2-3. 커밋 내용 확인하기

버전을 관리하기 위해서는 지금까지 어떤 버전을 만들었는지 알 수 있어야 한다. 또 각 버전마다 어떤 차이가 있는지도 파악할 수 있어야한다.

**\* Commit 기록 자세히 살펴보기 - git log**

깃에서 자주 사용하는 명령 중 하나가 지금까지 커밋했던 기록을 살펴보기 위한 명령인 git log이다. git log 명령을 입력하면 **지금까지 만든 버전이 화면에 출력되고 각 버전마다 설명도 함께 나타난다.** 위에서 git log 명령을 입력했을 때 나타난 화면을 더 자세히 살펴보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

commit 항목 옆에 **영문과 숫자로 된 긴 문자열**이 나타나는데 이것을 **commit hash** 또는 **git hash**라고 부른다. **commit을 구별하는 아이디**라고 생각하면 된다. 그리고 commit hash 옆에 있는 **(HEAD -> master)**는 이 버전이 **가장 최신이라는 표시**다. 그리고 **Author 항목은 누가 만들었는지, Date는 버전이 언제 만들어졌는지 나타낸다.** 그리고 그 아래에는 작성자가 기록한 commit message가 나온다. 이렇게 git log 명령을 입력했을 때 나오는 정보를 묶어 **commit log**라고 부른다.

**\* 변경 사항 확인하기 - git diff**

큰 규모의 프로그램을 짠다고 생각해 보자. 수만 줄의 소스 코드를 수정한 다음 저장소에 있는 최근 버전과 비교해 어떤 부분이 다른지 찾아야 한다면 어떻게 할까? commit message를 참고해도 쉽지 않을 것이다.

이럴 때 **git diff 명령**을 사용하면 **작업 트리에 있는 파일과 스테이지에 있는 파일을 비교**하거나, **스테이지에 있는 파일과 저장소에 있는 최신 commit을 비교**해 수정한 파일을 커밋하기 전에 최종적으로 검토할 수 있다.

1. 먼저 hello.txt 파일을 열고 기존의 내용 중 ‘2’를 지우고 ‘two’를 추가한 후 저장하자.

2. 깃의 상태를 보면 hello.txt 파일이 수정되었고, 아직 Staging된 상태가 아니라고 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

3. **git diff 명령**을 사용해 방금 수정한 hello.txt 파일이 저장소의 hello.txt와 어떻게 다른지 확인해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 작업 트리에서 수정한 파일과 최신 버전을 비교한 다음, 수정한 내용으로 버전을 만들려면 Stage에 올린 후 commit 명령을 수행하면 된다.

마지막으로 이어지는 실습을 위해 hello.txt 파일을 열고 ‘two’ 부분을 ‘2’로 수정한 후 저장하자.