

02

깃으로 버전 관리하기

02-1 깃 저장소 만들기

02-2 버전 만들기

02-3 커밋 내용 확인하기

02-4 단계마다 파일 상태 알아보기

02-5 작업 되돌리기

깃 초기화

```
$ git init
```

원하는 디렉토리를 저장소로 사용할 수 있게 만들 초기화 후에 .git 디렉터리가 생김 (숨김 폴더임)

```
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git
$ ls -la
total 12
drwxr-xr-x 1 funco 197610 0 7월  26 11:13 ./
drwxr-xr-x 1 funco 197610 0 7월  26 11:13 ../

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git
$ git init
Initialized empty Git repository in C:/Users/funco/hello-git/

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ ls -la
total 16
drwxr-xr-x 1 funco 197610 0 7월  26 11:16 ./
drwxr-xr-x 1 funco 197610 0 7월  26 11:13 ../
drwxr-xr-x 1 funco 197610 0 7월  26 11:16 .git/

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ !
```

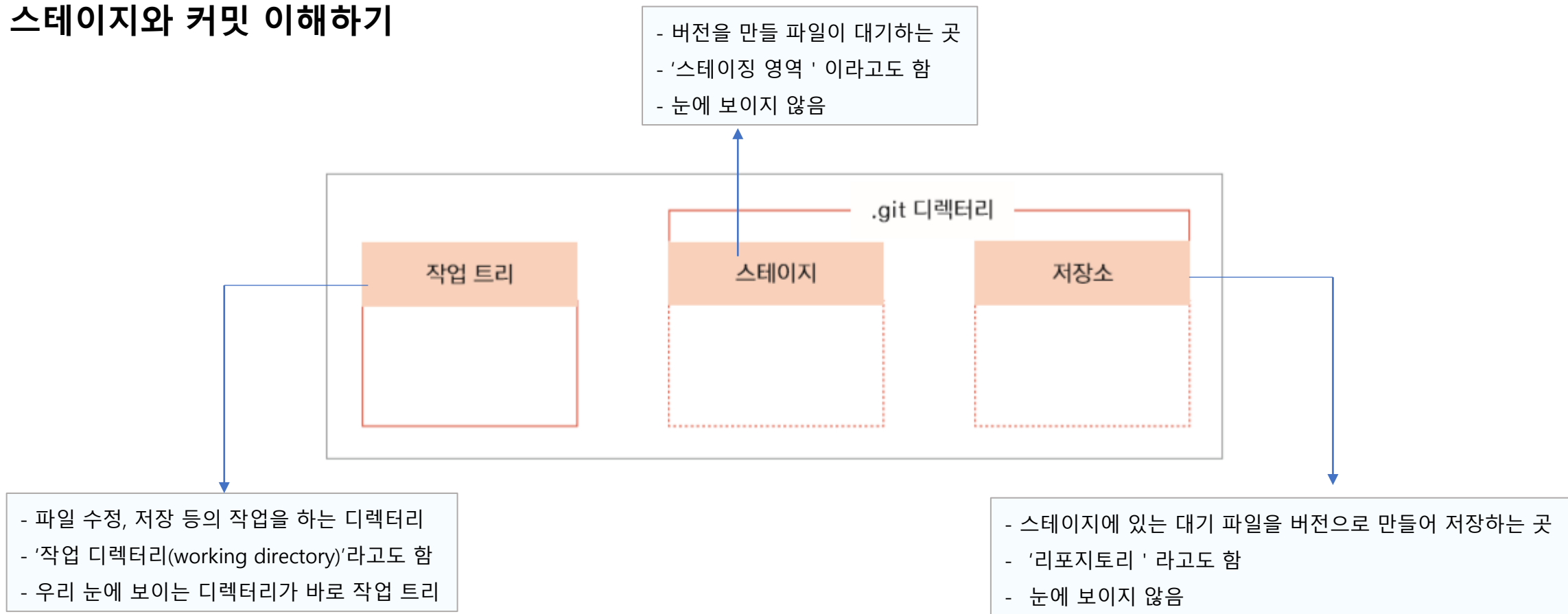
[실습]

- ① hello-git 디렉터리 만들고 이동
- ② 디렉터리 내용 살펴보기 -> 파일도, 디렉터리도 없음
- ③ 깃 초기화
- ④ 다시 한번 디렉터리 내용 살펴보기 -> .git 디렉터리 생김

버전이란

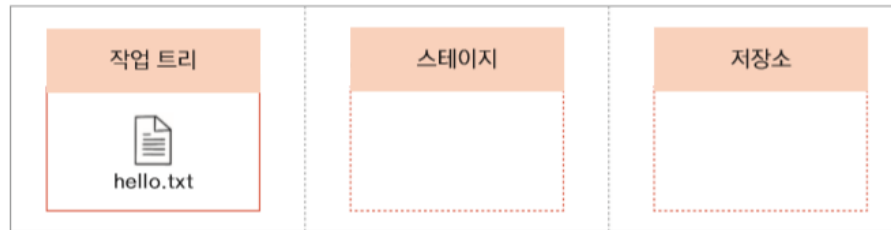
- 깃에서 문서를 수정하고 저장할 때마다 생기는 것.
- 버전마다 변경 시점과 변경 내용을 확인할 수 있음
- 이전 버전으로 되돌아갈 수 있음

스테이지와 커밋 이해하기



버전을 만드는 과정

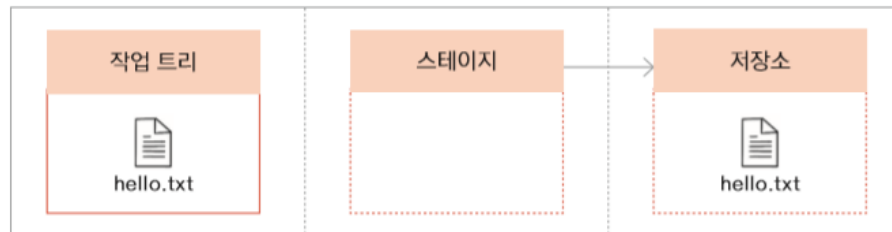
1) 작업 트리에서 파일을 수정하고 저장



2) 버전을 만들고 싶다면 스테이지에 넣음



3) 스테이지에 있던 파일을 저장소에 버전으로 저장



[실습] 작업 트리에서 문서 수정하기

1) git status

저장소 상태 확인 (앞에서 초기화한 hello-git)

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

① On branch master : 현재 master 브랜치에 있습니다.
② No commits yet : 아직 커밋한 파일이 없습니다.
③ nothing to commit : 현재 커밋할 파일이 없습니다.

2) vim hello.txt

빔 편집기로 문서 작성

- ① 빔 화면에서 [I] 또는 [A] 누르고
- ② 숫자 '1' 입력
- ③ [ESC] 누른 후
- ④ ':wq' + [Enter] 눌러서 저장 & 종료

3) ls -la

디렉터리 안에 hello.txt 문서가 만들어졌는지 확인

```
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ ls -la
total 21
drwxr-xr-x 1 funco 197610 0 7월 26 13:18 ./
drwxr-xr-x 1 funco 197610 0 7월 26 13:18 ../
drwxr-xr-x 1 funco 197610 0 7월 26 13:12 .git/
-rw-r--r-- 1 funco 197610 2 7월 26 13:18 hello.txt
```

4) git status

다시 저장소 상태 확인 → untracked files

```
funco@DESKTOP-CKTPGVC 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)

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```

[실습] 수정한 파일을 스테이징하기

1) **git add hello.txt**

작업 트리에서 수정한 파일(hello.txt)을 스테이지에 추가

2) **git status**

다시 저장소 상태 확인 → **changes to be committed**

```
funco@DESKTOP-CKTPGVC 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

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```



[실습] 스테이지에 있는 파일을 커밋하기

1) `git commit -m "message1"`

- 버전을 만드는 것을 '커밋한다'라고 함
- 버전에 어떤 변경이 있었는지 메시지를 함께 기록해 둬.

```
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git commit -m "message1"
[master (root-commit) ac5b4fa] message1
1 file changed, 1 insertion(+)
create mode 100644 hello.txt
```

2) `git status`

다시 저장소 상태 확인 → **nothing to commit**

```
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git status
On branch master
nothing to commit, working tree clean

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```

3) `git log`

- 저장소에 있는 버전(커밋)을 확인
- 커밋을 만든 사람, 만든 시간, 커밋 메시지 등이 나타남

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

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git log
commit ac5b4fa5e2813c1e2a7934f964fcf27dc90f1f88 (HEAD -> master)
Author: Kyunghee <jump2dev@gmail.com>
Date:   Fri Jul 26 16:22:32 2019 +0900

    message1

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```



[실습] 스테이징과 커밋을 한꺼번에 처리하기

1) vim hello.txt

- 커밋하려면 수정 내용이 있어야 하므로
- 빔 편집기에서 숫자 '2' 입력& 저장해서 hello.txt 파일 수정

2) git commit -am "message2"

- commit 명령에 (add를 의미하는) -a 옵션 추가
- git commit -a -m 처럼 옵션을 따로 써도 됨.
- 스테이징과 커밋을 한꺼번에 처리하려면
무조건 한번은 커밋했던 파일이어야 함

3) git log

- 저장소에 있는 버전 확인
- 커밋을 만든 사람, 만든 시간, 커밋 메시지 등이 나타남

```
funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git log
commit d3eb0361d7592f0195fe16e414d4ccbe6b291c5d (HEAD -> master)
Author: Kyunghee <jump2dev@gmail.com>
Date:   Fri Jul 26 23:22:45 2019 +0900

    message2

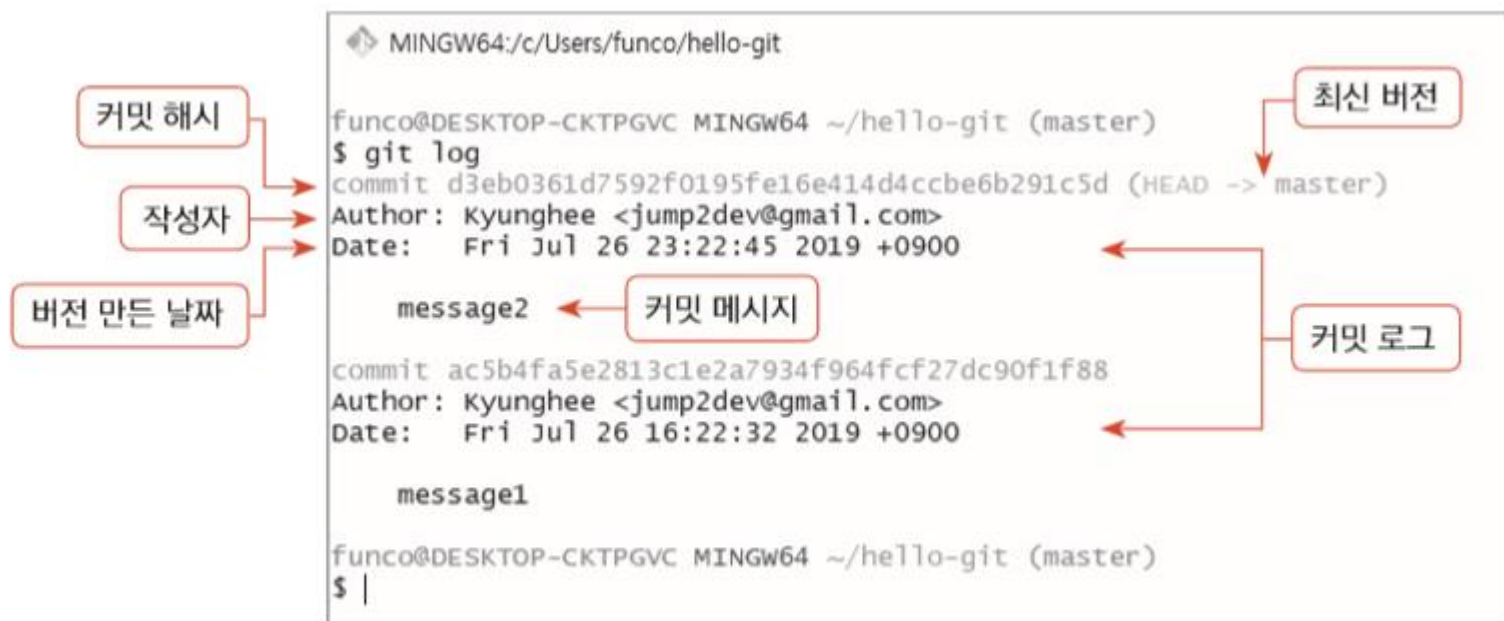
commit ac5b4fa5e2813c1e2a7934f964fcf27dc90f1f88
Author: Kyunghee <jump2dev@gmail.com>
Date:   Fri Jul 26 16:22:32 2019 +0900

    message1

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```

git log

- 커밋 기록 확인
- 지금까지 만든 버전이 나타나고 각 버전마다 설명이 함께 표시됨



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

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git log
commit d3eb0361d7592f0195fe16e414d4ccbe6b291c5d (HEAD -> master)
Author: Kyunghee <jump2dev@gmail.com>
Date:   Fri Jul 26 23:22:45 2019 +0900

    message2

commit ac5b4fa5e2813c1e2a7934f964fcf27dc90f1f88
Author: Kyunghee <jump2dev@gmail.com>
Date:   Fri Jul 26 16:22:32 2019 +0900

    message1

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ |
```

git diff

특정 파일의 현재 상태와 최신 버전의 차이

1) vim hello.txt

- vim 편집기에서 숫자 '2' 를 'two'로 변경

2) git diff

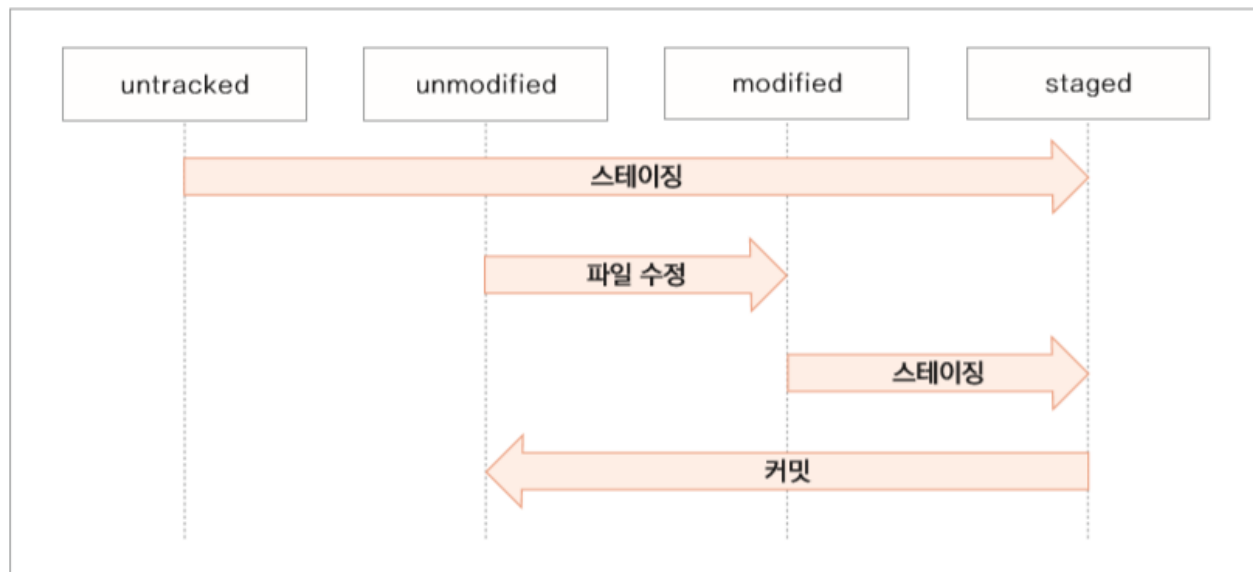
- 수정한 파일과 저장소에 있는 파일 비교

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

funco@DESKTOP-CKTPGVC MINGW64 ~/hello-git (master)
$ git diff
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 1191247..0b66db0 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,2 @@
1
-2
+two
```

수정한 내용으로 다시 버전을 만들려면 스테이지에 올린 후 커밋하고
수정한 내용을 버리려면 git checkout 명령을 사용해 수정 내용 취소

- 버전을 만드는 각 단계마다 파일의 상태가 달라짐.
- 파일의 상태를 이해하면 현재 어느 단계인지, 그 상태에서 어떤 일을 할 수 있는지 알 수 있음
 - tracked : 버전을 만든 후에 깃에서 변경 내역을 추적 중인 상태
 - untracked : 한번도 버전을 만들지 않은 상태
 - unmodified : 수정되지 않은 상태 (working tree is clean)
 - modified : 작업 트리에서 수정한 후 아직 스테이지에 저장하지 않은 상태
 - staged : 스테이지에 있고 아직 커밋하지 않은 상태



작업 트리에서 수정한 파일 되돌리기

git checkout -- 파일명

파일을 수정한 후 최신 버전으로 되돌리려고 할 때

스테이징 되돌리기

git reset HEAD 파일명

파일을 스테이지에 올린 상태에서 되돌리려고 할 때

최신 커밋 되돌리기

git reset HEAD^

가장 마지막에 했던 커밋을 되돌리려고 할 때

커밋 취소와 함께 스테이지에서도 내려짐

작업 트리에서 수정한 파일 되돌리기

git reset -- hard 커밋해시

특정 버전 상태로 되돌리려고 할 때

git log를 사용해 커밋 해시 확인한 후

해당 커밋 해시를 가진 커밋으로 리셋 & 이후 커밋은 삭제됨

작업 트리에서 수정한 파일 되돌리기

git revert 커밋해시

특정 버전 상태로 되돌리려고 할 때

git log를 사용해 커밋 해시 확인한 후

해당 커밋 해시의 직전 커밋으로 돌아감 & 이후 커밋 그대로 유지