//1-1

GIT은 리눅스 소스코드를 잘 관리하기 위해서 개발하였으며 현재는 수만은 소프트웨어의 소스코드를 깃을 통해서 관리하고 있다.

GIT의 3대 목적 version(버전), backup(백업), collaborate(협업)

GIT의 종류

TortoiseGit 윈도우에서만 사용할 수 있다.

GithubDesktop 기능이 심플하고 배우기 용이하다.

Sourcetree 복잡하지만 자유로운것이 장점이다.

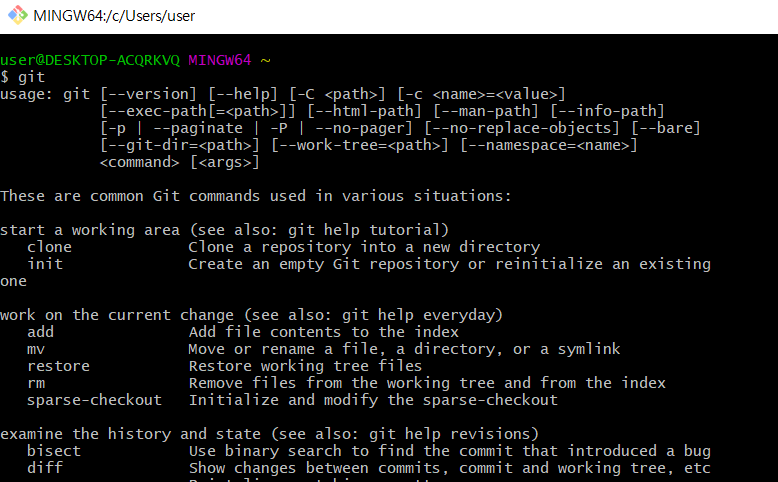
Git 명령어를 통해서 Git를 제어할수있다.

GIt CLI 버전관리

git이라고 하면 command line에서 사용되는 프로그램이다.

설치 방법

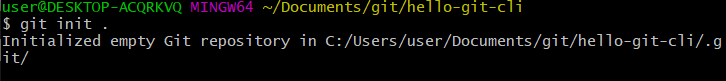
https://git-scm.com 에 접속하여 운영체제에 맞게 설치를 하면 된다.

설치 후 git bash를 실행하여 콘솔창이 나오는데 git을 입력했을 때 에러가 없다면 잘 실행되는 것이다.

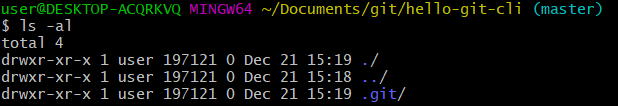
저장소 만드는 법

git bash에서 자신이 지정한 폴더에 접속한 후

git init를 입력하면 git이 현재 폴더를 버전관리 할 수 있다.

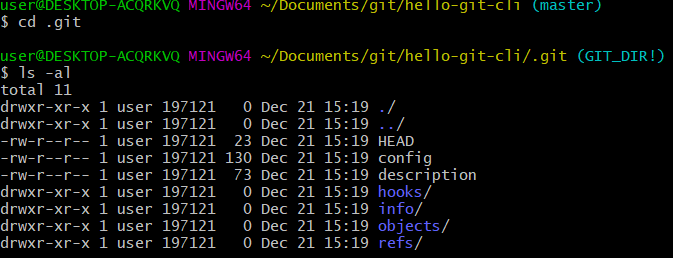


이후 폴더에는 .git이라는 폴더가 생긴다. / ls -al 폴더내 파일 확인



여러 디렉토리가 있는데 이것이 .git의 부모 디렉토리이다.

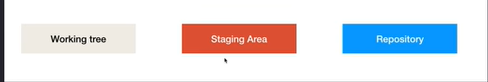
프로젝트 파일들을 보관하는 디렉토리에서 생성되는 변화들을 버전으로 만드는데 .git디렉토리 안에 각자 알맞게 저장하게 된다.(.git이라는 디렉토리는 지우면 안된다.)



Git에서 저장되는 곳이 Repository

Working tree 파일을 만들고 수정하는데 버전으로 만들어지기의 전단계

Staging Area 깃은 Staging Area에 있는 파일을 버전으로 만들어준다.



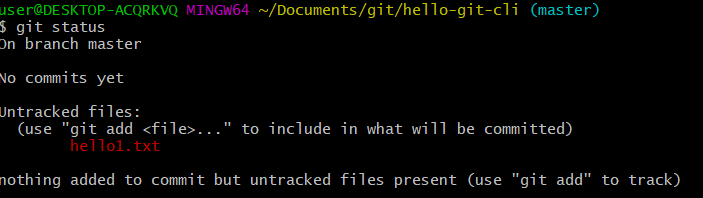
파일생성 후 화면이 출력되는데 내용을 입력 후 Crtl+x 후 y를 누르면 저장이 된다

EMB000040a83116

git status를 입력하면 git의 상태를 보여준다.

No commits yet = 아직 버전이 없음

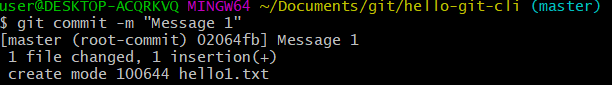
Untracked files = 아직 추적이 되지않는 파일

hello1.txt는 git이 아직 관리를 하지 않음

Staging Area에 올리기 위한 것이 add이다.

EMB000040a83118

git commit를 입력하면 Repository로 올라가게 된다.

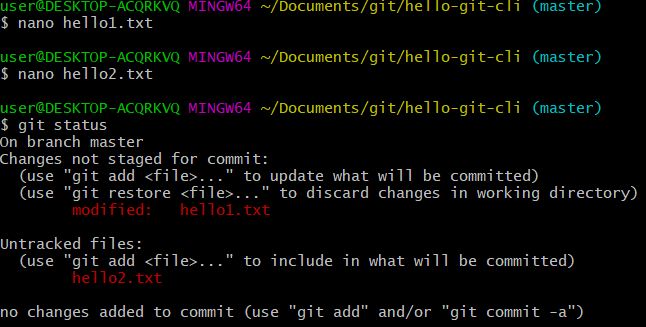


여러개의 파일을 하나의 버전으로 만들기

hello1.txt에 내용을 추가 / hello2.txt을 생성 후 status를 보면

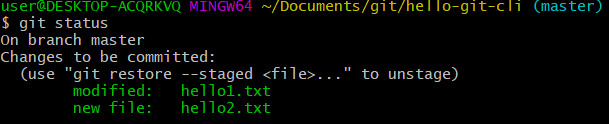
hello1.txt는 Staging Area에 올라와있지 않지만 hello2.txt는 Ubtracked files로 되어있음

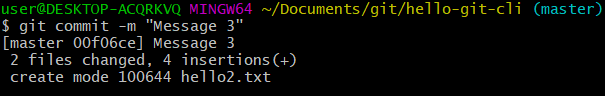
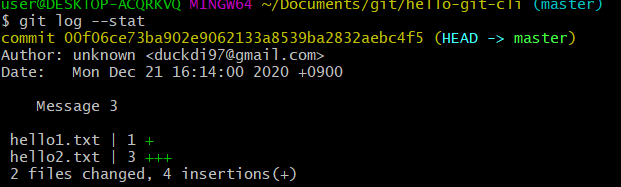
hello1.txt는 이미 버전관리를 하였기 때문에 git이 기억을 하지만 hello2.txt는 git이 알지못한다.



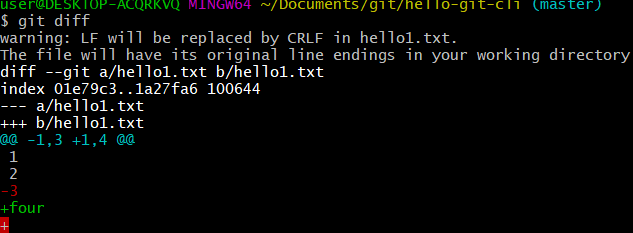
Staging Area에 올리기 위해서는 git add를 입력한다..

status를 확인하면 둘다 Staging Area에 올라간다.



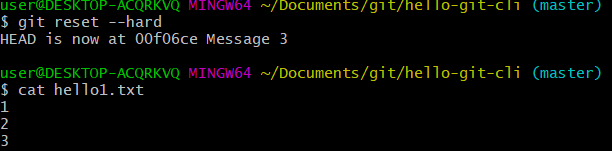
commit 후 Repository로 가지만 어떤 파일이 있는지 알수 없는데 git log –stat를 통해서 알 수 있다.

hello1.txt에서 내용을 추가하여 수정한 후 “git diff“를 입력하면 초록색은 새버전, 빨간색은 구버전인 것을 알려준다.



버전을 되돌리고 싶다면 git reset --hard를 입력한다.

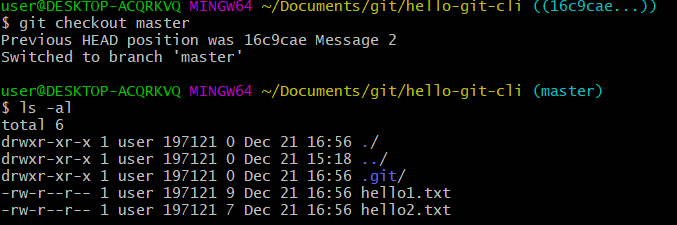
git log -p를 입력하면 버전마다 수정된 정보를 알려준다.



log에서 돌아가고 싶은 버전의 commitID를 복사하여 git checkout id를 입력하면 돌아갈 수 있다.

EMB000040a83120

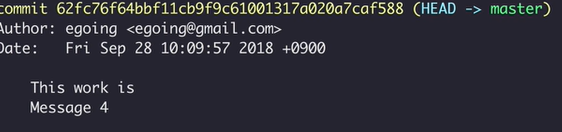
git checkout mastar을 입력하면 가장 최신버전으로 돌아간다.



Git commit –am “Message”에서 –am의 뜻은 add와 commit Message를 뜻한다.

그러나 한번도 add가 되지 않았던 파일은 사용할 수 없다.

여기서 commit만 입력하면 더 많은 내용을 저장할 수 있게 된다.



git reset --hard commitID는 해당commitID로 되겠다 라는 것이다.

EMB000040a83123

git reset --help를 입력하면 모드에 대한 정보를 얻을 수 있다.

EMB000040a83124

현재 log , R3,R4를 추가하고 git revert commitID를 입력

revert는 상태를 되돌리는 작업이다.(R3)

처음 버전으로 돌아가기 위해서는 역순으로 하나씩 revert를 해야한다 R4->R3->M2->M1

EMB000040a83125

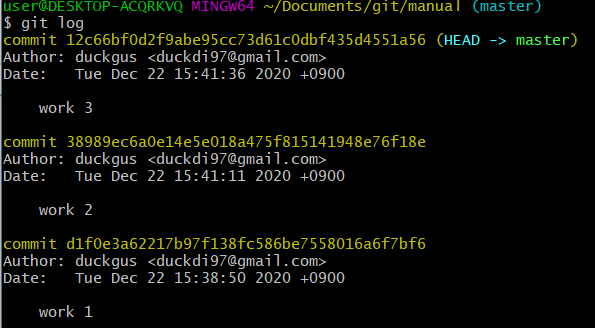
//1-2

Git CLI - Branch & Conflict

Branch(나뭇가지) / 같은 꼬리에서 나왔지만 서로 다른 버전들을 쓰고 있는 것을 말함

Conflict(충돌) / 2개의 같은 파일이 같은곳이 수정되었을 때 충돌이 일어남 -> 수정요청

밑 로그와 같이 내용을 추가하여



Branch 사용법

branch를 사용하면 현재버전의 파일을 복사하여 저장함

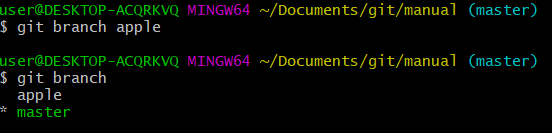
git log --all // 앞으로 만들게 될 모든 Branch가 보인다

git log --graph // 시각적으로 표현됨

git log -- oneline // 버전이 한줄로 나옴

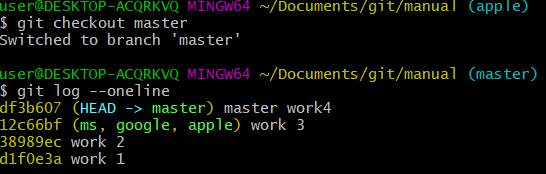
git branch // Branch의 목록을 표시 앞에 별이 어디에 속해있는지 알려줌

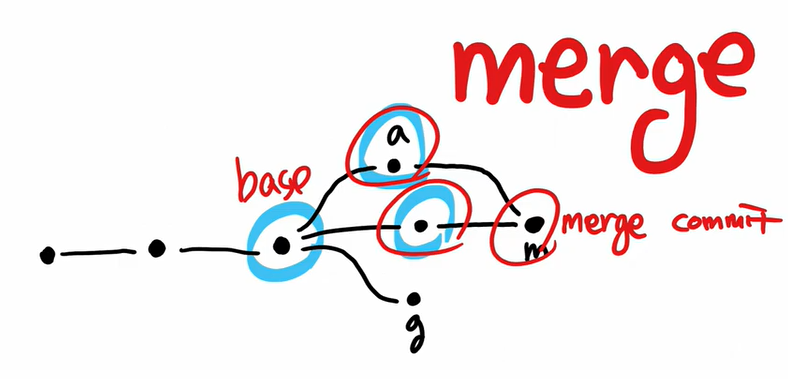
git branch “name”을 통하여 apple, google, ms를 추가한다.



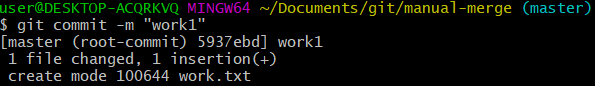
git checkout apple를 입력하면 HEAD는 apple를 가르키며. master는 살아있음

git checkout master을 입력하면 master쪽으로 다시 이동하게 된다.



Merg (병합)

work.txt 파일을 추가하고 commit를 해준다. 이후 master.txt파일을 하나 더 만들어준다.



commit 메시지를 수정하고 싶은 경우

git commit --amend 입력 후 수정하면 된다.

EMB000040a8312c

git checkout o2를 입력

EMB000040a8312d

o2.txt 파일을 하나 추가한다.

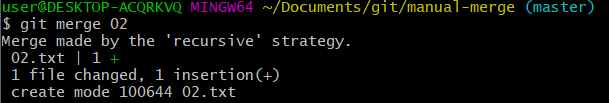
EMB000040a8312e

master에 o2를 병합.

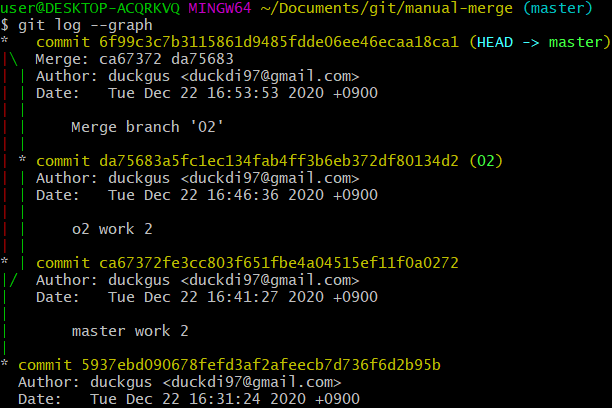
1. master branch상태가 되어야한다.

EMB000040a8312f

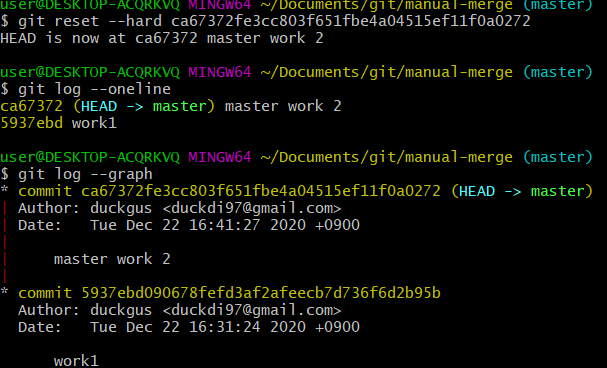
2. 합치고 싶은 branch를 가져온다.



git log --graph를 입력하여 병합된 것을 확인.

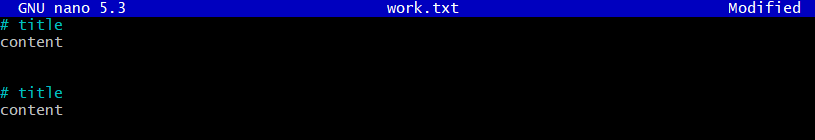


되돌리고 싶은 경우 git reset --hard commitID



같은 파일의 통합

새로운 파일을 추가한다.

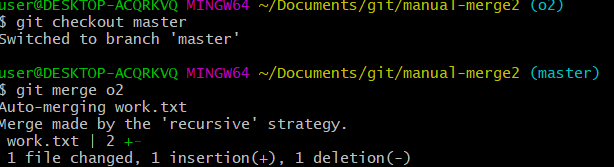


branch o2를 생성 후 work.txt의 위쪽을 변경 후 commit

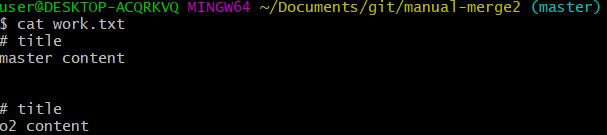
checkout o2로 설정 후 work.txt파일의 아래쪽을 수정 후 commit 한다.

EMB000040a83135

o2의 내용을 master에 병합하고 싶다면 checkout master로 바꾼다.

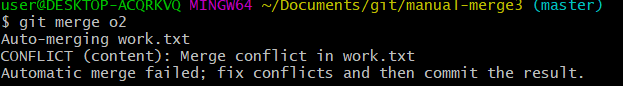
이후 git merge o2를 입력.

확인

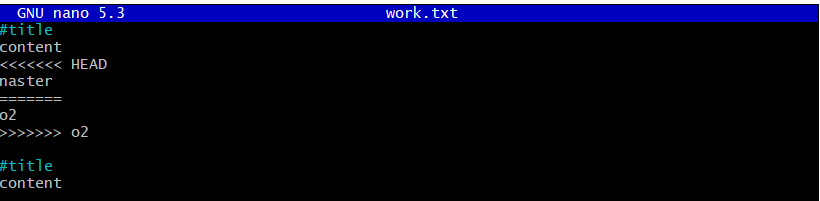


두 개의 branch가 같은 이름의 파일의 같은 곳을 수정을 했을 때는 실패한다.

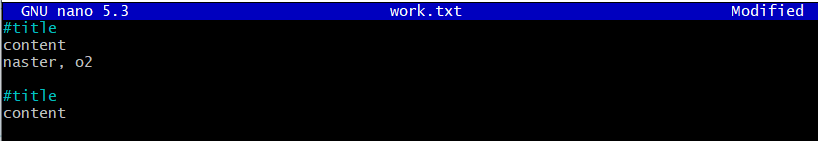
실패 -> 충돌을 해결한 후에 다시 시도

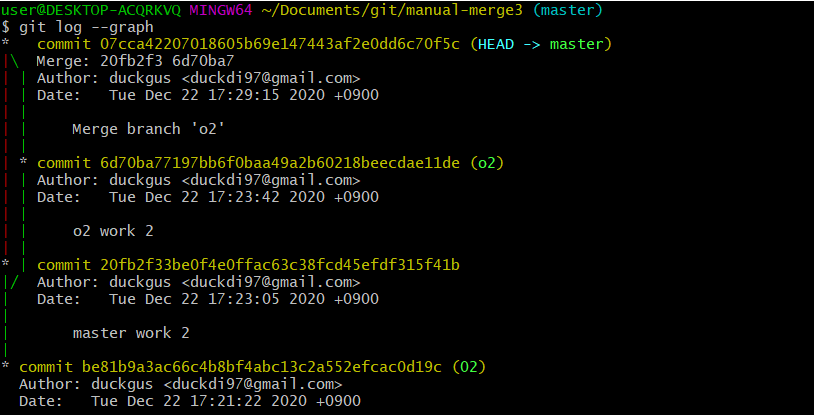


====== -> 구분자를 기준으로 위쪽 HEAD 아래쪽 o2



내용 수정 후 git commit -am "commit Message " 실행



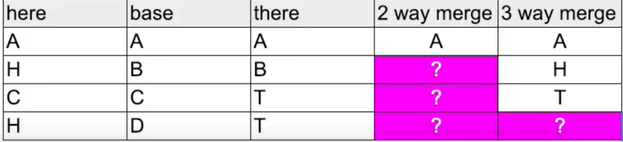


//1-3

Conflict, 3Way Merge

어떤 파일에 A, B, C, D 의 내용이 있다고 가정 여기서 Branch를 생성

H, T는 내용수정, 3way merge를 사용하면 2way merge에 비해서 훨씬더 많은 내용을 자동화하여 병합할 수 있다.



외부도구를 이용한 병합

git bash를 이용하여 위와같이 work.txt파일을 만들고 branch를 이용하여 here, there를 생성

here의 위치에서 there를 병합하려고 하면 충돌이 발생한다.

충돌을 해결하기 위해서 git mergetool를 입력하면 병합을 전문적으로 실행해주는 도구가 나온다.(p4Merge등등) 입력 후 충돌이 일어나지 않게 수정한다.

EMB000040a8313d

orig는 백업된 파일이며 rm work.txt.orig를 통하여 백업파일을 지울 수 있다.



EMB000040a8313f

GIT CLI - Backup

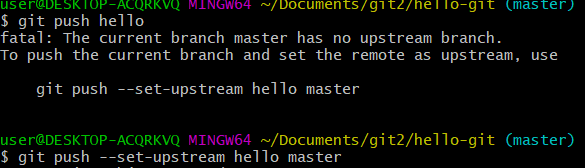
간단하게 할 수 있는 것은 Dropbox, Google Drive등과 같은 서비스를 이용하는 것.

저장소 만들기 <https://opentutorials.org/course/3841/22819>

git remote add (별명) github저장소주소

EMB000040a83140

git push를 하게되면 처음에는 에러가 뜨지만 나오는 것을 카피해서 붙여넣기를 하면됨

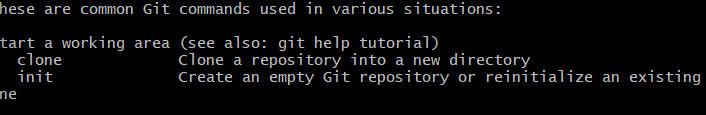


설정 후 git push를 하게되면 정상적으로 파일이 올라간다.

EMB000040a83142

로컬저장소를 만드는 방법은 2가지 이며 첫 번째는 init를 사용하는 것

두번째는 clone를 사용하는 것이다.



git clone 주소를 입력하면 현재 디렉토리에 복사되며 뒤에 mydir(이름)을 입력하면 폴더생성후 저장한다.

EMB000040a83144

원격저장소에서 받아올때는 git pull을 사용하면 받아올 수 있다.

EMB000040a83145

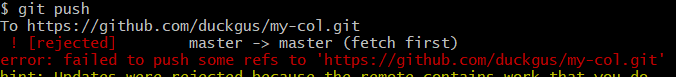
//1-4

GIT CLI - 협업

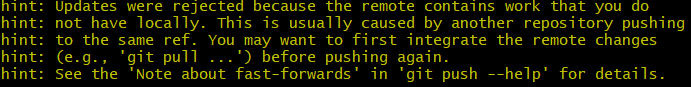
github에서 settings - collaboration에서 같이 할 사람을 초대할 수 있다.

a와 동일한 b가 있으며 a에서 파일을 수정 후 psuh

b에서 pull을 하지 않은 상태에서 파일을 수정 후 push를 하면 거절하며, 해결방안을 제시해준다.



이때 b에서 pull을 하게되면 충돌이 발생하게 되는데 이것은 툴을 사용하거나 내용을 수정하여서 push해주면 해결된다.



pull vs fetch

git fetch는 원격 저장소만 업데이트 하는 것이다.

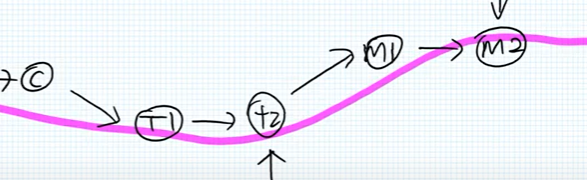
git pull = git fetch + git merge

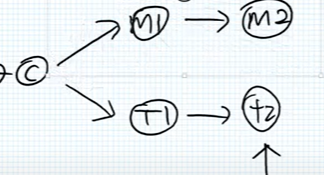
여기서 git pull을 하거나 git merge origin/master을 통하여 받거나 합칠 수 있다.

git fetch -> git log

EMB000040a83148

GIT CLI Cherry-pick / Rebase

Cherry-pick는 특정한 com mit를 픽업해서 다른 commit에 붙일 수 있는 기능

Rebase 병렬로 이루어진 작업을 하나의 작업이 이루어 진 것처럼 만드는 것

ex) master -> init, m1, m2 / topic init, t1, t2, t3

m2에서 Cherry-pick로 t2를 선택하면 t1->t2로 바뀔 때의 변화만 가져오는 것이다.

master에서 git cherry-pick commitID를 입력하면 master에는 t2가 추가됨

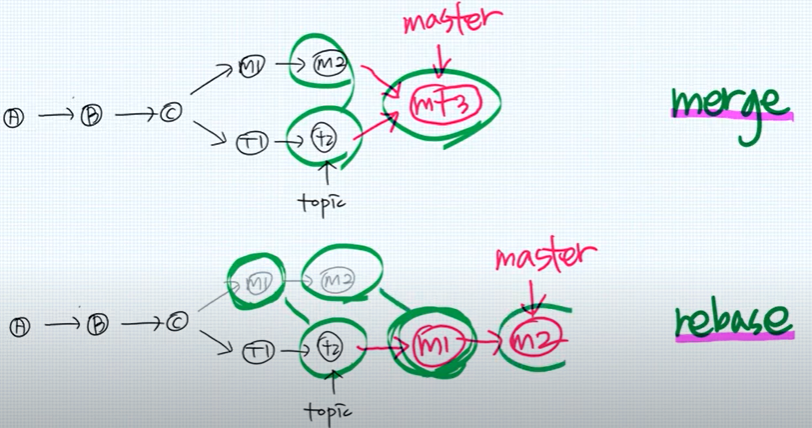
Rebase

ex) master에서 m1의 base를 t2로 바꾸는 것이다.

git rebase topic를 입력하면 topic가 가르키고 있는 commitID로 베이스를 옮기겠다 라는 것

rebase는 버전을 하나하나 재작성 해서 붙인다.

여기서 cherry-pick로 만든 mt3와 rebase로 만든 m2는 결과가 같다.



GIT CLI - PULL Request

pull request, merge request

exp branch를 master branch로 병합 해주세요, 이것을 검토해주세요 라는 의미를 팀원에게 표명한 것이다.

master branch의 안정성을 높이것이 핵심이다. 2가지 방법이 존재함

1. 원격저장소가 나의 저장소이거나 원격 저장소에 직접적으로 접근할수 있을 때

병합할 때 다른사람의 의견을 받고 싶을 때 쓰는 방식

2. 원격저장소에 대한 권한이 없을 때 -> 원격 저장소를 복제하고 작업을 함 이후 직접적인 권한이 있는 사람이 가져가서 원래의 원격저장소에 적용하는 것

[https://www.youtube.com/watch?v=Ru9qv-tHj7I&list=PLuHgQVnccGMBXv1OKe3Hn3Jq6F735-uWm&index=3](https://www.youtube.com/watch) 참고