오늘 배울 내용.

1.Git에 사용되는 브랜치(Branch)의 개요와 용도를 배워볼 것이다.

2.git에서 사용될 브랜치를 새로 생성 및 merge를 이용해서 파일 합쳐보기. + 브랜치삭제하는 방법.

1.Git에 사용되는 브랜치(Branch)의 개요와 용도를 배워볼 것이다.

-git이 좋은점 :

1. 깃은 동시에 여러 개발자들이 한가지 프로그램을 동시에 다른 기능을 개발 할수 있도록 Branch(브랜치)라는 것을 제공해준다.

- Branch를 사용함으로써 다양한 작업을 한번에 할 수있기 때문에 빠르게 작업할 수 있다.

- 서로 다른 Branch는 작업을 함에 있어서 서로에게 영향을 받지 않는다는 점에서 마음 놓고 다른 개발 작업을 수행할 수 있다.

2. 브랜치에는 2가지 종류가 있다.

-1. 마스터 브랜치 (Master Branch) 또는 통합 브랜치 :

기본적으로 git 저장소를 생성하면 자동으로 함께 생성이 되고, 이 마스터 브랜치는 일반적으로 배포가 가능한 수준의 안정화된 버전을 포함하고 있다.

1. 실제로 주된 브랜치가 된다. ,배포용 브랜치

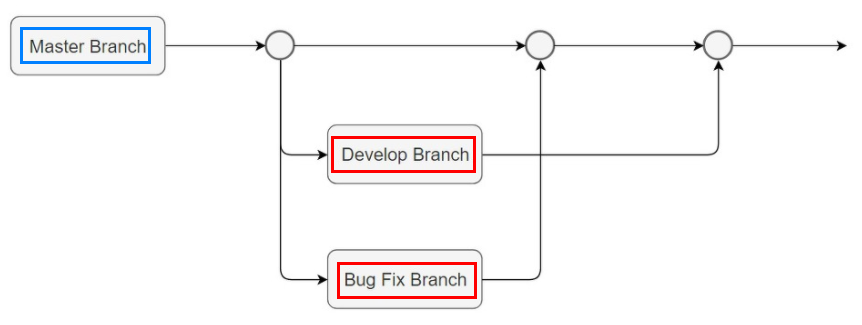
2. 언제나 배포가 가능한 브랜치여야한다. 언제나 안정된 브랜치여야한다.

= 위조건을 만족시 개발의 안정성이 뛰어나진다.

-2. 토픽 브랜치 : 특정한 기능을 위해 만들어진 브랜치로 일반적으로 마스터

(Master)브랜치이외의 브랜치를 말한다.

토픽 브랜치와 마스터 브랜치의 사용 법 .



파란색 =마스터 브랜치

빨간색 =토픽 브랜치

1개의 안정된 브랜치를 이용해서 동시에 개발을진행하고 다른 하나는 버그를 고치기위한 브랜치를 만들어 동시에 일을 처리한다. 이렇게 나중에는 marge(병합)하여 마스터 브랜치가 안정화될 수 있게 한다.

마스터 브랜치는 항상 안정화 되어있기 때문에 개발환경이 안정되어 있다.ㅣ

출처 : <https://ndb796.tistory.com/190?category=1029186>

2.git에서 사용될 브랜치를 새로 생성 및 merge를 이용해서 파일 합쳐보기.

새로 배울 함수 요약

Git branch 현재 어떤 브랜치가 있나 확인하는 식.

Git branch [브랜치이름] 새로운 브랜치를 생성한다.

Git branch -d [브랜치이름] 생성한 브랜치를 삭제한다.

Git branch -D [브랜치이름] 강제로 해당 브랜치를 삭제한다. (merge안한브랜치는 이렇게

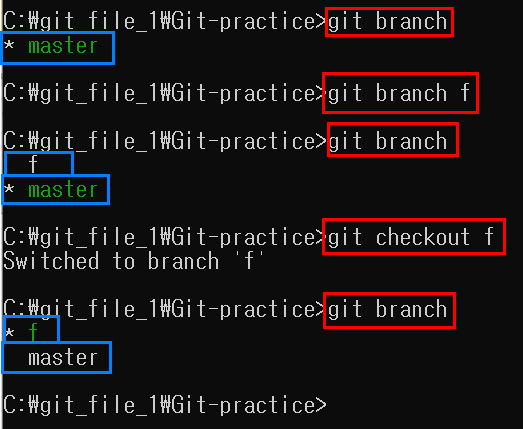
삭제해야한다.)

Git checkout [브랜치이름] 브랜치의 권한(?)을 움직일(선택할) 수있다.

Git merge 다른 브랜치와 병합할 때 쓰는 식.

---------------------------------------------------------------------------------------

2.git에서 사용될 브랜치를 새로 생성 및 merge를 이용해서 파일 합쳐보기.



1. git branch를 사용하여 현재 존재하는 브랜치가 master(첫 번째 파란상자) 한 개라는 것을 알았다.

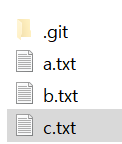
2. git branch f 를 사용하여 새로운 브랜치 ‘f’를 생성시켰다.

3. git branch 를 사용하여 현재 있는 브랜치를 확인결과 f 와 master이 있는 것을 확인하였다.

4.git checkout을 이용해서 f브랜치로 사용할 수 있도록 설정하였다.

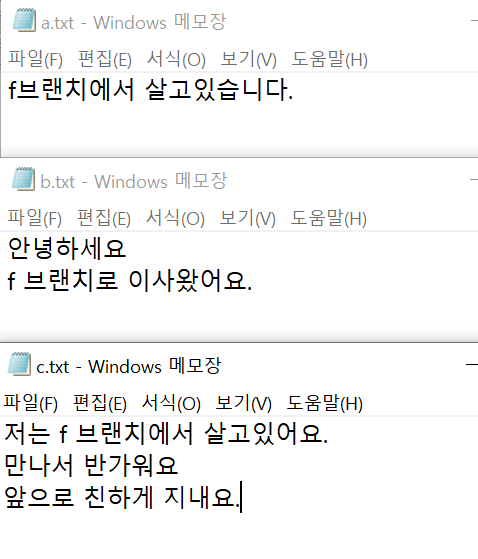
-이로써 나는 master브랜치가 아닌 f 라는 브랜치에 다른 평행우주에서 서로다른 파일을 사용 할 수 있게 되었다.

5. git branch를 통해서 f로 사용설정이 되었는지 확인하였다.

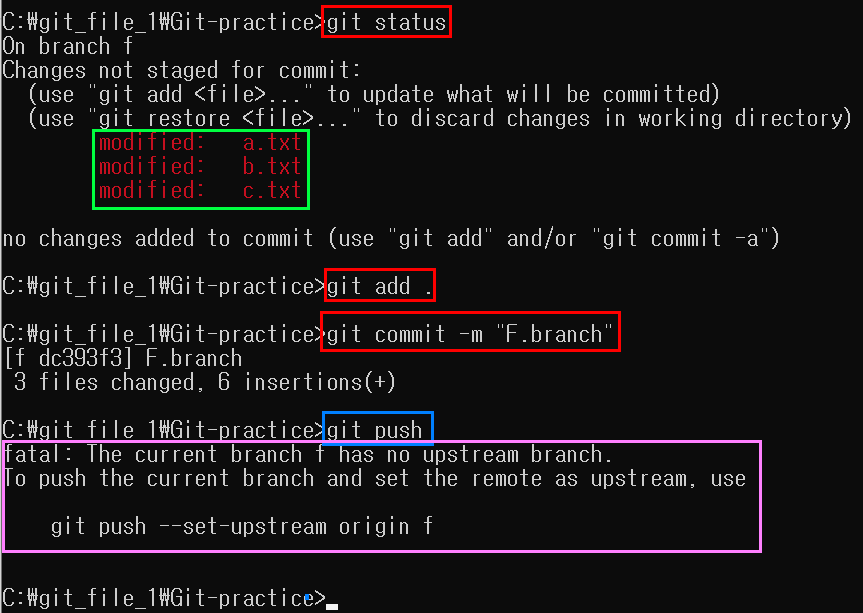


6. 현재 내가 사용할 파일은 이 3가지 인데 내용은 모두 비어있다. 현재 위의 설정 즉

f 브랜치에서 사용할 내용을 이 3가지 파일에 적어보았다.



잘보면 내가 3가지 파일에 각 다른 문장수를 적어 넣었다. 이제 이것을 적용 시켜 볼 것이다.



1. 우선 git status로 현재 storage area에 무엇이 있는지 확인해 보았다.

-확인결과 아까 수정한 내역 3파일(초록상자)이 무사히 있다는 것을 확인하고

2. git add .를 사용하여 모든 파일을 올려주었다.

3. git commit -m 도 당연히 해주어야한다.

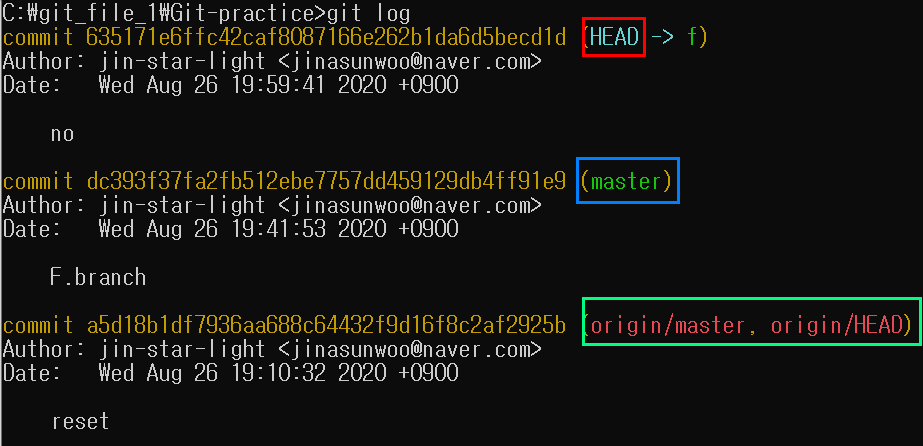
4. 파란상자 git push를 해보았지만 되지 않았다. 그 이유는 내가 사용중인 브랜치가 upstream이 아니기 때문이라는 것이다.

간단히 말해서

-origin파일로 만들지 않으면 올릴수가없다.

-에초에 우리는 git 이 의도한 브랜치는 변경 사항을 만들고 병합용이기 때문에

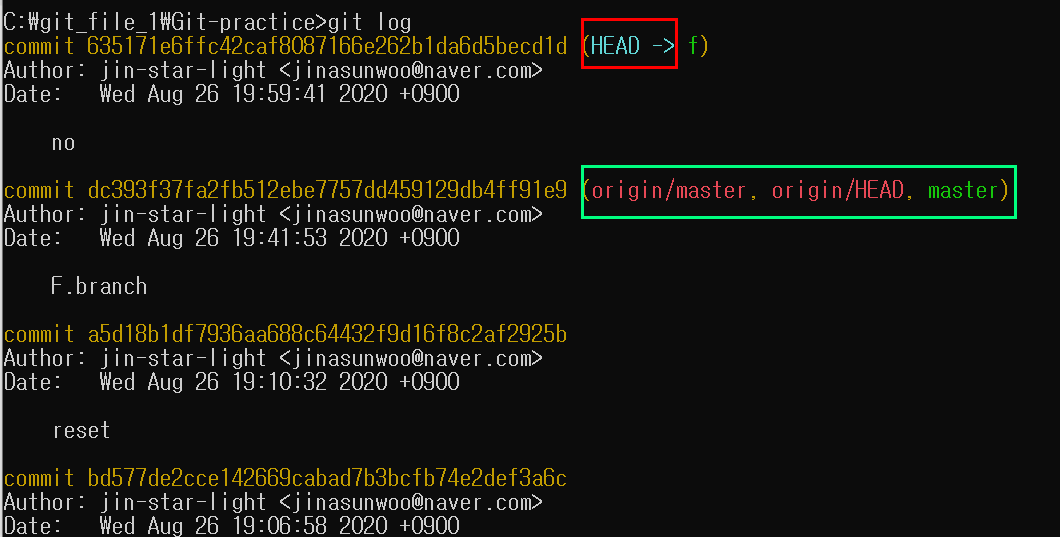
결론 : 다음부터는 git push가 아닌 merge로 마스터 브랜치에 병합하자.



(예시사진 - 중간에 사진 빠트렸어서 다시 만듬 ㅠㅠ commit 메시지는 다르니 신경쓰지말자)

여기서 잘보면 HEAD가 f 를 가르키고 있다. 이것은 현재 내가 사용하는 브랜치를 가르키는 것이 바로 HEAD 이고 , master 은 원래 저장소를 생성하면 생기는 기본적인 브랜치이다. 근데.. 위사진은 초록색 상자는 바로 Remote Repository의 저장된 commit을 말하는것인데… 위 의 경우는 내가 실수로 branch master을 git push하지 않아서 생긴 상황이다.

정리해서 말하자면 master 브랜치의 변경사항이 Remote Repository의 저장소까지 올라가지 않아서 master와 origin master가 따로 노는 것이다.



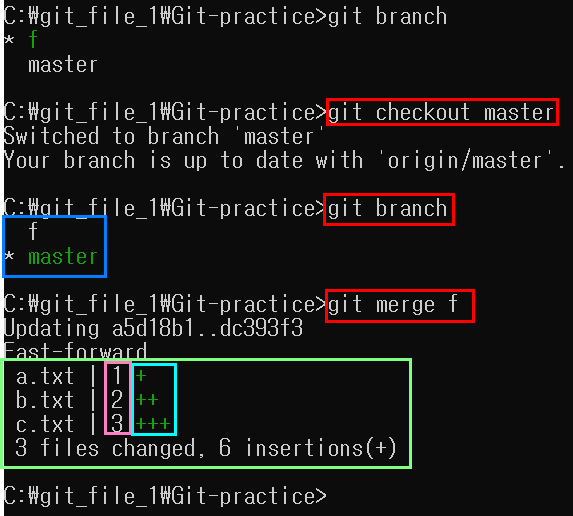
branch를 master로 바꾸어 git push를 하고 다시 돌아왔다.

무사히 master을 origin/master로 설정했다.

#여기서 알수있는사실#

1. git log를 통해서 어떤 브랜치가 master로 설정되어있는지 알수 있다.

2. git log를 통해서 파일이 Remote Repository로 올라갔는지 확인 할 수 있다.



1. git checkout master을 하여 master 브랜치로 이동해준다.

2. git branch를 확인해보면 무사히 master로 이동됨을 알 수 있다.

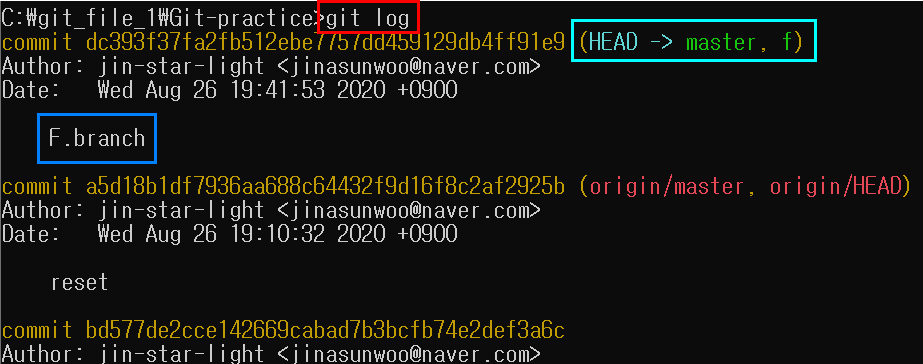
3. git merge f를 통해서 f브랜치와 master 브랜치를 통합시켜준다.

-초록상자를 잘보면 각 텍스트별 얼마나 추가되었는지 보인다.

-핑크상자는 몇줄이 추가되었는지 적혀있고

- 민트 상자를 보면 +는 한줄을 말한다.

-참고로 사진에는 없지만 -는 줄이 삭제됨을 의미한다.



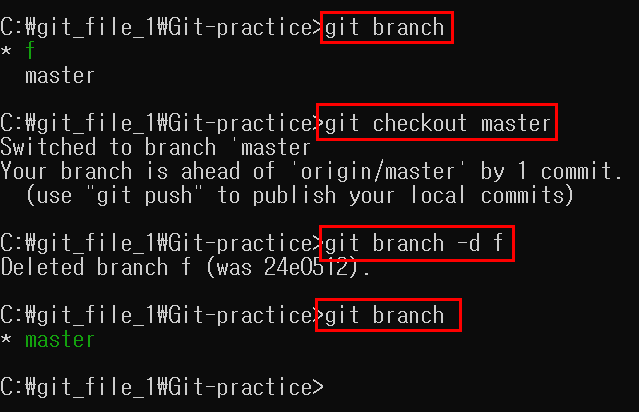
Git log를 통해서 확인해보아도, master브랜치와 f브랜치가 병합이 되었으며

HEAD (현재 master 브랜치에 있음을 알 수 있다.)

-------------

이렇게 브랜치 병합까지 배워보았다.

이번에는 쓸모없어진 브랜치를 삭제하는 방법을 배워보겠다.



1. 우선 현재 어떤 브랜치에 속해있는지 확인을 한다. (git branch를 통해서 확인)

2. git checkout 으로 master 브랜치로 이동해주어야한다!

-만약 master브랜치가 아닌 해당 삭제할 브랜치에 선택된 채로 삭제하려면 오류가 뜬다!

3. git branch -d f를 적어서 삭제를 한다.

-git branch -D는 강제로 삭제하는 식이다.

(merge안한브랜치는 변경사항이 master에 적용되지 않은 브랜치이기 때문에 이렇게

삭제해야한다.)

4. 삭제가 잘 되었는지 git branch를 통해서 확인한다.