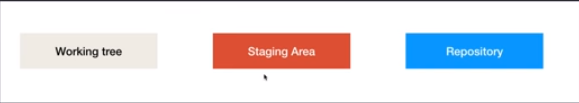
* **Git 설치하기.**

Git bash를 열고 git을 입력하면 버전이 뜸.

-git init . 🡨 현재 디렉토리를 버전관리하겠다.

\*\*.git 이라는 디렉토리를 지우면 안됌. git으로 작업하는 내용들이 .git디렉토리안의 세부 디렉토리에 전부 저장되기 때문이다.



Working tree = 파일을 만들고 수정하는, 버전이 되기 전 단계.

Staging Area = 버전으로 올리고 싶은 파일들을 Working tree에서 가져온다.

Repository = 버전으로 올라온 파일들.

* **여러 개의 파일을 하나의 버전으로**

여러 개의 파일을 스테이지에 올리고 커밋을 하면 한 버전으로 그 파일들이 전부 들어가게 된다. 하지만 버전안에 어떤파일들이 있는지 어떻게 알 수 있을까?

명령어 git log –stat 를 치면 됨.

h2.txt | 1 +

hello.txt | 1 +

2 files changed, 2 insertions(+)

H2.txt 파일과 hello.txt 파일이 새 버전에 등록되었고,

2개의 파일이 바뀌었고 각 파일마다 한 줄씩 추가됨을 알 수 있다.

* **과거 버전으로 돌아가기**

HEAD 🡪 master 이렇게 현재 가장 최근에 commit한 내용을 가리키고 있을 텐데, 이 헤더를 과거 버전으로 가리키면 버전관리를 하고있는 디렉토리 전체가 과거 버전으로 돌아가게 된다.

로그 명령어를 치고 그 버전에 해당하는 아이디를 복사한다.

또, 다시 가장 최근에 commit 했던 시점으로 돌아가고 싶다면

Git checkout master 를 입력한다.

* **Add를 커밋할때마다 하지 않는법, 기본 commit 에디터 바꾸기.**

이 명령어로 에디터 바꿔주자.

git config --global core.editor "nano"

* **버전 삭제**

정확히는 버전 삭제가 아니고 과거 버전으로 돌아가는 것.

Git reset –hard ID

* **버전 되돌리기**

만약 과거 버전으로 돌아가고 싶다면, 역순으로 그 이전단계까지 revert 해야한다. 안그러면 충돌남.

* Revert와 reset의 차이

Revert는 커밋을 타고 내려가서 그 일을 수정하는 것. 특정 커밋을 이런 이유때문에 수정했다는 이력이 남아있음

Reset은 현 커밋부터 특정 커밋까지 내용을 없던일로 만든다. 아예 과거로 돌아가는 것.

협업 시, 이미 push한 상태라면 revert하는 게 맞고, 안 했으면 reset하는게 맞음.

* Tip  
  커밋 id를 태그로 설정하거나, .gitignore 파일을 만드는 등 적재적소에 쓸만한 기술들이 있다. 그리고 반드시 구글 드라이브나 다른 저장소에 협업하는 디렉토리를 올려놓는게 좋다.

**\*\*명령어**

Cat = 파일의 내용을 화면에 출력

Git status = 현재 깃의 버전관리 진행상황을 알림.

Git add 파일명 = 파일명을 Staging Area 에 올림.

Git add 디렉토리명 = 해당 디렉토리 하위에 있는 모든 파일을 스테이지에 올림.

Git commit –m “메세지” = 메시지의 이름을 가진 버전을 올림.

Git commit = 메시지를 입력할 수 있는 창이 뜬다.

Git commit –am = 워킹에 있는 파일들을 전부 add, commit 시킴. 단, 한번 이상은 add 즉, tracked 된 파일만 가능하다.

Git log = 어떤사람이 언제 무슨버전을 commit했는지 로그가 나옴.

Git log –p = 현재까지 commit된 모든 로그를 나열함.

git config --global user.name "박현우" = git에 로그인? 하는 방식인 것 같다. 이걸안하면 commit이 안됌.

Git diff = 어떤 변경사항이 있는지 알려줌.

Git reset –hard = 마지막 버전 이후 작업한 내용을 지움.

Git checkout ID = id에 해당하는 버전으로 버전관리를 하고있는 디렉토리 전체가 해당 시점으로 돌아간다.

Git checkout master = 가장 최신이었던 버전으로 돌아간다.

**\*\*에러**

warning: LF will be replaced by CRLF in hello.txt.

The file will have its original line endings in your working directory

이는 맥 또는 리눅스를 쓰는 개발자와 윈도우 쓰는 개발자가 Git으로 협업할 때 발생하는 **Whitespace** 에러다. 유닉스 시스템에서는 한 줄의 끝이 **LF(Line Feed)**로 이루어지는 반면, 윈도우에서는 줄 하나가 **CR(Carriage Return)**와 **LF(Line Feed)**, 즉 **CRLF**로 이루어지기 때문이다. 따라서 어느 한 쪽을 선택할지 Git에게 혼란이 온 것이다.

에러 해결 방법은 git config --global core.autocrlf true

입력.