2-2. 버전 만들기

깃을 통해 버전을 관리하면 원래 파일 이름은 그대로 유지하면서 파일에서 무엇을 변경했는지를 변경 시점마다 저장할 수 있다. 또한 각 버전마다 작업했던 내용을 확인할 수 있고, 그 버전으로 되돌아갈 수도 있다.

그러면 깃은 어떻게 파일 이름은 그대로 유지하면서 수정 내역을 기록할까? 먼저 3가지 용어를 살펴보자.

**(1) Working Tree**

**Working Tree**는 **파일 수정, 저장 등의 작업을 하는 디렉터리**로 **Working Directory**라고도 부른다. 앞에서 만들었던 hello-git 디렉터리가 작업 트리가 된다. 즉, **우리 눈에 보이는 디렉터리**가 바로 작업 트리이다.

**(2) Stage**

**Stage**는 **버전으로 만들 파일이 대기하는 곳**이다. **Staging Area**이라 부르기도 한다. 예를 들어, Staging Tree에 10개의 파일을 수정했는데 4개의 파일만 버전으로 만들려면 4개의 파일만 스테이지로 넘겨주면 된다.

**(3) Repository**

**Repository**는 **스테이지에서 대기하고 있던 파일들을 버전으로 만들어 저장하는 곳**이다.

(참고) Stage의 내용은 .git/index 파일에 저장되고, Repository의 내용은 .git/HEAD 파일에 저장된다.

Stage와 Repository는 눈에 보이지 않는다. 깃을 초기화 했을 때 만들어지는 .git 디렉터리 안에 숨은 파일 형태로 존재하는 영역이다. 이제 깃이 버전을 만드는 과정을 살펴보자.

**1. hello.txt. 파일 문서를 수정하고 저장하면 그 파일은 Working Tree에 있게 된다.**

**2. 수정한 hello.txt. 파일을 버전으로 만들고 싶다면 Stage에 넣는다.**

**3. 파일 수정을 끝내고 Stage에 넣었다면 버전을 만들기 위해 commit 명령을 내린다. commit 명령을 내리면 새로운 버전이 생성되면서 Stage에 대기하던 파일이 모두 Repository에 저장된다.**

**\* Working Tree에서 빔으로 문서 수정하기**

이제부터 Working Tree에서 문서를 직접 수정해 보자. 앞에서 만들었던 hello-git 디렉터리에 새로운 파일을 만들어보자.

1. 터미널 창을 열어 hello-git 디렉터리로 이동하자. 앞에서 hello-git 디렉터리에서 깃을 초기화했으므로 버전 관리를 할 수 있다. 먼저 **깃 상태를 확인**하기 위해 **status 명령**을 사용해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

깃의 상태를 나타내는 메시지의 의미를 간단히 살펴보자.

**On branch master: 현재 master 브랜치에 있습니다.**

**No commits yet: 아직 commit한 파일이 없습니다.**

**nothing to commit: 현재 커밋할 파일이 없습니다.**

2. hello-git 디렉터리에 새로운 파일을 만들어보자.



해당 파일에는 숫자 1을 입력해 놓자.

3. 터미널 창으로 돌아가 **ls -la 명령**을 수행해보자. 방금 만든 파일이 만들어졌을 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

4. git status 명령을 사용해 다시 깃의 상태를 확인해보자. 이번에는 조금 다른 메시지가 나타날 것이다. **branch master에 hello.txt라는 untracked files가 있다**고 한다. 깃에서는 **아직 한 번도 버전 관리하지 않은 파일**을 **untracked files**라고 부른다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**\* 수정한 파일을 Staging하기 - git add**

지금까지의 과정은 Working Tree에 문서 파일을 수정하고 저장하는 과정이었다.

이번에는 **Working Tree에서 만들거나 수정한 파일을 Stage에 추가하는 방법**을 알아보자. 이렇게 **깃에게 버전 만들 준비를 하라고 알려주는 것**을 **Staging** 혹은 **‘스테이지에 올린다’** 라고 표현한다.

(참고) Staging 내용을 .git/index 파일에 저장하기 때문에 스테이지에 올리는 것을 ‘Index에 등록한다’라고도 한다.

1. 깃에서 **Staging할 때 사용하는 명령**은 **git add**이다. 터미널에 입력해도 아무런 내용도 나타나지 않는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

(참고) **깃에서 사용하는 명령은 리눅스를 기반으로 하고, 윈도우에서는 깃 배시 프로그램을 거쳐서 깃 명령을 사용한다.** 그런데 윈도우에서 깃을 사용할 때 주의할 점이 있다. 윈도우의 줄바꿈 문자와 리눅스의 줄바꿈 문자가 다르다는 것이다.

\* 윈도우에서는 줄이 바뀌는 자리에 CR 문자와 LF 문자가 삽입된다. (합쳐서 CRLF 문자라고 부른다.)

\* 리눅스와 맥에서는 줄이 바뀌는 자리에 LF 문자가 삽입된다.

따라서 윈도우에서 텍스트 문서를 스테이지에 올릴 때는 **‘warning: LF will be replaced by CRLF in hello.txt.’**라는 경고 메시지가 나타난다. 이 메시지는 **깃에서 자동으로 텍스트 문서의 CRLF 문자를 LF 문자로 변환해서 커밋할 것이라는 의미**다.

2. 다시 **깃 상태를 확인**해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

untracked files: 라는 문구가 **changes to be committed:**로 바꼈다. 그리고 hello.txt. 파일 앞에 **new file:** 이라는 수식어가 추가로 나타난다. **‘새 파일 hello.txt를 (앞으로) commit할 것이다.’**라는 뜻이다.

**\* Stage에 올라온 파일 Commit하기 - git commit**

이제 hello.txt가 Stage에 추가되었다. 이제 버전을 만들 준비가 끝난 것이다.

파일이 Stage에 올라갔다면 이제 버전을 만들 수 있다. **깃에서는 버전을 만드는 것**을 **‘commit한다’**고 말한다. 커밋할 때는 그 버전에 어떤 변경 사항이 있었는지 확인하기 위해 **메시지를 함께 기록해 두어야 한다.**

1. **깃에서 파일을 커밋하는 명령**은 **git commit**이다. 한 칸 띄운 후에 **-m 옵션**을 붙이면 **commit과 함께 저장할 메시지를 적을 수 있다.** 이를 **‘commit message’**라고 부른다.

텍스트이(가) 표시된 사진

자동 생성된 설명

commit 후 결과 메시지를 보면 **파일 1개가 변경되었고, 파일에 1개의 내용이 추가되었다**고 나타난다. 스테이지에 있던 hello.txt 파일이 저장소에 추가된 것이다.

2 이제 깃 상태를 확인해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

버전으로 만들 파일이 없고(nothing to commit) Working Tree도 수정사항 없이 깨끗하다(working tree clean)고 나타난다.

3. **버전이 제대로 만들어졌는지 확인**하려면 **git log 명령**을 사용하면 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

방금 커밋한 버전에 대한 설명이 나타난다. commit을 만든 사람, 만든 시간과 commit 메시지가 함께 나타난다. 수정한 파일을 commit하면 이렇게 수정과 관련된 여러 정보를 함께 저장할 수 있고 필요할 때 확인할 수 있다.

**\* Staging과 Commit 한꺼번에 처리하기 - git commit -am**

commit 명령에 **-am 옵션**을 사용하면 **스테이지에 올리고 커밋하는 과정을 한꺼번에 처리할 수 있다.** **단, 한 번이라도 커밋한 적이 있는 파일을 다시 커밋할 때만 사용할 수 있다.**

1. 먼저 hello.txt 파일을 열어 파일을 수정하자. (숫자 2를 추가하자.)

2. git commit 명령에 -am 옵션을 붙여서 Staging과 commit을 동시에 처리해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

(참고) “git commit -a -m “message”라고 입력해도 된다.

3. git log 명령을 사용해 방금 커밋한 버전에 어떤 정보가 들어있는지 확인해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

수정한 hello.txt를 저장한 두 번째 버전의 정보가 message2라는 메시지와 함께 나타난다.