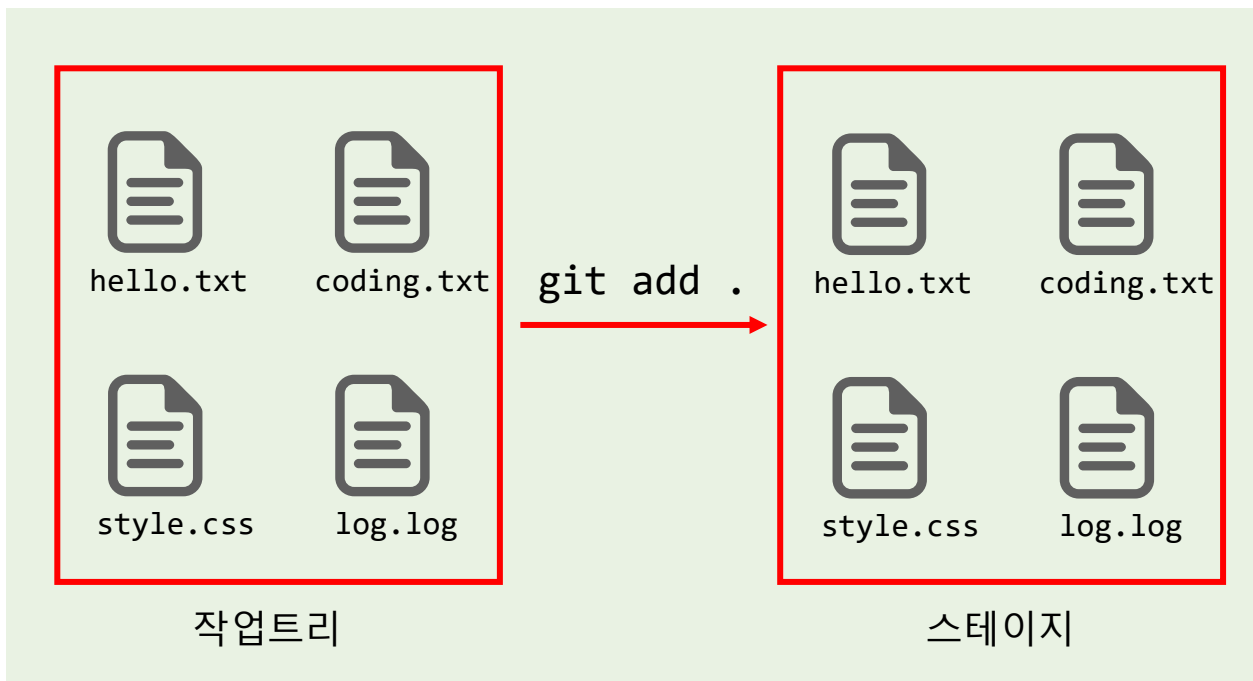
04

**.gitignore**

# .gitignore

## ❖ .gitignore 파일

- git에서 관리할 필요가 없는 파일 또는 디렉터리를 작성
- 작성한 파일 또는 디렉터리는 git add 실행 시, 스테이징 안됨

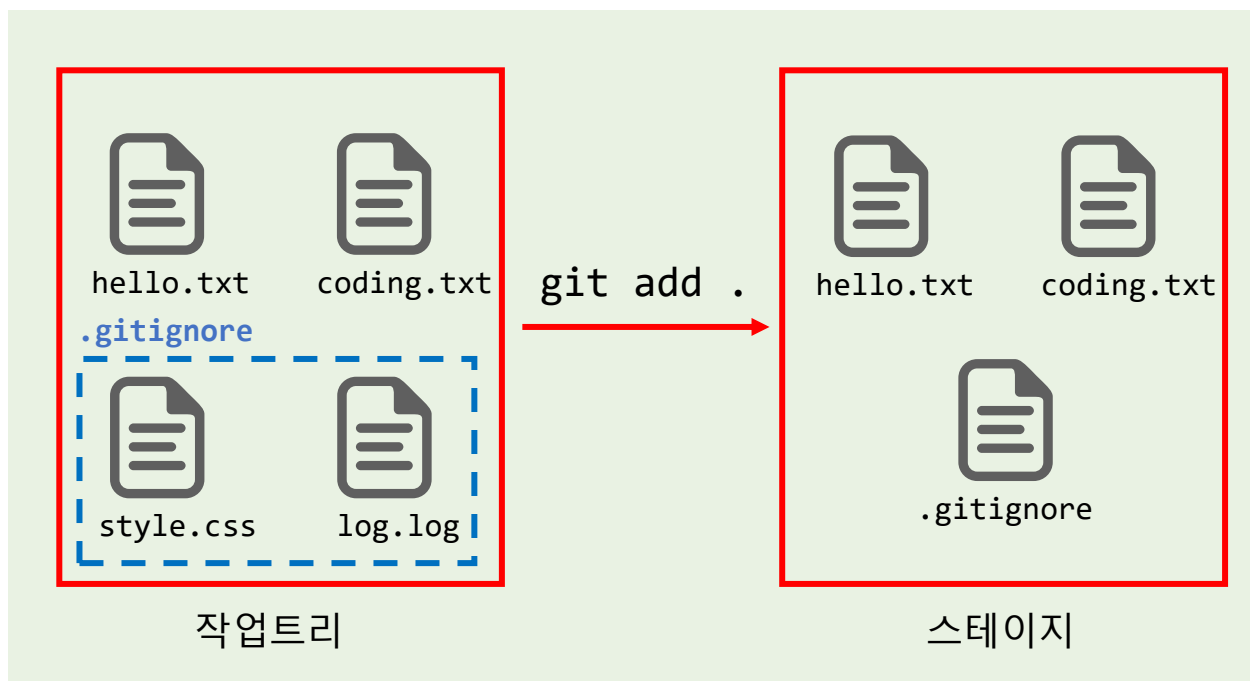




# .gitignore

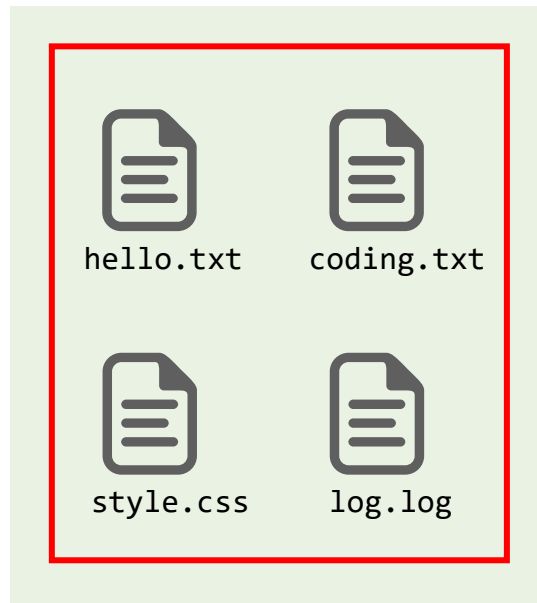
## ❖ .gitignore 파일

- git에서 관리할 필요가 없는 파일 또는 디렉터리를 작성
- 작성한 파일 또는 디렉터리는 git add 실행 시, 스테이징 안됨



# [실습] .gitignore

## ❖ 작업트리에 파일 생성

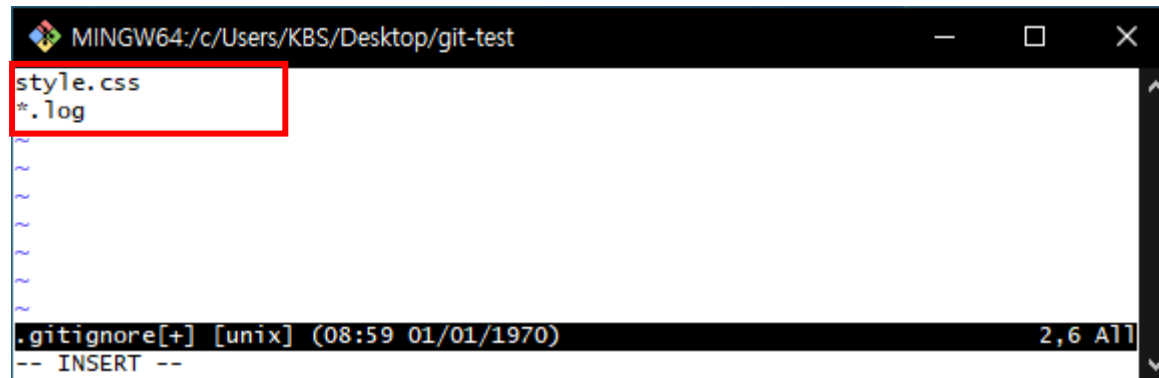


# [실습] .gitignore

## ❖ .gitignore 파일 생성

```
$ vim .gitignore
```

## ❖ 제외할 파일 또는 디렉터리 작성



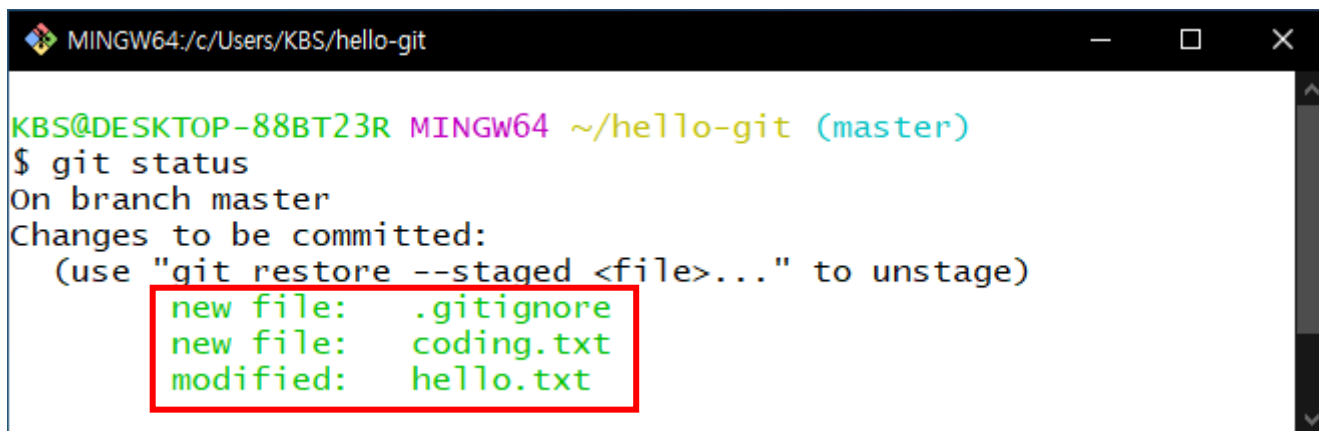
# [실습] .gitignore

## ❖ 작업트리에 있는 전체 파일을 스테이지에 추가

```
$ git add .
```

## ❖ 깃 상태 확인하기

```
$ git status
```



```
MINGW64: /c/Users/KBS/hello-git

KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   coding.txt
    modified:   hello.txt
```

## ❖ 어떤 파일을 제외할까?

- 보안상 위험성이 있는 파일
- 프로젝트와 관련 없는 파일
- 용량이 너무 커서 제외해야 하는 파일

## ❖ .gitignore 파일 작성 방법

- 특정 파일 제외

```
style.css
```

- 현재 경로에 있는 특정 파일만 제외

```
/style.css
```

- 특정 디렉터리 안의 모든 파일 제외

```
folder/
```

# .gitignore

## ❖ .gitignore 파일 작성 방법

- 특정 디렉터리의 특정 파일 제외

```
folder/style.css
```

- 특정 확장자 파일 모두 제외

```
*.css
```



05

## 단계마다 파일 상태 알아보기



# tracked 파일 vs untracked 파일

## ❖ tracked 파일

- 깃이 상태를 계속 추적하고 있는 파일
- hello.txt 파일 수정

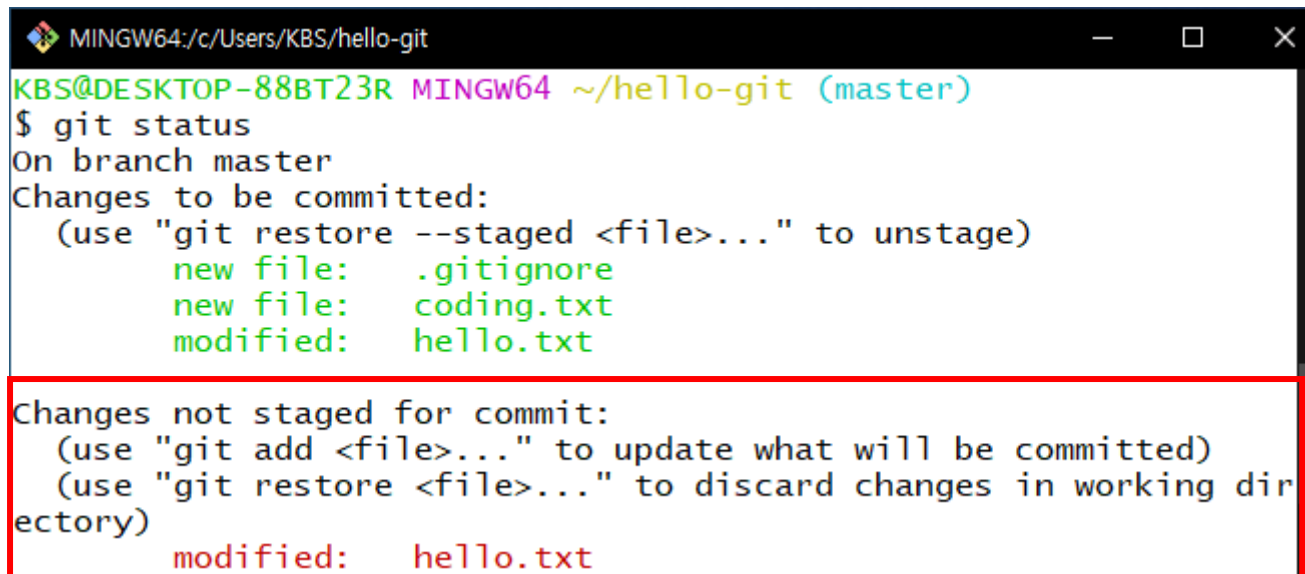
안녕하세요  
반갑습니다

hello1.txt[+] [unix] (13:02 30/06/2022) 2,16-11 All  
-- INSERT --

# tracked 파일 vs untracked 파일

## ❖ tracked 파일

```
$ git status
```

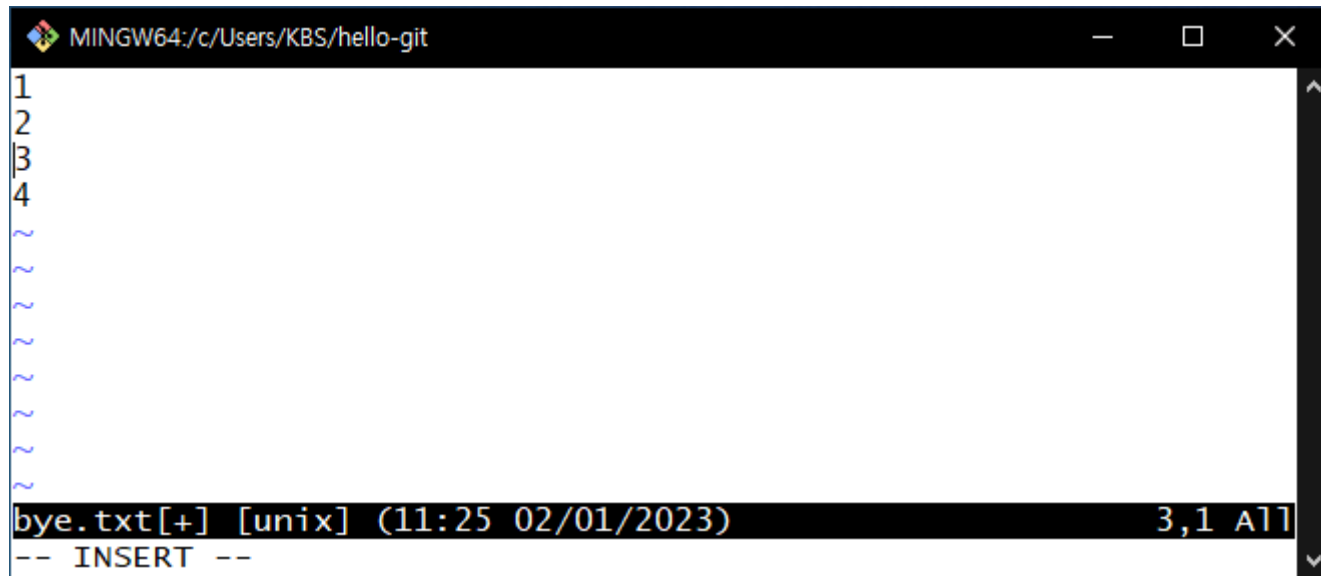
A terminal window titled 'MINGW64:/c/Users/KBS/hello-git' showing the output of the 'git status' command. The output is as follows:  
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)  
\$ git status  
On branch master  
Changes to be committed:  
 (use "git restore --staged <file>..." to unstage)  
 new file: .gitignore  
 new file: coding.txt  
 modified: hello.txt  
  
Changes not staged for commit:  
 (use "git add <file>..." to update what will be committed)  
 (use "git restore <file>..." to discard changes in working directory)  
 modified: hello.txt  
The bottom section of the output, 'Changes not staged for commit:', is highlighted with a red rectangular box.

# tracked 파일 vs untracked 파일

## ❖ untracked 파일

- 깃이 상태를 추적하지 않는 파일

```
$ vim bye.txt
```

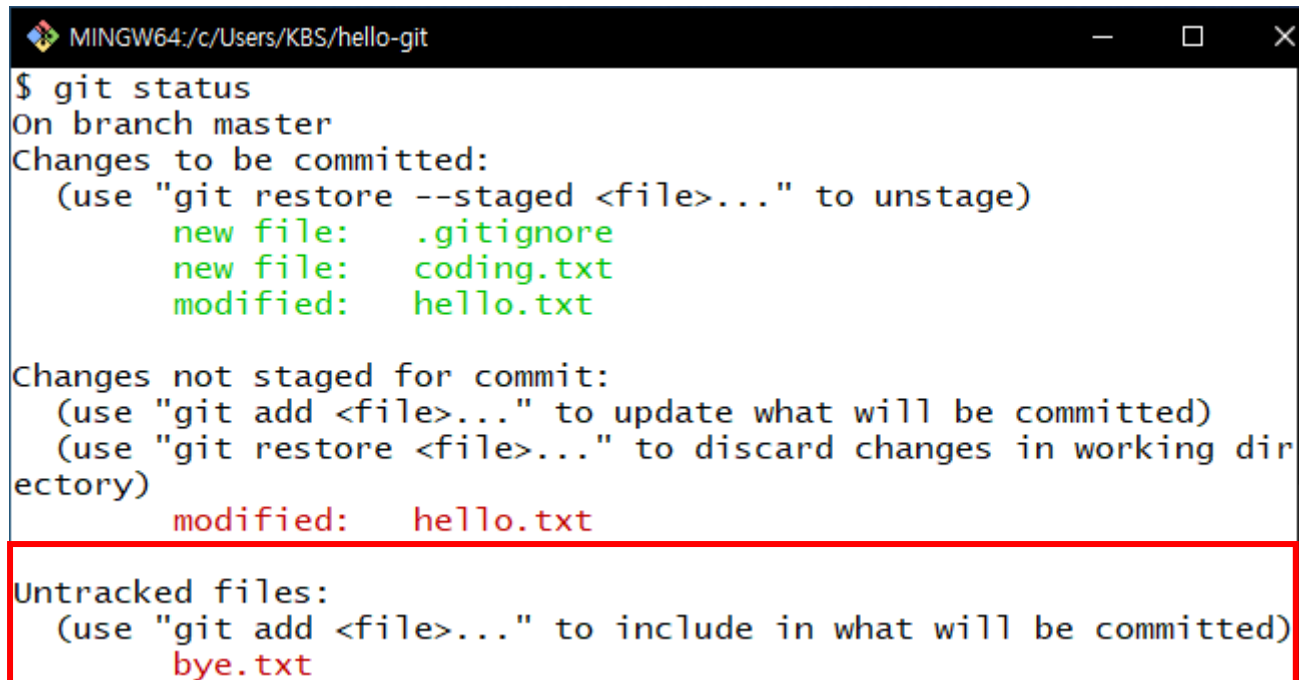


The screenshot shows a terminal window with the title bar "MINGW64:/c/Users/KBS/hello-git". The Vim editor is open, displaying a file named "bye.txt". The editor shows line numbers 1 through 10 on the left margin. The first four lines contain the text "1", "2", "3", and "4" respectively. Lines 5 through 10 are marked with blue tilde symbols (~). The status bar at the bottom of the editor displays "bye.txt[+] [unix] (11:25 02/01/2023) 3,1 A11" and "-- INSERT --".

# tracked 파일 vs untracked 파일

## ❖ untracked 파일

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   coding.txt
    modified:   hello.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   hello.txt

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

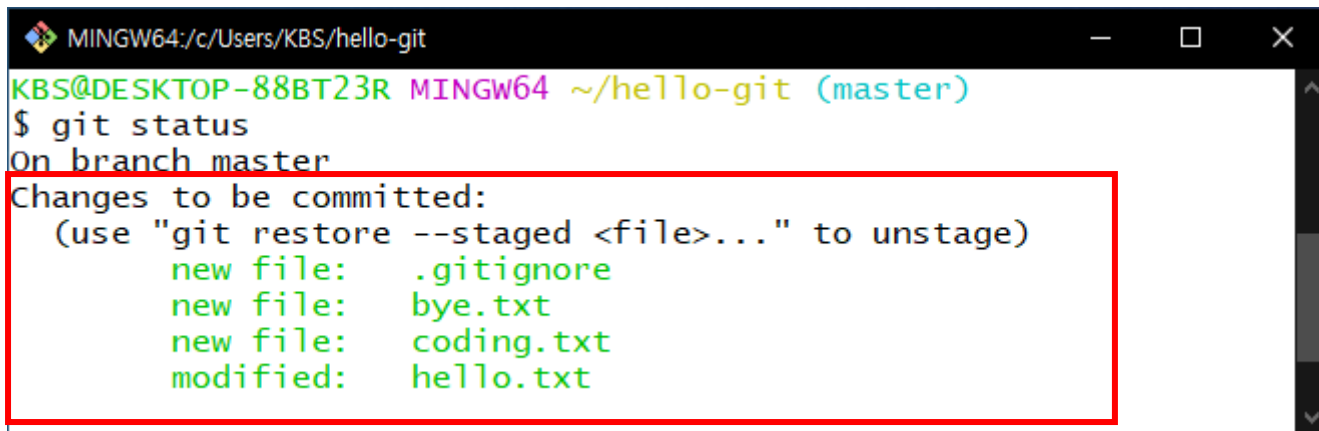
# tracked 파일 vs untracked 파일

## ❖ 스테이지에 추가하기

```
$ git add .
```

## ❖ 깃 상태 확인하기

```
$ git status
```

A screenshot of a Windows command prompt window titled 'MINGW64: c:/Users/KBS/hello-git'. The prompt shows the user is in the directory '~/hello-git' on the 'master' branch. The command '\$ git status' has been executed, and the output is displayed. A red rectangular box highlights the section 'Changes to be committed:' and its contents. The output shows four files: '.gitignore', 'bye.txt', 'coding.txt', and 'hello.txt'. The first three are listed as 'new file:' and the last one as 'modified:'.

```
MINGW64: c:/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   bye.txt
    new file:   coding.txt
    modified:   hello.txt
```

# tracked 파일 vs untracked 파일

## ❖ 스테이지에 올라온 파일 커밋하기

```
$ git commit -m "message3"
```

## ❖ 커밋 기록 확인하기

```
$ git log
```

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The terminal shows the output of a 'git log' command. The first line is 'commit 6939d47c67d59e71075e9df440be74ab9c8c8a6c (HEAD -> master)'. Below this, the commit details are shown: 'Author: Sooa <jsa0820@gmail.com>' and 'Date: Tue Jan 4 12:00:16 2022 +0900'. The commit message 'message3' is also visible. A red rectangular box highlights the author and date information.

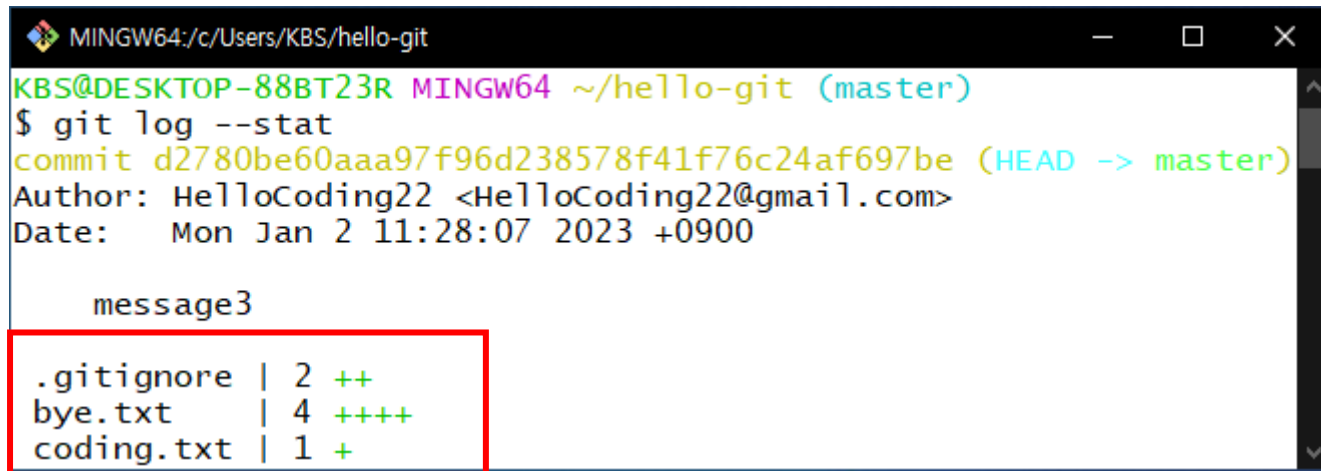
```
commit 6939d47c67d59e71075e9df440be74ab9c8c8a6c (HEAD -> master)
Author: Sooa <jsa0820@gmail.com>
Date: Tue Jan 4 12:00:16 2022 +0900

    message3
```

# tracked 파일 vs untracked 파일

## ❖ 커밋의 통계 정보 확인하기

```
$ git log --stat
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log --stat
commit d2780be60aaa97f96d238578f41f76c24af697be (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Mon Jan 2 11:28:07 2023 +0900

    message3

.gitignore | 2 ++
bye.txt    | 4 ++++
coding.txt | 1 +
```



06

## 작업 되돌리기



# 작업 되돌리기

## ❖ 작업 트리에서 수정한 파일 되돌리기

```
$ git restore 파일명
```

- 파일을 수정한 후, 소스가 정상적으로 작동하지 않는 등의 이유로 수정한 내용을 취소하고 가장 최신 버전으로 되돌려야 하는 경우에 사용
- restore 명령어로 되돌린 내용은 다시 복구할 수 없음

# [실습] 작업 트리에서 수정한 파일 되돌리기

## ❖ bye.txt 파일 수정하기

```
$ vim bye.txt
```

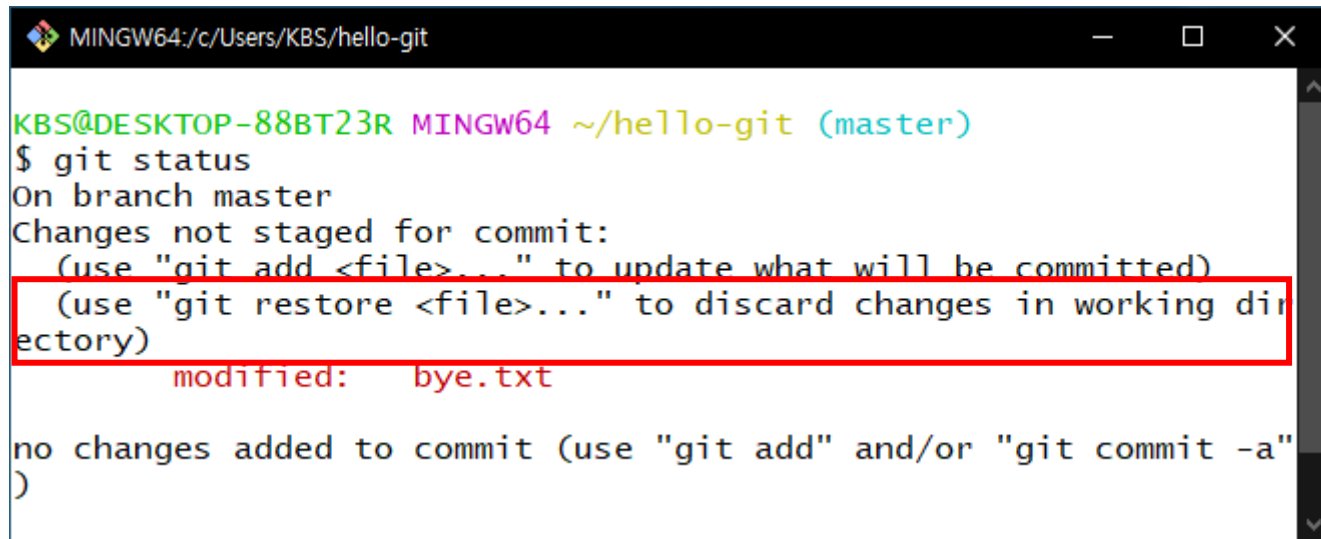


```
MINGW64:/c/Users/KBS/hello-git
1
2
three
4|
~
~
~
~
bye.txt[+] [unix] (11:26 02/01/2023) 4,2 A11
-- INSERT --
```

# [실습] 작업 트리에서 수정한 파일 되돌리기

## ❖ 깃 상태 확인하기

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git

KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   bye.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

# [실습] 작업 트리에서 수정한 파일 되돌리기

## ❖ 작업 트리에서 수정한 파일 되돌리기

```
$ git restore bye.txt
```

## ❖ bye.txt 파일 확인

A terminal window titled 'MINGW64:/c/Users/KBS/hello-git' showing the command 'cat bye.txt' and its output. The output consists of four lines: '1', '2', '3', and '4'. The line '3' is highlighted with a red rectangular box.

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ cat bye.txt
1
2
3
4
```

# 작업 되돌리기

## ❖ 스테이징 취소하기

```
$ git restore --staged 파일명
```

- 수정한 파일을 스테이징했을 때, 스테이징을 취소해야 하는 경우에 사용

# [실습] 스테이징 취소하기

## ❖ bye.txt 파일 수정하기

```
$ vim bye.txt
```



The screenshot shows a terminal window with the title bar "MINGW64:/c/Users/KBS/hello-git". The Vim editor is open, displaying the file "bye.txt". The content of the file is as follows:

```
a  
b  
c  
d  
~  
~  
~  
~
```

The status bar at the bottom of the editor shows "bye.txt[+] [dos] (09:35 03/01/2023) 4,2 A11" and "-- INSERT --", indicating that the editor is in insert mode at line 4, column 2.

# [실습] 스테이징 취소하기

## ❖ 스테이지에 추가하기

```
$ git add bye.txt
```

## ❖ 깃 상태 확인하기

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git

KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   bye.txt
```

# [실습] 스테이징 취소하기

작업트리



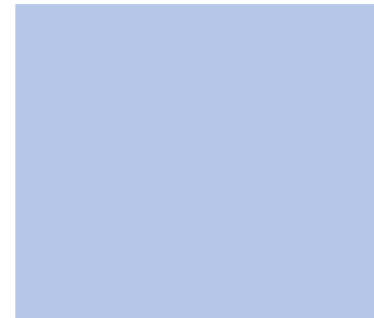
bye.txt

스테이지



bye.txt

저장소





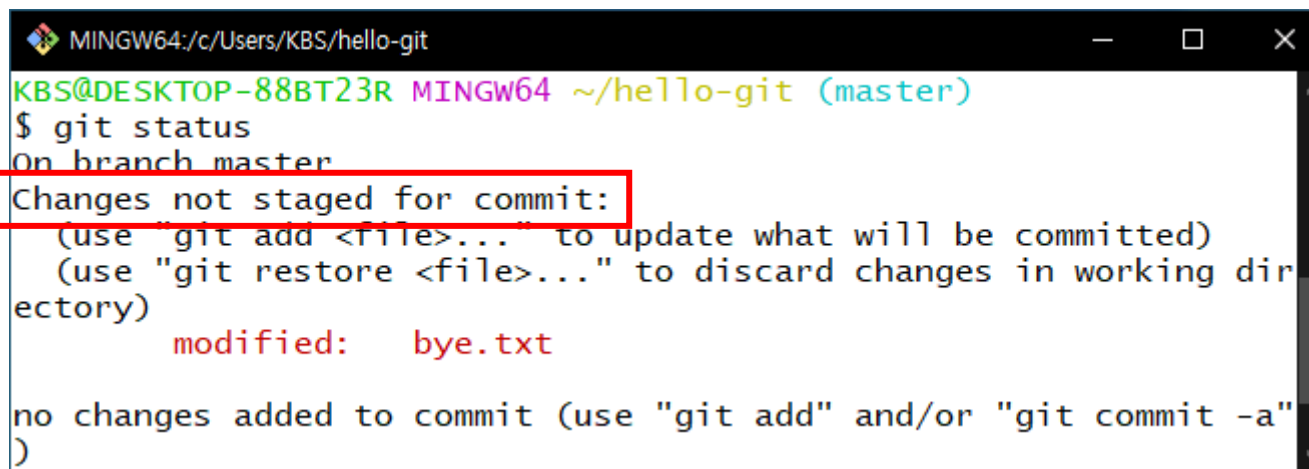
# [실습] 스테이징 취소하기

## ❖ 스테이징 취소하기

```
$ git restore --staged bye.txt
```

## ❖ 깃 상태 확인하기

```
$ git status
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   bye.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

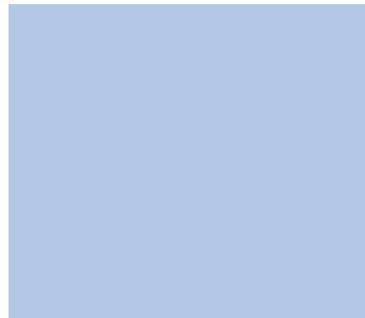
# [실습] 스테이징 취소하기

작업트리

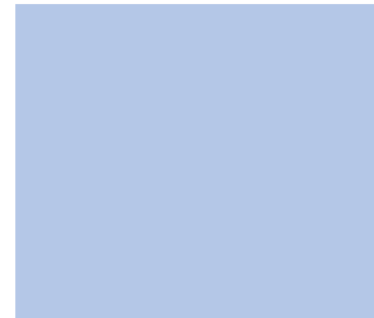


bye.txt

스테이지



저장소



# 작업 되돌리기

## ❖ 최신 커밋 취소하기

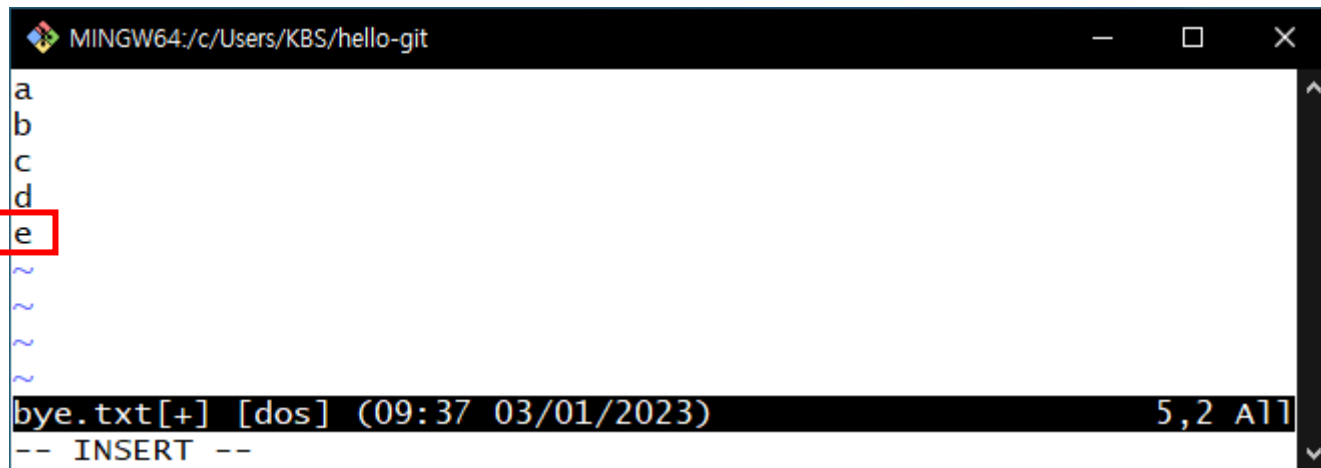
```
$ git reset HEAD^
```

- 수정한 파일을 스테이징하고 커밋까지 했을 때, 가장 마지막에 한 커밋을 취소해야 하는 경우에 사용

# [실습] 최신 커밋 취소하기

## ❖ bye.txt 파일 수정하기

```
$ vim bye.txt
```

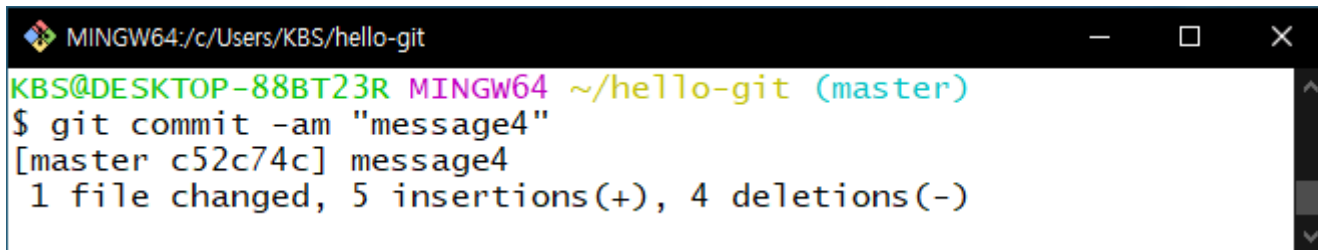


```
MINGW64:/c/Users/KBS/hello-git
a
b
c
d
e
~
~
~
~
bye.txt[+] [dos] (09:37 03/01/2023) 5,2 A11
-- INSERT --
```

# [실습] 최신 커밋 취소하기

## ❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "message4"
```

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/KBS/hello-git'. The prompt is 'KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)'. The command '\$ git commit -am "message4"' has been entered and executed. The output shows '[master c52c74c] message4' followed by '1 file changed, 5 insertions(+), 4 deletions(-)'.

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "message4"
[master c52c74c] message4
1 file changed, 5 insertions(+), 4 deletions(-)
```

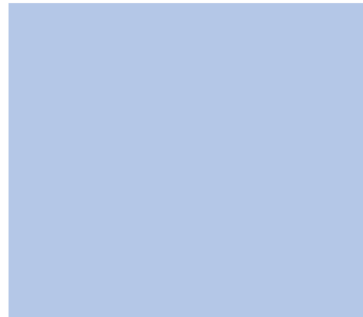
# [실습] 최신 커밋 취소하기

작업트리



bye.txt

스테이지



저장소

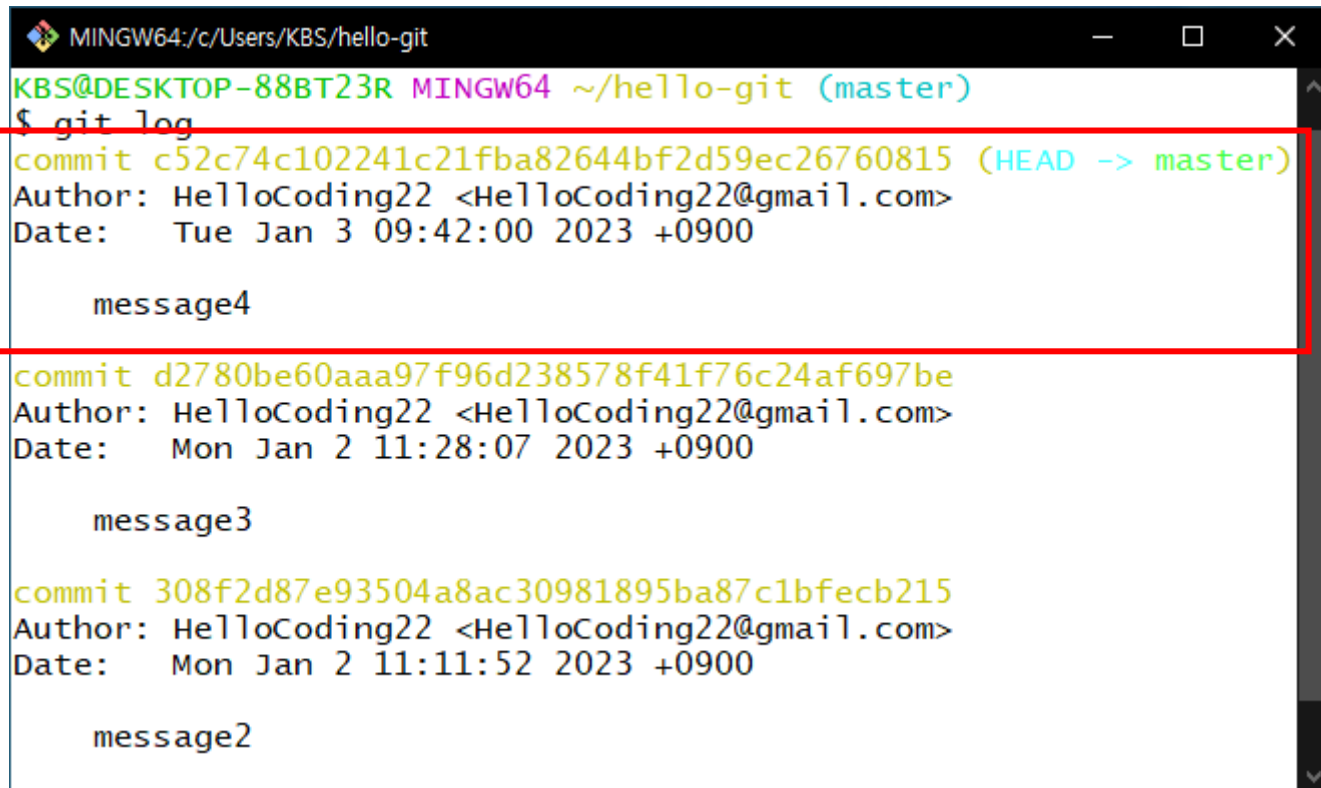


bye.txt

# [실습] 최신 커밋 취소하기

## ❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit c52c74c102241c21fba82644bf2d59ec26760815 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 09:42:00 2023 +0900

    message4

commit d2780be60aaa97f96d238578f41f76c24af697be
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Mon Jan 2 11:28:07 2023 +0900

    message3

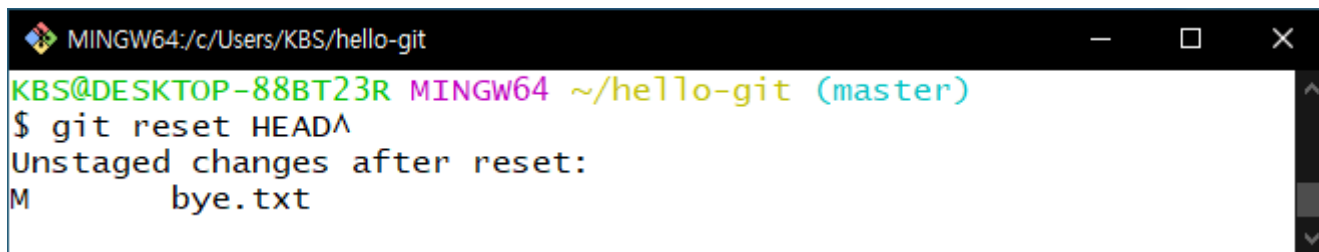
commit 308f2d87e93504a8ac30981895ba87c1bfecb215
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Mon Jan 2 11:11:52 2023 +0900

    message2
```

# [실습] 최신 커밋 취소하기

## ❖ 최신 커밋 취소하기

```
$ git reset HEAD^
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git reset HEAD^
Unstaged changes after reset:
M      bye.txt
```

- 최신 커밋이 취소되고, 스테이지에서도 내려감
- 취소한 파일은 작업 트리에 남아 있음



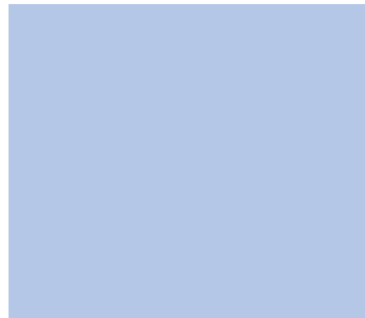
# [실습] 최신 커밋 취소하기

작업트리

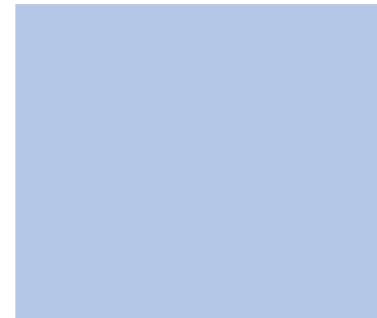


bye.txt

스테이지



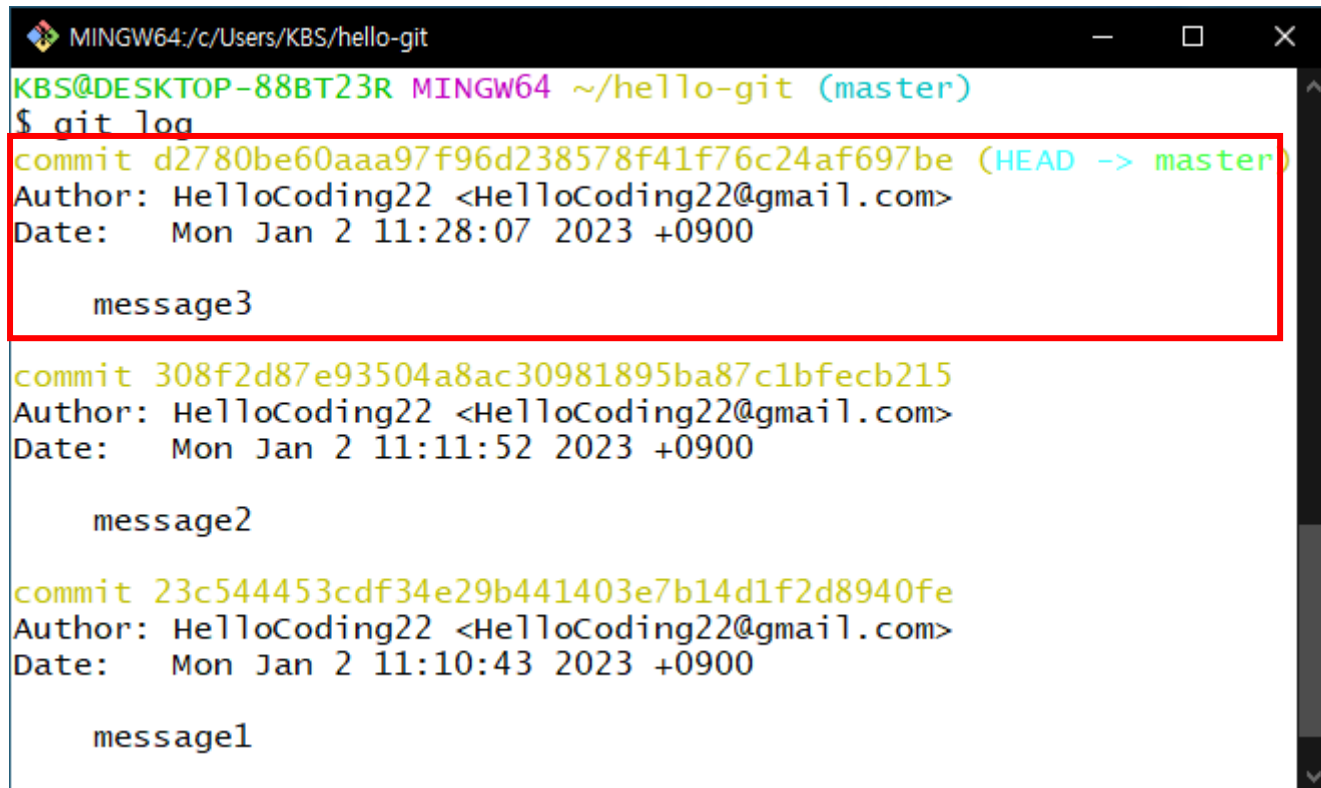
저장소



# [실습] 최신 커밋 취소하기

## ❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit d2780be60aaa97f96d238578f41f76c24af697be (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Mon Jan 2 11:28:07 2023 +0900

    message3

commit 308f2d87e93504a8ac30981895ba87c1bfecb215
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Mon Jan 2 11:11:52 2023 +0900

    message2

commit 23c544453cdf34e29b441403e7b14d1f2d8940fe
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Mon Jan 2 11:10:43 2023 +0900

    message1
```



07

**특정 커밋으로 되돌리기**

# 특정 커밋으로 되돌리기(reset)

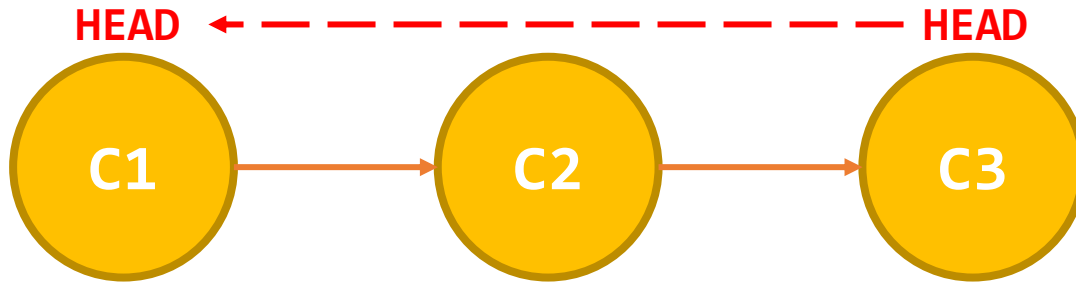
## ❖ 특정 커밋으로 되돌리기

```
$ git reset --옵션 되돌아갈_커밋해시
```

옵션	설명
--hard	특정 버전의 커밋으로 되돌린 다음 그 이후 버전을 삭제
--soft	커밋 이력은 삭제되지만 변경 내용은 스테이지 상태로 남아있음
--mixed	커밋 이력은 삭제되지만 변경 내용은 작업트리에 남아있음

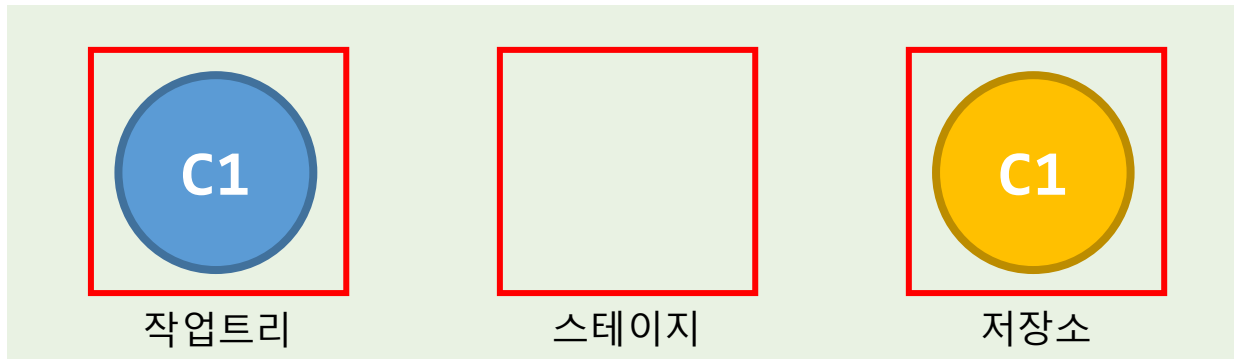
# 특정 커밋으로 되돌리기(reset)

## ❖ --hard 옵션



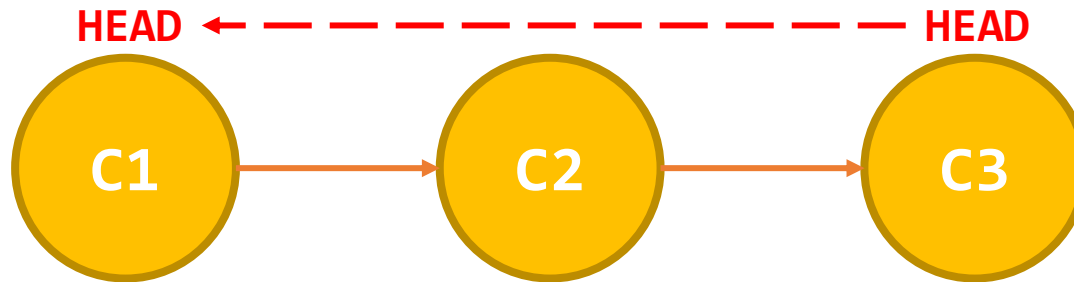
```
$ git reset --hard C1 커밋해시
```

## • 실행 결과



# 특정 커밋으로 되돌리기(reset)

## ❖ --soft 옵션



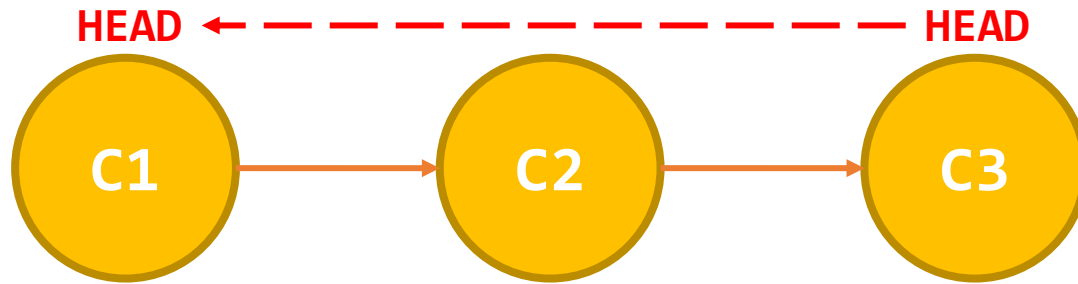
```
$ git reset --soft C1커밋해시
```

## • 실행 결과



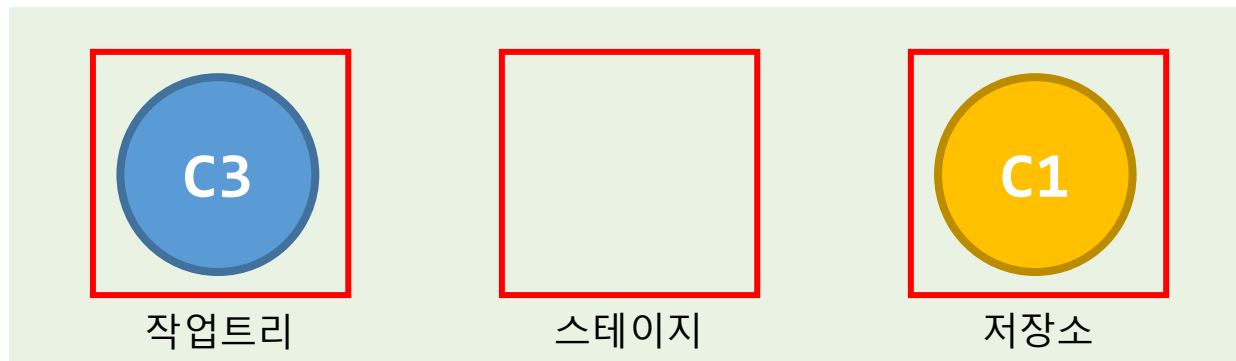
# 특정 커밋으로 되돌리기(reset)

## ❖ --mixed 옵션



```
$ git reset --mixed C1 커밋해시
```

## • 실행 결과



# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ 새로운 파일 생성 및 내용 입력(a)

```
$ vim test.txt
```



## ❖ 스테이지에 추가 및 커밋 하기

```
$ git add test.txt
```

```
$ git commit -m "test commit 1"
```



# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ test.txt 파일에 내용 추가하기(b)

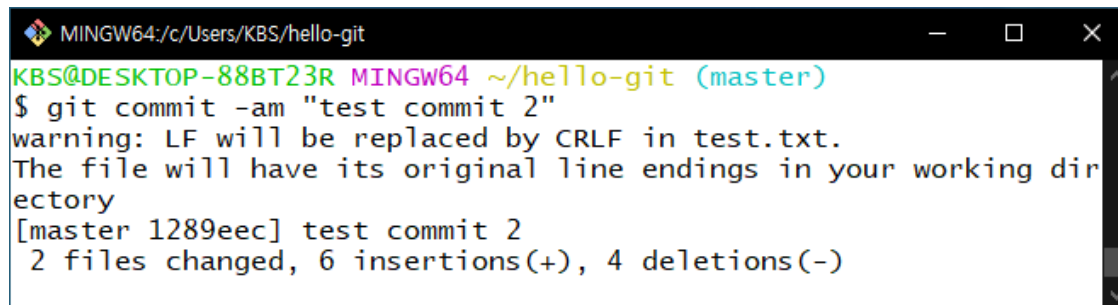
```
$ vim test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
a
b
~
~
~
~
test.txt[+] [unix] (09:59 03/01/2023) 2,2 A11
-- INSERT --
```

## ❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "test commit 2"
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "test commit 2"
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory
[master 1289eec] test commit 2
2 files changed, 6 insertions(+), 4 deletions(-)
```

# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ test.txt 파일에 내용 추가하기(c)

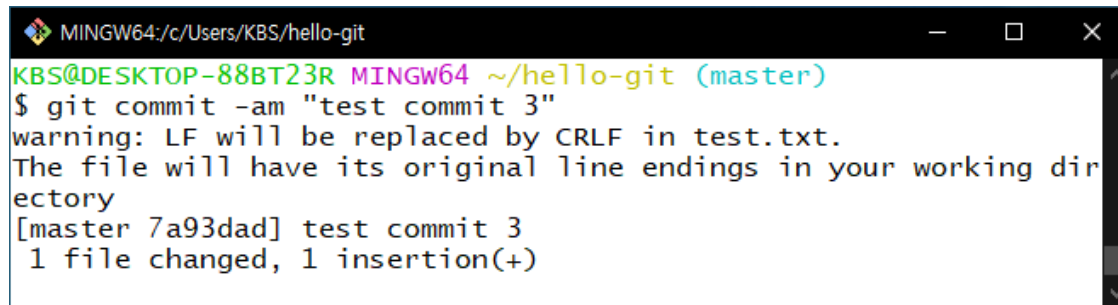
```
$ vim test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
a
b
c
~
~
~
test.txt[+] [unix] (10:02 03/01/2023) 3,2 All
-- INSERT --
```

## ❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "test commit 3"
```

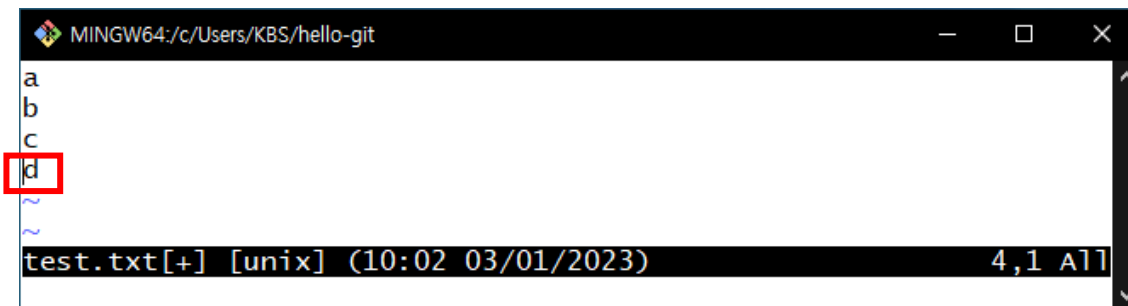


```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "test commit 3"
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory
[master 7a93dad] test commit 3
1 file changed, 1 insertion(+)
```

# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ test.txt 파일에 내용 추가하기(d)

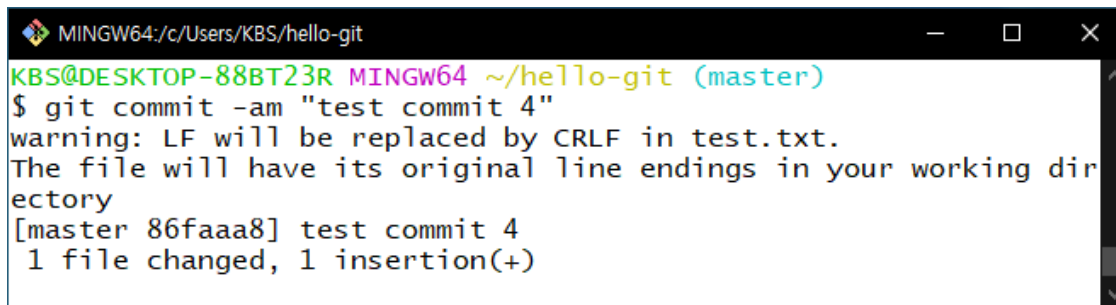
```
$ vim test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
a
b
c
d
~
~
test.txt[+] [unix] (10:02 03/01/2023) 4,1 All
```

## ❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "test commit 4"
```

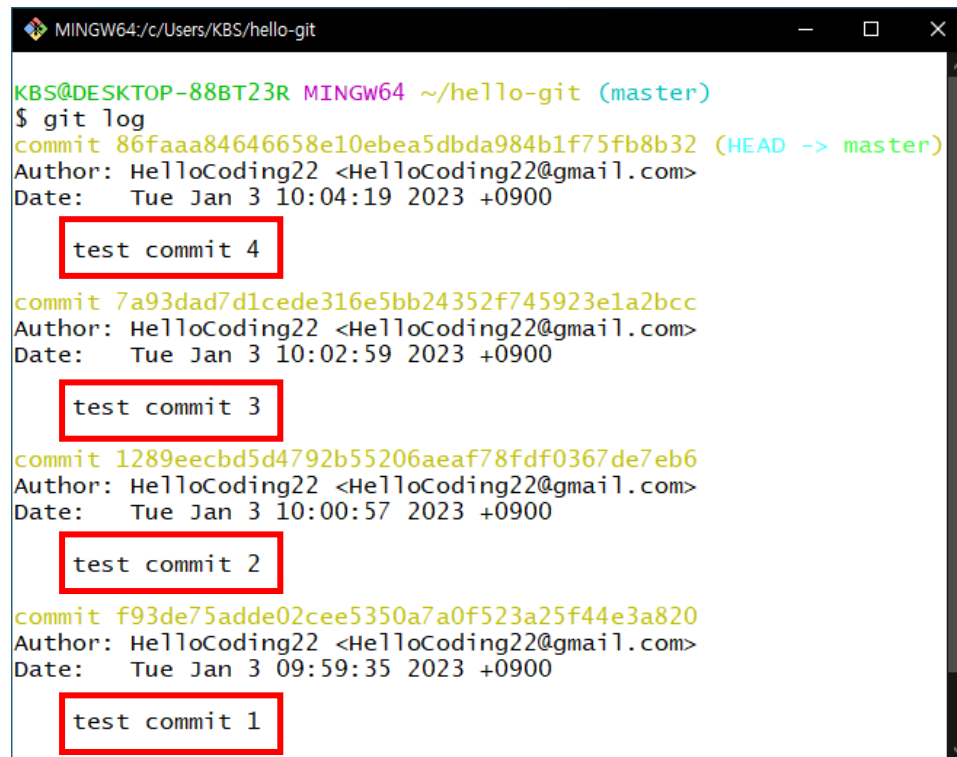


```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "test commit 4"
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory
[master 86faaa8] test commit 4
1 file changed, 1 insertion(+)
```

# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64: c:/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit 86faaa84646658e10e5dbda984b1f75fb8b32 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:04:19 2023 +0900
    test commit 4

commit 7a93dad7d1cede316e5bb24352f745923e1a2bcc
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:02:59 2023 +0900
    test commit 3

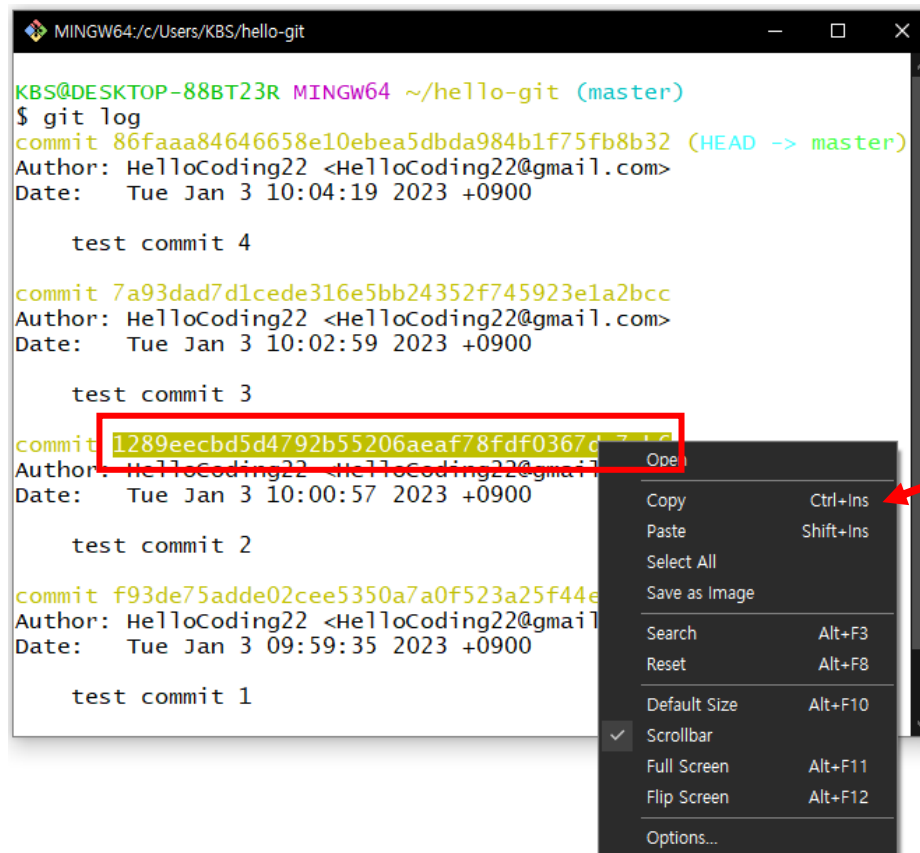
commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:00:57 2023 +0900
    test commit 2

commit f93de75adde02cee5350a7a0f523a25f44e3a820
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 09:59:35 2023 +0900
    test commit 1
```

# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ "test commit 2"를 최신 커밋으로 만들기

- "test commit 2" 커밋 해시를 복사



```
MINGW64:/c/Users/KBS/hello-git

KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit 86faaa84646658e10e5dbda984b1f75fb8b32 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:04:19 2023 +0900

    test commit 4

commit 7a93dad7d1cede316e5bb24352f745923e1a2bcc
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:02:59 2023 +0900

    test commit 3

commit 1289eecbd5d4792b55206aeaf78fdf0367d
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:00:57 2023 +0900

    test commit 2

commit f93de75adde02cee5350a7a0f523a25f44e
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 09:59:35 2023 +0900

    test commit 1
```

# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ "test commit 2"를 최신 커밋으로 만들기

```
$ git reset --hard 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
```



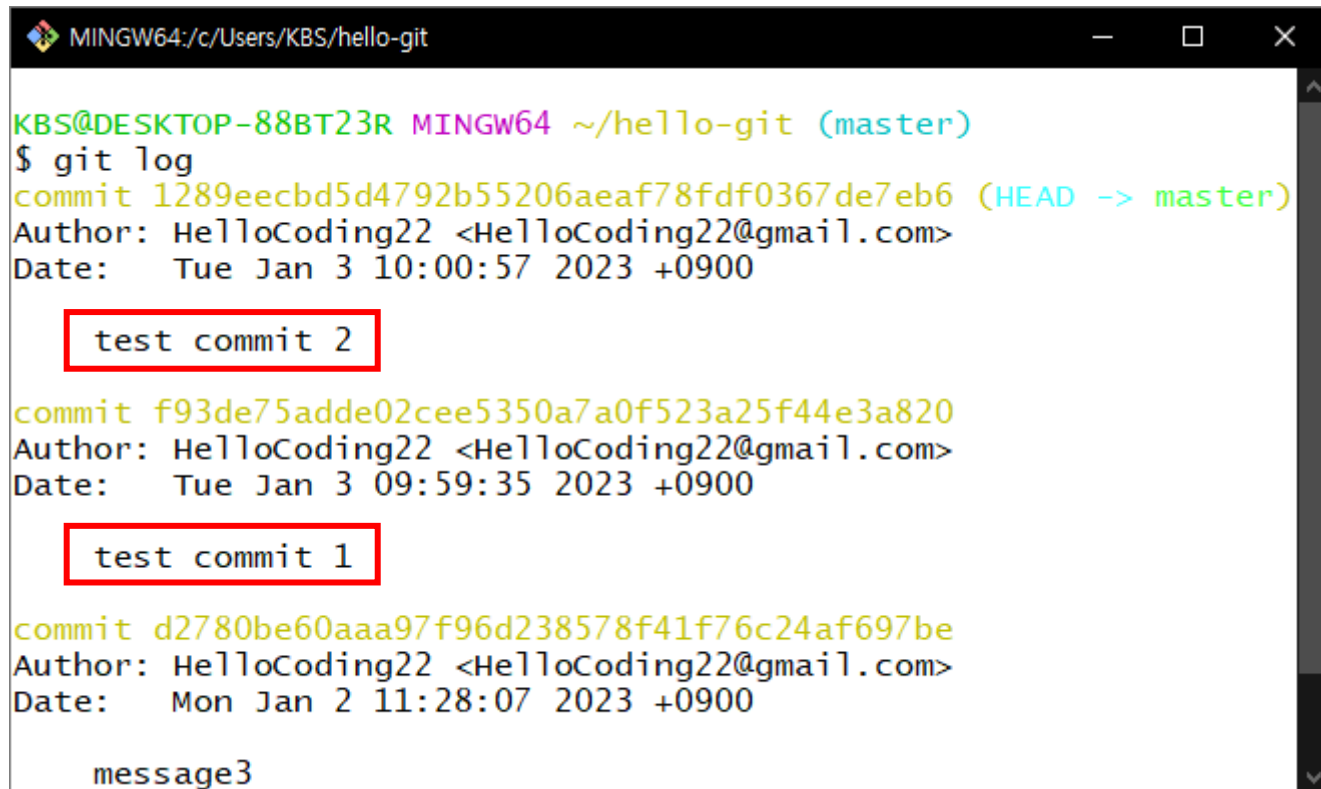
A screenshot of a Windows terminal window titled "MINGW64:/c/Users/KBS/hello-git". The prompt is "KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)". The command entered is "\$ git reset --hard 1289eecbd5d4792b55206aeaf78fdf0367de7eb6". The output is "HEAD is now at 1289eec test commit 2", which is highlighted with a red rectangular box.

```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git reset --hard 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
HEAD is now at 1289eec test commit 2
```

# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64: c:/Users/KBS/hello-git

KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:00:57 2023 +0900

    test commit 2

commit f93de75adde02cee5350a7a0f523a25f44e3a820
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 09:59:35 2023 +0900

    test commit 1

commit d2780be60aaa97f96d238578f41f76c24af697be
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Mon Jan 2 11:28:07 2023 +0900

    message3
```

# [실습] 특정 커밋으로 되돌리기(reset) --hard

## ❖ test.txt 파일 확인

```
$ cat test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ cat test.txt
a
b
```



# 특정 커밋으로 되돌리기(revert)

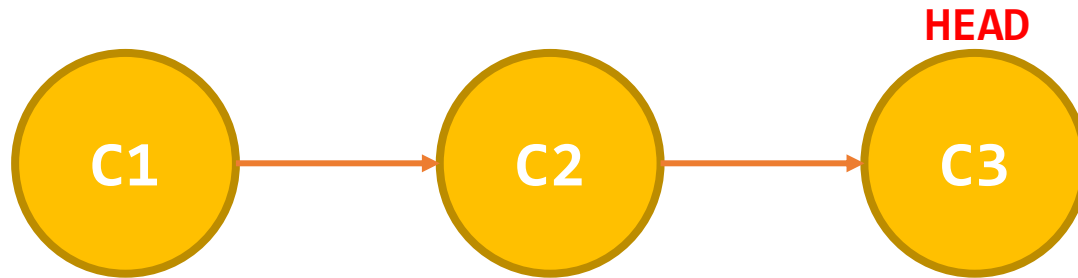
## ❖ 특정 커밋으로 되돌리기

```
$ git revert 취소할_커밋해시
```

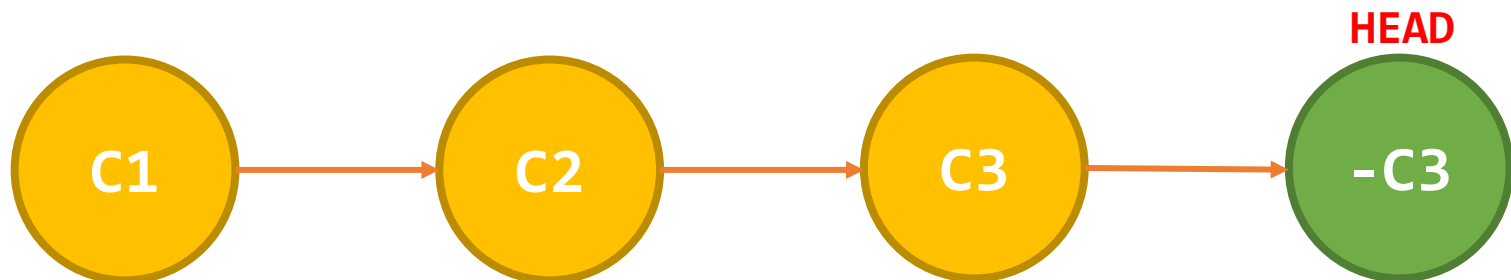
- 특정 버전의 커밋으로 되돌린 다음 그 이후 버전을 남겨두려는 경우에 사용

# 특정 커밋으로 되돌리기(revert)

## ❖ C1 커밋으로 되돌아가기



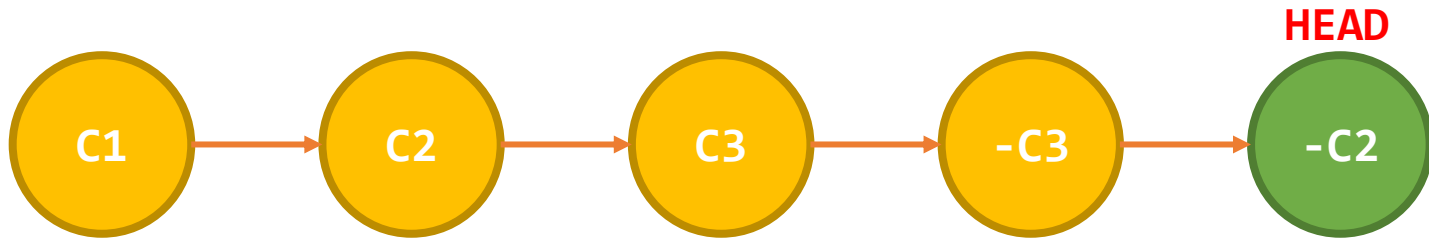
- C3 커밋 취소



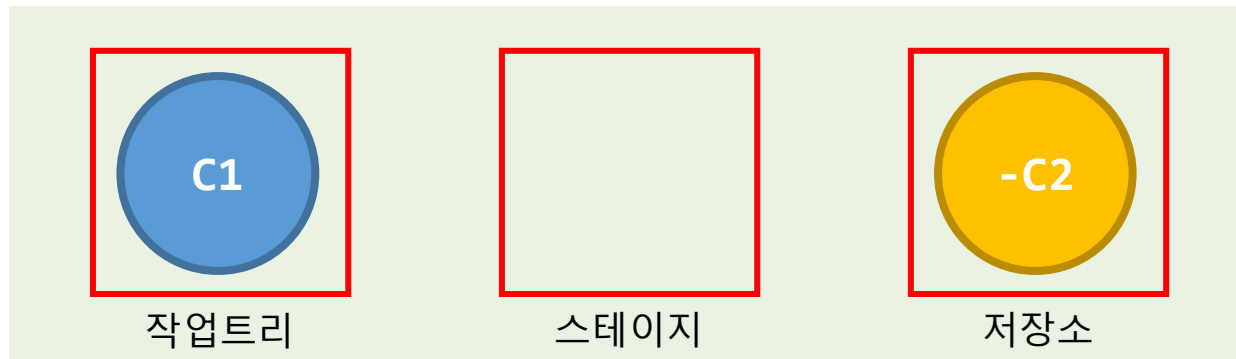
# 특정 커밋으로 되돌리기(revert)

## ❖ C1 커밋으로 되돌아가기

- C2 커밋 취소



- 취소한 커밋의 이력을 남기면서 C1 커밋으로 돌아갈 수 있음



# [실습] 특정 커밋으로 되돌리기(revert)

## ❖ test.txt 파일에 내용 추가하기(e)

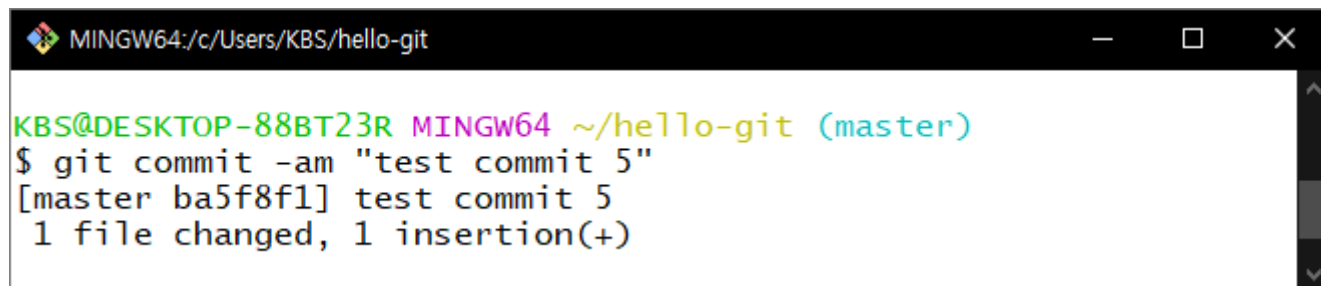
```
$ vim test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
a
b
e
~
test.txt[+] [dos] (10:10 03/01/2023) 3,2 A11
-- INSERT --
```

## ❖ 스테이징과 커밋을 동시에 실행하기

```
$ git commit -am "test commit 5"
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git commit -am "test commit 5"
[master ba5f8f1] test commit 5
1 file changed, 1 insertion(+)
```

# [실습] 특정 커밋으로 되돌리기(revert)

## ❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit ba5f8f120021eac6808bc0ee5a360c3192496523 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:26:38 2023 +0900
    test commit 5

commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:00:57 2023 +0900
    test commit 2

commit f93de75adde02cee5350a7a0f523a25f44e3a820
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 09:59:35 2023 +0900
    test commit 1

commit d2780be60aaa97f96d238578f41f76c24af697be
Author: HelloCoding22 <HelloCoding22@gmail.com>
```

# [실습] 특정 커밋으로 되돌리기(revert)

- ❖ "test commit 5"를 취소하고 "test commit 2"로 돌아가기
  - "test commit 5"의 커밋 해시를 복사

```
$ git revert ba5f8f120021eac6808bc0ee5a360c3192496523
```

revert한 버전명  
커밋 메시지 입력(옵션)



```
MINGW64: c:/Users/KBS/hello-git
Revert "test commit 5"
커밋 메시지 작성 : 커밋 보류함
This reverts commit ba5f8f120021eac6808bc0ee5a360c3192496523.
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   test.txt
#
~
~
~
~
~
<lo-git/.git/COMMIT_EDITMSG[+] [unix] (10:32 03/01/2023)7,31 All
-- INSERT --
```

# [실습] 특정 커밋으로 되돌리기(revert)

## ❖ test.txt 파일 확인

```
$ cat test.txt
```

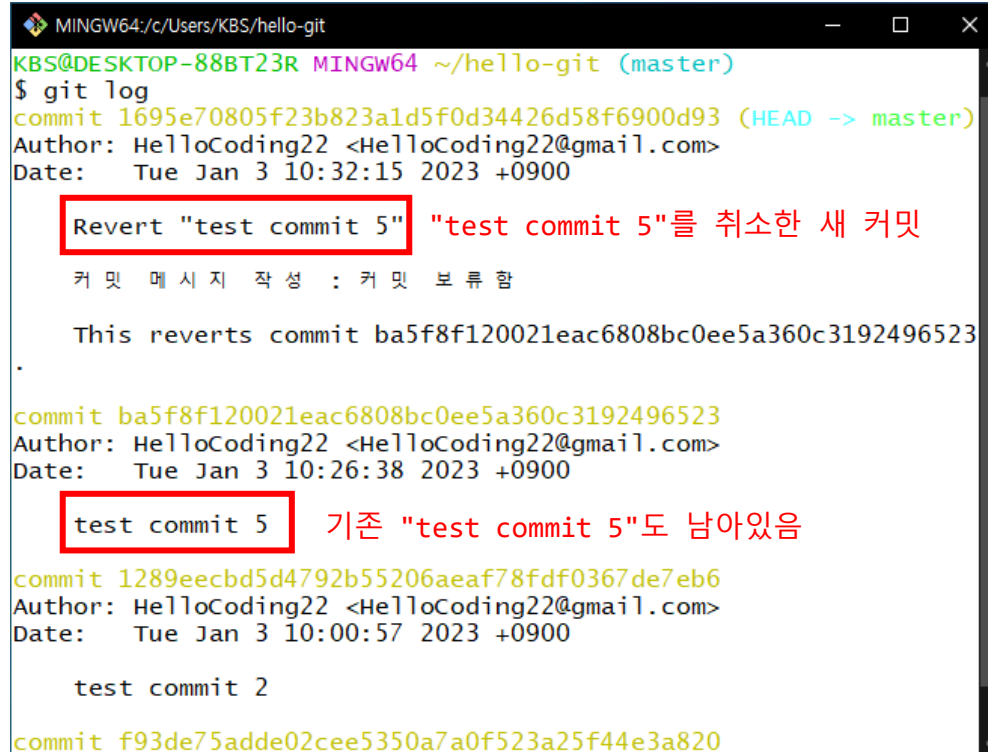
A screenshot of a Windows terminal window titled 'MINGW64:~/c/Users/KBS/hello-git'. The prompt is 'KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)'. The command '\$ cat test.txt' has been entered, and the output is 'a' followed by 'b' on the next line.

```
MINGW64:~/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ cat test.txt
a
b
```

# [실습] 특정 커밋으로 되돌리기(revert)

## ❖ 커밋 기록 확인하기

```
$ git log
```



```
MINGW64: c:/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ git log
commit 1695e70805f23b823a1d5f0d34426d58f6900d93 (HEAD -> master)
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:32:15 2023 +0900

    Revert "test commit 5" "test commit 5"를 취소한 새 커밋

    커밋 메시지 작성 : 커밋 보류함

    This reverts commit ba5f8f120021eac6808bc0ee5a360c3192496523
    .
commit ba5f8f120021eac6808bc0ee5a360c3192496523
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:26:38 2023 +0900

    test commit 5 기존 "test commit 5"도 남아있음

commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
Author: HelloCoding22 <HelloCoding22@gmail.com>
Date: Tue Jan 3 10:00:57 2023 +0900

    test commit 2

commit f93de75adde02cee5350a7a0f523a25f44e3a820
```



# [실습] 특정 커밋으로 되돌리기(revert)

- ❖ "test commit 2"를 취소하고 "test commit 1"로 돌아가기
  - "test commit 2"의 커밋 해시를 복사

```
$ git revert 1289eecbd5d4792b55206aeaf78fdf0367de7eb6
```

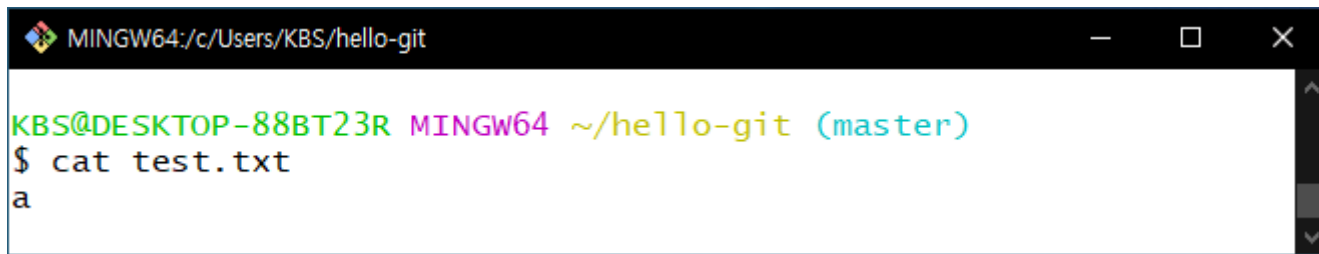


```
MINGW64:/c/Users/KBS/hello-git
Revert "test commit 2"
This reverts commit 1289eecbd5d4792b55206aeaf78fdf0367de7eb6.
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   bye.txt
#   modified:   test.txt
#
~
~
~
~
~/hello-git/.git/COMMIT_EDITMSG [unix] (10:39 03/01/2023)1,1 All
```

# [실습] 특정 커밋으로 되돌리기(revert)

## ❖ test.txt 파일 확인

```
$ cat test.txt
```



```
MINGW64:/c/Users/KBS/hello-git
KBS@DESKTOP-88BT23R MINGW64 ~/hello-git (master)
$ cat test.txt
a
```

# [실습] 특정 커밋으로 되돌리기(revert)

## ❖ 커밋 기록 확인하기

```
$ git log
```



08

정리하기

# 정리하기

<code>git init</code>	현재 위치에 저장소 생성하기
<code>git status</code>	깃 상태 확인하기
<code>git add test.txt</code>	test.txt 파일을 스테이지에 올리기
<code>git commit -m "first commit"</code>	first commit 메시지와 함께 커밋하기
<code>git add .</code>	수정한 전체 파일을 스테이징
<code>git log</code>	커밋 정보 확인하기
<code>git reset HEAD^</code>	최신 커밋 취소하기
<code>git reset</code> 옵션 커밋해시	지정한 커밋해시로 이동 후, 이후 커밋 취소하기
<code>git revert</code> 커밋해시	지정한 커밋해시의 변경이력을 남기면서 취소하기

**THANK 😊 YOU**