19강

Index 파일? Status 원리?

Git status: gistory중 index라는 파일

index에는 가장 최신 commit 트리를 클릭해볼 때 내용과 index 내용이 일치한다면 현재 commit할 것이 없는 것을 의미한다.

우리에게 수정된 것을 알려주는 방법: index 파일에 적혀있는 값과 트리의 값이 다르다면 내용이 수정된 것을 알려줌

Git은 index의 내용과 가장 최신 commit의 트리가 가리키는 텍스트의 내용이 다르면, git은 index 내용과 tree가 가리키는 f2.txt의 내용이 다르다면 index에 add 되면 commit 대기상태임을 알 수있음

Staging area: commit 대기 상태

Work tree에서 add -> regist가 되고 repository에 등록이 된다

Working directory – index, staging area, cache – repository

20강: branch

Branch: 나무의 가지.

e.g. 내가 파일을 수정(report.xsl) / report1.xsl / report2.xsl 순차적으로 저장하려 함

다양한 상황에서 사용해야 할 상황이 생기면 나는 report2 -> report2\_client로 바꿀 것

이때, report2을 수정해야 하면 상황이 복잡해지지만, report3 -> report4…로 수정할 것

수정 내용이 없는 이상 계속 뻗어가게 될 것임

client에게 제공하는 문서 내용이 우리 original 문서에 들어가야 할 때: report2\_client2\_report4 = report 5 -> report6.xsl ….

But, git을 사용하면 쉽게 작업 가능

Branch를 만든다: 필요에 의해서 작업이 분기되는 것

Git은 모든 것을 branch로 다룸

21강

Vim f1.txt -> insert -> 입력 -> esc -> :wq

$ git commit -m: vim 안하고 바로 입력 가능

-am: 한번도 add 하지 않은 파일은 자동으로 add가 되지 않음

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

필요 없는거 버리기: 분기(branch)를 해서 버리기

지금까지 작업했던 것을 server에 반영해야 할 때, 여러가지 테스트+문제점이 없는지 체크하는 방법: main, test를 하려는 작업을 분기해야 함

\*master: 현재 내가 master라는 branch를 쓰고 있다. 기본 branch(약속 같은 특별한 이름의 branch)

Git branch exp\_delete(add)/feature: exp라는 branch를 만들어라

Git checkout exp: 숙박할 때 checkout. master에서 checkout 후 exp로 들어가는 것

이후 git log를 보면 현재 branch의 상태를 복사한 것이 나와있음

Vim.txt1에 3 추가 후 git checkout을 하면 1, 2가 보임.