2-4. 버전 만드는 단계마다 파일 상태 알아보기

깃에서는 **버전을 만드는 각 단계마다 파일 상태를 다르게 표시**한다. 그래서 **파일의 상태를 이해하면 이 파일이 버전 관리의 여러 단계 중 어디에 있는지, 그 상태에서 어떤 일을 할 수 있는지 알 수 있다.**

**\* tracked 파일과 untracked 파일**

git status 명령을 사용하면 화면에 파일 상태와 관련된 여러 메시지가 출력된다. **Working Tree에 있는 파일은 크게 tracked 상태와 untracked 상태로 나뉜다.** 두 개의 상태가 무엇을 의미하는 지 알아보자.

1. hello.txt 파일을 열고 숫자 ‘3’을 추가한 후 저장하자.

2. hello2.txt라는 새로운 파일을 만들고 a, b, c, d를 한 줄에 한 글자씩 입력하고 파일을 저장한 후 편집기를 종료한다.

3. hello.txt와 hello2.txt 파일 모두 작업 트리에 있다. **git status 명령**을 사용해 어떤 상태인지 확인해 보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

앞에서 커밋했던 hello.txt 파일은 **‘changes not staged for commit:’**이라고 되어있다. 즉, **변경된 파일이 아직 Stage에 올라가지 않았다**는 의미이다. 그리고 파일 이름 앞에 **‘modified:’**라고 되어 있어 **hello.txt가 수정되었다**는 것을 알 수 있다. 이렇게 깃은 한 번이라도 commit을 한 파일의 수정 여부를 계속 추적한다. 이렇게 **깃이 추적하고 있는 파일**을 **tracked 파일**이라고 부른다.

반면에 hello2.txt 파일 앞에는 아무것도 없고 바로 위에는 **‘untracked files:’**라고 되어 있다. hello2.txt 파일은 **한 번도 깃에서 버전 관리를 하지 않았기 때문에 수정 내역을 추적하지 않는다.** 이러한 파일을 **untracked 파일**이라고 부른다.

4. 수정했던 hello.txt 파일과 hello2.txt 파일을 모두 **git add 명령**을 사용해 Stage에 올리고 **깃의 상태를 확인**해 보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

마지막 버전 이후에 수정된 hello.txt는 **‘modified:’**로 표시되고, 한 번도 관리하지 않았던 hello2.txt는 **‘new file:’**로 표시된다. 또한 **tracked 파일과 untracked 파일 모두 Stage에 올라온 것을 확인**할 수 있다.

5. 이제 commit을 수행하고 log를 확인해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

‘message3’라는 메시지를 붙인 commit이 보인다. 하지만 **각 commit에 어떤 파일들이 관련된 것인지 알 수 없다.** commit에 관련된 파일까지 함께 살펴보려면 git log 명령에 **--stat 옵션**을 사용해야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

로그 메시지가 너무 많을 경우 한 화면씩 나누어 보여준다. Enter를 누르면 다음 로그 화면을 볼 수 있고, Q를 누르면 로그 화면을 빠져나와 다시 깃 명령을 입력할 수 있다.

**\* unmodified, modified, staged 상태**

한 번이라도 버전을 만들었던 파일은 tracked 상태가 된다. tracked 상태인 파일은 깃 명령으로 파일 상태를 확인하면 현재 작업 트리에 있는지, 스테이지에 있는지 등 더 구체적인 상태를 알려준다. 깃의 commit 과정 중에서 tracked 파일의 상태가 어떻게 바뀌는 지 확인해 보자.

1. ls 명령을 사용해 hello-git 디렉터리를 살펴보면 hello.txt 파일과 hello2.txt 파일이 있다. 여기서는 hello2.txt 파일의 상태를 따라가 보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

2. git status 명령을 사용해 깃의 상태와 파일의 상태를 살펴보면 **Working Tree에 아무 변경 사항도 없는 것을 알 수 있다.** ‘working tree clean’이라고 나타나면 현재 Working Tree에 있는 모든 파일의 상태는 **unmodified 상태**이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

3. hello2.txt 파일을 수정해보자. ‘a’만 남기고 모든 내용을 삭제해보자.

4. 다시 깃의 상태를 확인해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

hello2.txt 파일이 수정되었고 아직 스테이지에 올라가지 않았다고 나타난다. ‘changes not staged for commit:’ 이라는 메시지가 나타나면 **파일이 수정만 된** **modified 상태**이다.

5. **git add** 명령을 사용해 스테이지에 올리고 git status 명령을 실행해 보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

‘changes to be committed:’라는 메시지가 나타나면 commit 직전 단계, 즉 staged 상태가 된다.

6. 스테이지에 있는 hello2.txt 파일을 **commit**해보자. 그리고 git status 명령을 실행해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

**commit을 수행하면 hello2.txt 파일의 상태는 unmodified로 돌아간 것을 볼 수 있다.**

**(참고) 커밋한 메시지 수정하기**

문서의 수정 내용을 기록해둔 commit message를 잘못 입력했다면 commit을 만든 즉시 수정할 수 있다. **가장 최근의 commit message를 수정하려면** git commit 명령에 **--amend 옵션**을 추가한다.



명령을 입력하면 빔이 실행되면서 원래 commit message가 화면 위쪽에 나타난다.

텍스트이(가) 표시된 사진

자동 생성된 설명

I를 눌러 입력 모드로 바꾸면 메시지를 수정할 수 있다.

수정이 끝났다면 ex 모드로 돌아가 빔을 종료하면 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명