**GIT**

소스코드의 버전 관리와 이력 관리가 가능한 분산형버전관리시스템(DVCS - Distributed Version Control System)

장점

소스 코드가 변경된 이력을 쉽게 확인 가능

특정 시점에 저장된 버전과 비교하거나 특정시점으로 되돌아갈 수 있음

다른 사람과 동시에 편집 시 경고가 발생하기에 merge를 쉽게할 수 없음

1. **repository(저장소)**

파일이나 폴더를 저장해두는 장소로 하나의 파일이 변경 이력 별로 구분되어 저장됨



기존에 저장소에 저장된 파일에 변경사항이 발생하면 다른 파일로 인식하기 때문에 파일을 변경사항 별로 구분하여 저장 가능

원격 저장소와 로컬 저장소

원격저장소(Remote repository): 여러 사람과 공유하기 위한 저장소로, 파일이 원격 저장소 전용 서버에서 관리됨

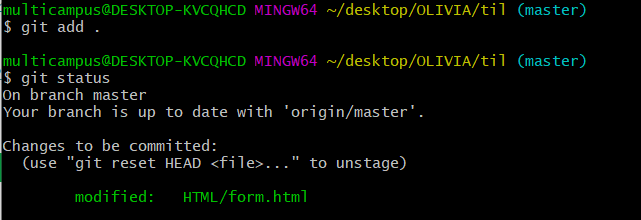
로컬 저장소(Local Repository): PC의 개인 전용 저장소로, 작업한 내용을 원격 저장소에 업로드

1. **변경 기록**
2. Git add .

현재 working directory 에서 commit할 목록에 담아놓는 것 ($ git add .)

로컬 디렉토리에서 작업한 파일을 스테이징 영역에 올려놓는 것으로, 파일 선언하는 단계

// git status <– git의 현재 상태 확인 명령어



1. Git commit

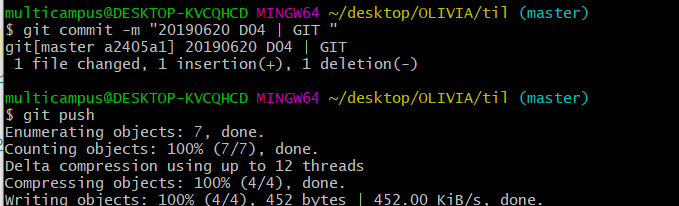
현재 소스 코드 상태를 스냅샷 찍는 것과 동일한 느낌으로, staging area에 담겨 있는 내용을 이력으로 기록

변경된 파일(staging 된 파일)을 로컬 저장소에만 반영하는 것(! 원격 서버에는 올라가지 않았음)

($ git commit –m “ 커밋 메시지”)

1. Git push

원격 저장소로 보내기 ($ git push –u 000(저장소 url 이름) master)



1. **작업트리와 인덱스**

index스: 커밋을 실행하기 전, 저장소와 작업 트리 사이에 존재하는 공간

Commit은 작접 트리에 있는 변경내용을 저장소에 바로 기록하는 것이 아니라, 사이 공간인 index에 파일 상태를 staging 하는 것

1. **Diff**

* 리눅수에서 diff 명령어는 두 파일의 콘텐츠를 한줄씩 비교하여 서로 차이가 나는 부분을 요약해서 보여줌
* git에서도 diff명령이 존재
* git에서의 diff는 세가지 기본 형태의 트리객체나 그와 유사한 객체를 활용함
  + 전체 커밋 그래프 내에 있는 임의의 트리 객체
  + 사용자의 작업디렉토리
  + 인덱스
* git diff

$ git diff

작업디렉토리와 인덱스 간의 차이점을 보여준다. 작업디렉토리에서 변경된사항으로, 다음커밋동안 준비된 상태(onstage)가 될 후보 항복들을 보여주는 것.

$ git diff HEAD

작업 디렉토리와 HEAD를 비교해준다.

$ git diff --cached

인덱스와 HEAD를 비교해준다.

$ git diff HEAD^ HEAD

이전 헤드와 현재 헤드의 비교

$ git diff --cached [ commit ]

onstage상황과 지정된 커밋간의 차이점

$ git diff [ commit 1 ] [ commit 2 ]

두 커밋과의 차이점

1. **Bisect**

git을 활용하다가, 프로그램이 뭔가 결함이 있어 작동하지 않는다고 가정하자. 이전의 작동하는 나쁜버전이 아니라, 좋은 버전을 찾아야 할 때, 해당 명령어로 확인 가능

bisect는 한쪽 끝에서는 좋은 동작을 기준으로, 한쪽 끝에서는 나쁜 동작을 기준으로 범위를 좁혀가면서 새 커밋을 선택하게 되는 과정이고, 결과적으로 잘못된 동작이 도입된 커밋에 도달 가능  
한마디로, 이분검색을 통해 디버깅해가는 과정을 의미함

$ git bisect start

이분 검색을 위한 새로운 브랜치가 생성되는 순간

$ git bisect bad

현재 헤드(HEAD)에 문제가 있음을 알려주거나, bad명령어 뒤에 커밋id를 남겨서 bad 커밋이 무엇인지 확인 가능

$ git bisect good [ commit\_ID ]

오류가 없는 커밋을 알려줌

git은 자동으로 사용자의 디렉토리를 good커밋과 bad 커밋사이의 특정 리비전으로 바꿔주기에, 사용자는 해당 버전이 좋은 버전인지 나쁜 버전인지 테스트 한 다음, git에게 결과를 알려주면 됨

$ git bisect [good/bad]

이분검색을 최대로 실행할 경우 단 하나의 커밋에 도달할때까지 검색이 반복되며, 하나의 good을 찾는 경우 이분검색이 끝났음을 알리는 커밋메시지가 뜬다.

Good, bad 선택 시 잘못되었거나, 다시 시작하고 싶으면 해당 명령어로 한단계 돌아갈 수 있음

$ git bisect reset

$ git bisect replay

1. Reset
2. Soft : 지정된 commit을 가리키도록 head 참조 변경하는 명령어, 인덱스와 작업중인 컨텐츠 그대로 유지됨 ($git reset –soft [commit])
3. Mixed: 지정된 commit을 가리키도록 head를 변경하는 명령어. 트리구조에 맞게 인덱스의 컨텐츠가 변경됨 ($ git reset –mixed [commit])
4. Hard: 지정된 commit을 가리키도록 head를 변경하는 명령어, 인덱트 컨텐츠, 작업 디렉토리 컨텐츠 모두 변경됨