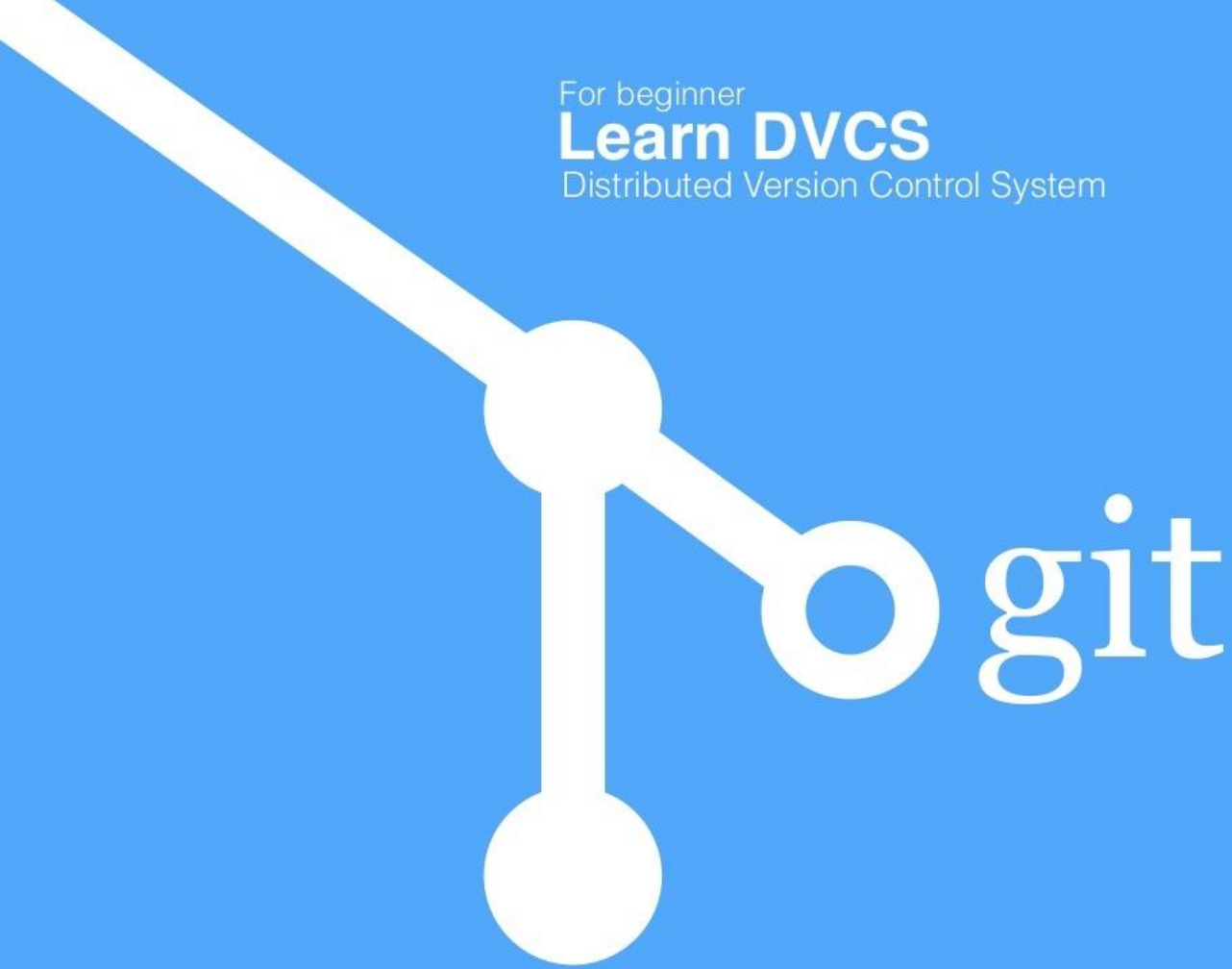
For beginner

Learn DVCS

Distributed Version Control System



텍스트 파일이 하나 있다

memo.txt

하고싶은 것?

수정한 내용을 기록해두고 싶다

memo.txt

어떻게 할까?

수정할 때 마다 다른 파일로 저장

memo.txt

memo1.txt

memo2.txt

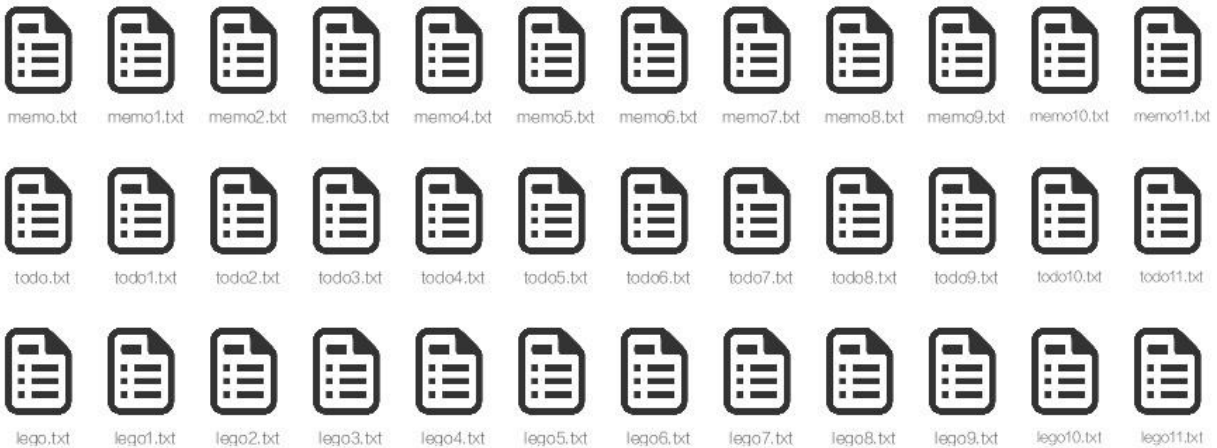
시간이 지나면

흠.....


memo.txt
memo1.txt
memo2.txt
memo3.txt
memo4.txt
memo5.txt
memo6.txt
memo7.txt
memo8.txt
memo9.txt
memo10.txt
memo11.txt

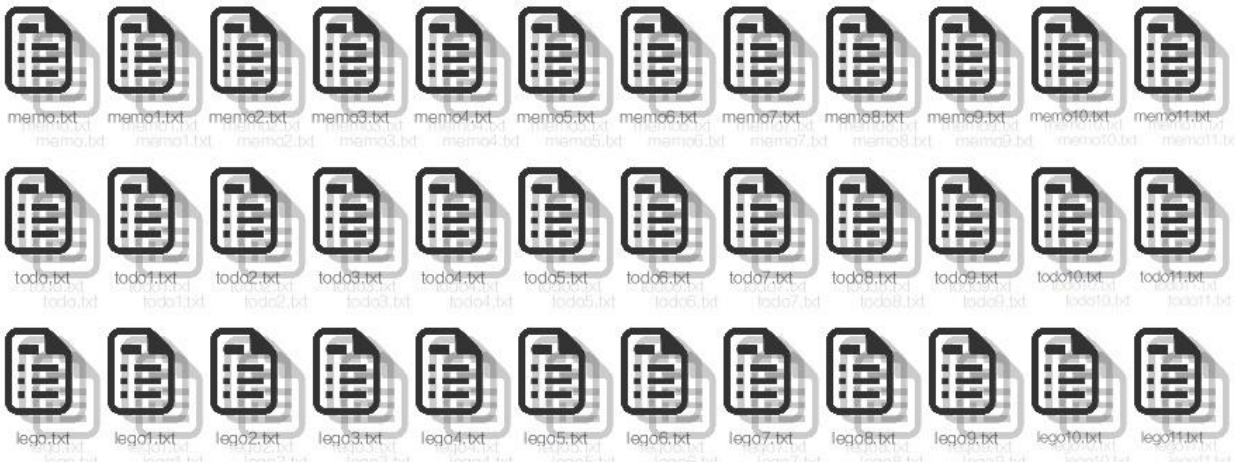
파일이 하나가 아니라면?

흠.....



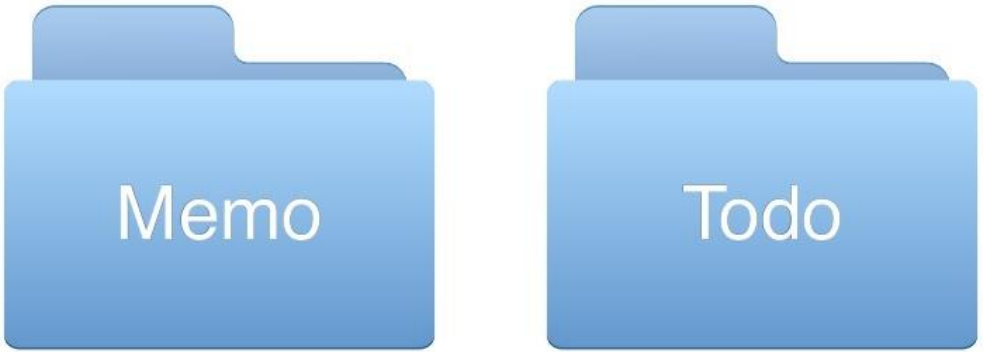
수정하는 사람이 여러명이면?

OTL



폴더로 정리하면 되지 않을까?

이런 생각 하시는 분이 없기를 바랍니다



어떻게 할까?

이런일을 전문으로 하는 소프트웨어가 있다? 없다?

Yes! 있다!

첫 번째 문제로 돌아가봅시다

텍스트 파일이 하나 있고,
수정한 내용을 기록해두고 싶다

"컵라면 1박스 구입" 이라는 수정이 발생

“수정”의 단계를 어느 정도까지 기록해야할까?

저장할 때 마다?

컵(Save)라(Save)면(Save)... 이런식으로
저장했다해도 모두 기록해야할까?

규칙 1

"수정"의 단계는 "의미"를 기준으로

의미를 어떻게 알지?

SW가 의미를 인식할 수
없으니 정할 수 있는 사람 즉,
사용자가 정하는 것으로 하자

규칙 1

"수정"의 단계는 "의미"를 기준으로

commit

이 파일의 수정이 끝났다
"이 파일의 의미있는 수준의 수정 작업이 끝났음"
이라고 git 에게 알려주는 일

세 번의 커밋

commit 3

2014년 7월 10일 17:00

컴라면 구매 할 일 추가

@홍길동

commit 2

2014년 7월 9일 10:03

쓰레기 봉투 구매 할 일 추가

@홍길동

commit 1

2014년 7월 6일 12:42

맥주 구매 할 일 추가

@홍길동

수정한 내용을 간략하게 표현하기 위한 “커밋 메세지” 기록

두번만 할 수 도 있겠죠?

커밋의 기준을 정하는건 사용자.
커밋의 단위를 작게 할 수록 변경 내용의 기록
수준은 정밀해지며 파악할 수 있는 정보도 많아진다

commit 2

2014년 7월 10일 17:00

쓰레기 봉투, 컴라면 구매 할 일 추가

@홍길동

commit 1

2014년 7월 6일 12:42

맥주 구매 할 일 추가

@홍길동

규칙 2

하나의 커밋(Commit)은 여러개의
파일이 포함될 수 있다

“수정의 의미”를 확장하자

하나의 파일을 수정하는 것,
하나의 수정 작업이 여러 파일에 영향을
줄 수 있는 것. 의미라는 관점에선 하나의
수정이라고 할 수 있다

커밋은 수정의 단위. 수정은 여러 파일에 걸친 변경 내역의 묶음일 수 있다.

규칙 3

파일을 수정하지 않고
새로운 실험을 해 볼 순 없을까?

월,화,수,목,금 시간표 파일이 있는데
이 파일들을 변경하지 않고 새로운 월,
화,수,목,금 일정을 만들어 보고 싶다

가능할까?

규칙 3

상태를 저장하는 공간을 만들 수 있다

branch

상태란 모든것이다. 파일, 파일의 내용, 커밋 정보 등 git 이 관리하는 모든 것을 의미한다

git 의 시작

지정한 폴더의 변경 내용을 추적하기위한 준비

init

선택된 폴더의 변경 내용을 추적하기 위해 git 저장소를 만든다. 저장소가 만들어지면 git은 지정한 폴더를 포함하여 하위 폴더의 모든 변경 내용을 커밋 단위로 추적한다.

git 의 시작

지정한 폴더의 변경 내용을 추적하기위한 준비

master

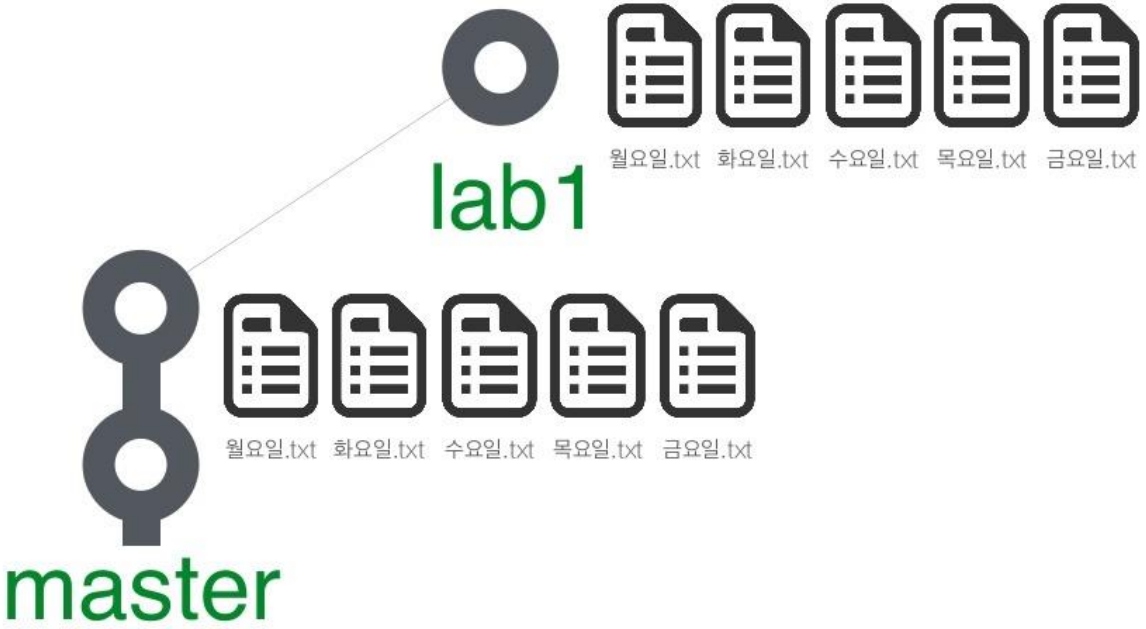
init 명령으로 저장소가 만들어질 때
git은 master라는 이름의 브랜치를
기본으로 생성하고 이 후 커밋 내용
을 브랜치 기준으로 저장한다

커밋은 브랜치 공간에 기록된다

새로운 브랜치 만들기

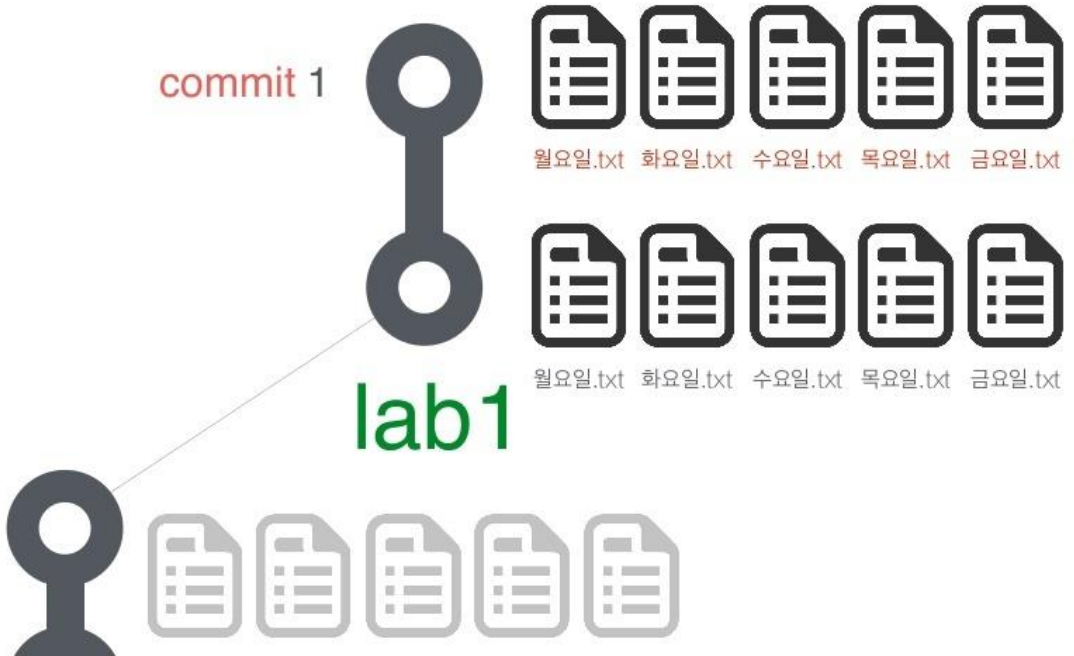
파일을 수정하지 않고 새로운 실험 하기

branch lab1

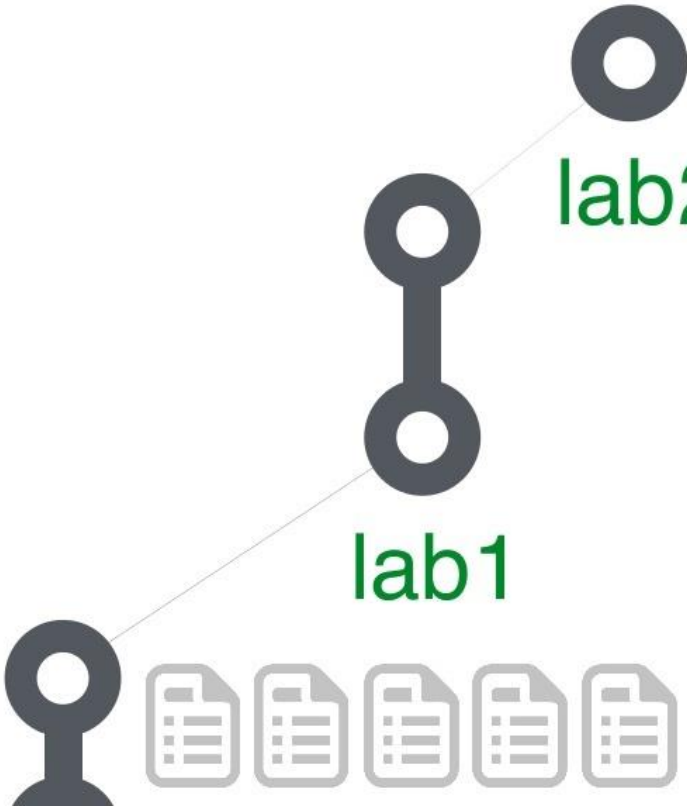


완전히 독립된 브랜치 lab1

새로 만든 브랜치 lab1 은 master와 완전히 동일한 상태를 가진 공간. lab1 브랜치에서 수정을 한 후 커밋하면 그 변경사항은 lab1에만 기록되며 master 브랜치에는 어떤 영향도 주지 않음



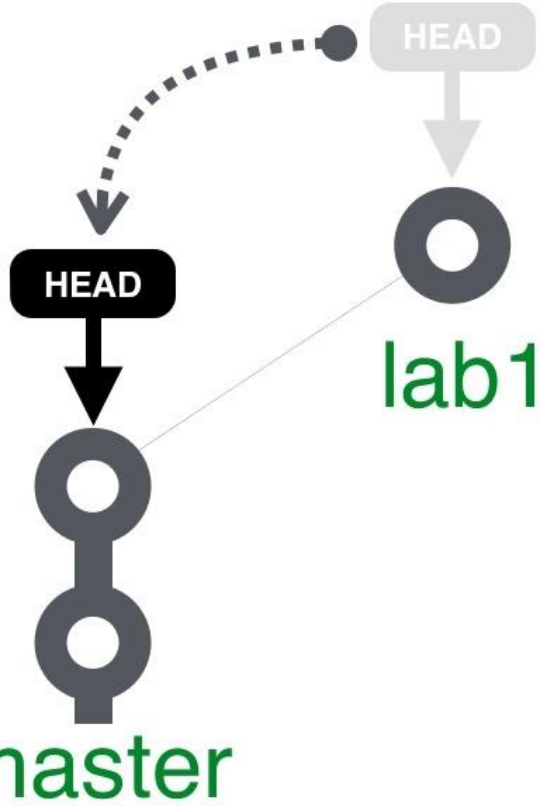
원하는 만큼 브랜치 생성 가능



git 은 매우 빠르게 새로운 독립 공간인 브랜치를 원하는 만큼 만들 수 있다. 브랜치 이름은 이름 정의 규칙 내에서 사용자가 원하는 형태로 작명할 수 있다

실험 중 다른 브랜치로 돌아가야한다면?

checkout master

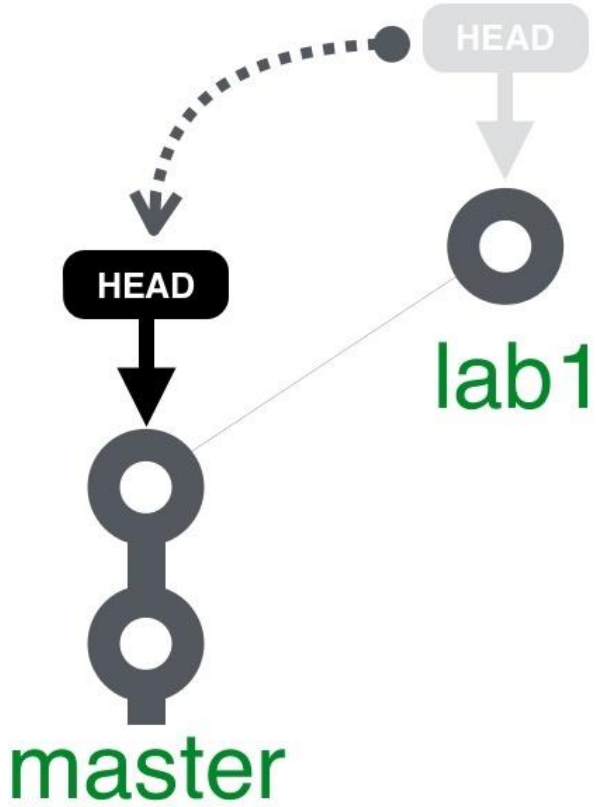


원한다면 언제든 다른 브랜치(작업 공간)로 이동할 수 있다. 브랜치는 마지막 커밋 상태를 유지한다.

작업 중인 위치를 가르키는 가상의 커서가 존재하는데 이를 git 에서는 HEAD 라 한다.

checkout master

브랜치를 이동하는 이유가 뭘까?



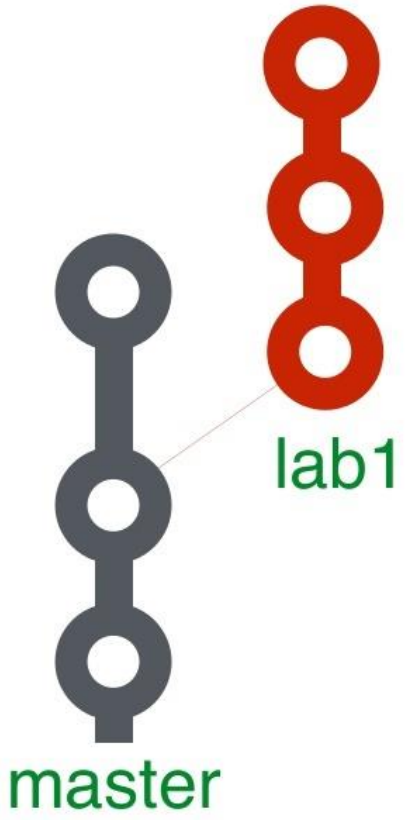
시나리오

새로운 실험을 하기 위해 lab1을 만들고 열심히 작업하고 있다. 그런데 master 브랜치에 내용을 변경할 일이 발생했다. 어떻게 해야할까?

정답

master 브랜치로 이동하여 변경 작업을 처리한 후 커밋. 다시 lab1 브랜치로 돌아와 하던 실험을 계속 한다.

실험 종료. 선택의 순간



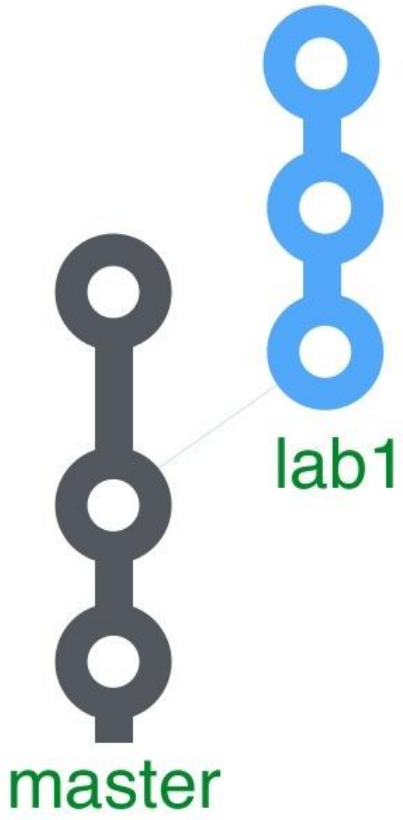
실험 실패

lab1에서 진행했던 실험이 예상과 달리 필요없는 작업이 되었다. 어떻게 하면 될까?

정답

master로 이동 후 lab1 브랜치를 삭제한다. lab1의 모든 기록이 제거된다. 기록 보관 차원에서 삭제하지 않아도 아무 문제 없다

실험 종료. 선택의 순간



실험 성공

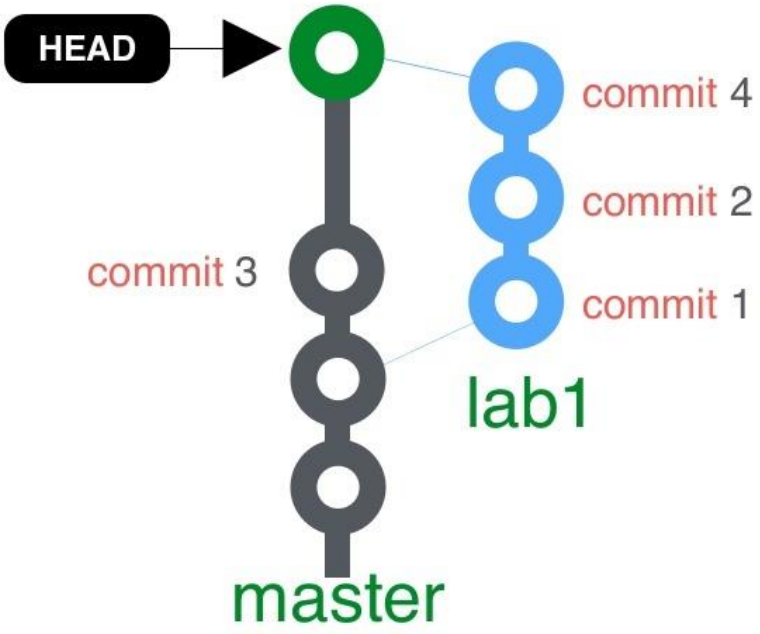
lab1에서 진행했던 실험이 성공적으로 끝났다. 실험의 결과를 master 브랜치에 옮기려한다. 어떻게 하면 될까?

정답

lab1 브랜치의 내용을 마스터 브랜치와 병합 (merge) 한다.

브랜치와 브랜치의 병합(Merge)

merge lab1



병합 결과

master 브랜치에 lab1 브랜치를 병합하면 git 은 lab1 브랜치의 내용과 master 브랜치의 commit 3 의 내용을 포함하여 두 브랜치를 병합한다

변경 내용에 따라 파일 내용이 변경되고 때론 파일이 삭제될 수도 있으며 추가될 수 도 있다.

정리

commit 수정 내역을 사용자 기준의 의미로 기록한다

branch 완전히 독립된 작업 공간을 만들 수 있다

checkout 독립된 작업 공간인 브랜치를 자유롭게 이동할 수 있다.

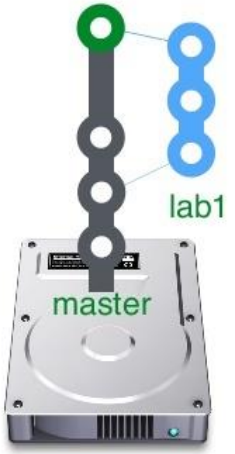
merge 브랜치와 브랜치간 내용을 병합할 수 있다

아직 해결되지 않은 문제

어떻게 다른 사람과 함께 작업할 수 있을까?

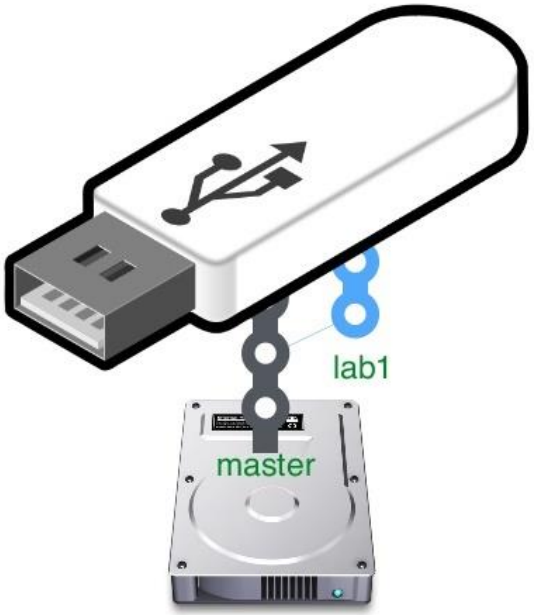
우리는 git 저장소를 하나 가지고 있다.
내 컴퓨터에 있기 때문에 나만 접근할 수 있다.

동료와 함께 작업 하려면?



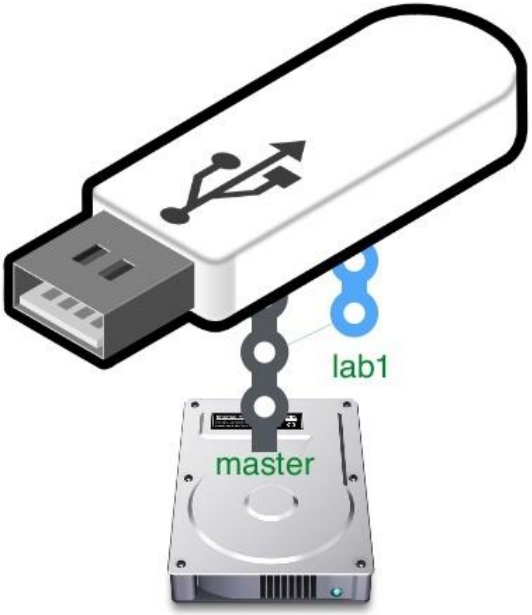
동료와 함께 작업하려면?

복사를 시도해 볼 수 있겠다



동료와 함께 작업하려면?

복사를 시도해 볼 수 있겠다

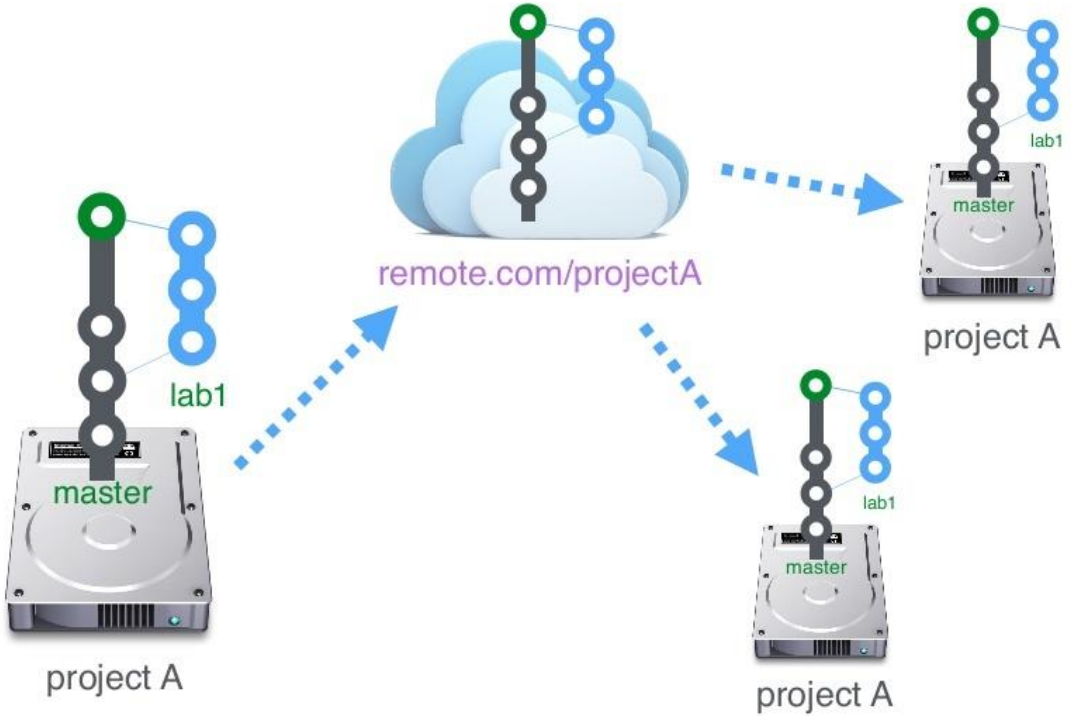


복사는 단방향이다. 한번 주고 동료가 작업한 결과를 돌려 받기 위해선 너무나 많은 어려움이 예상된다

git은 "리모트 저장소"를 지원한다

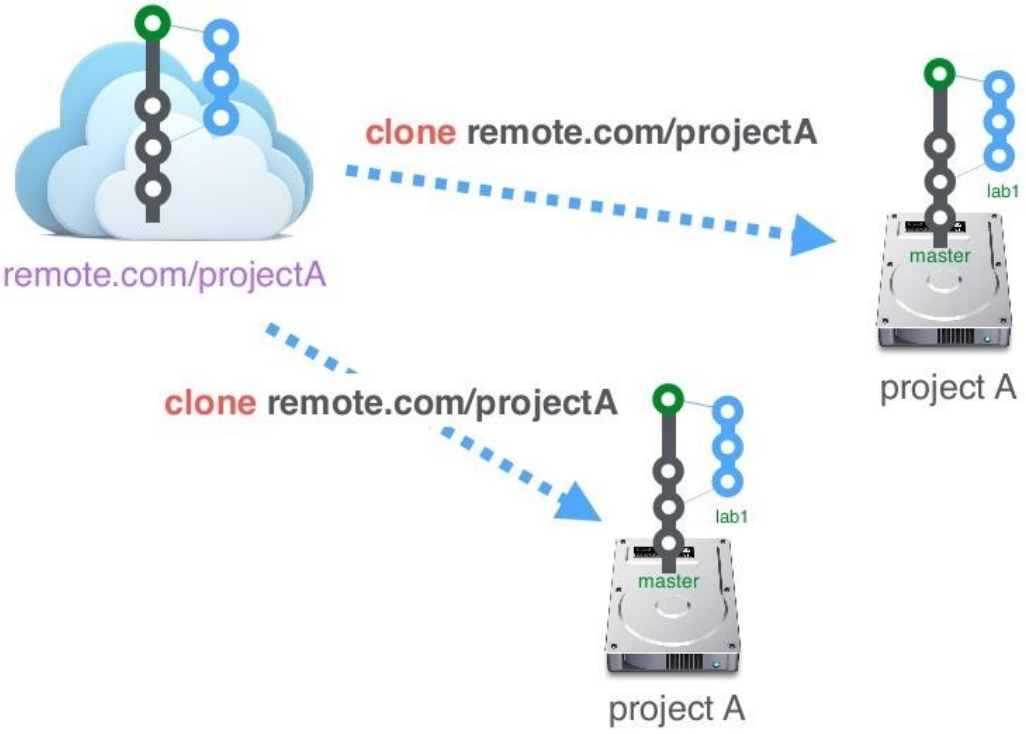
리모트 저장소가 있다면?

리모트 저장소 또한 원본 git 저장소와 동일한 저장소이다
리모트 저장소를 경유하여 함께 작업할 동료도 완전히 동일한
저장소를 다운로드 받을 수 있으며, 동일한 방식으로 작업 할 수 있다



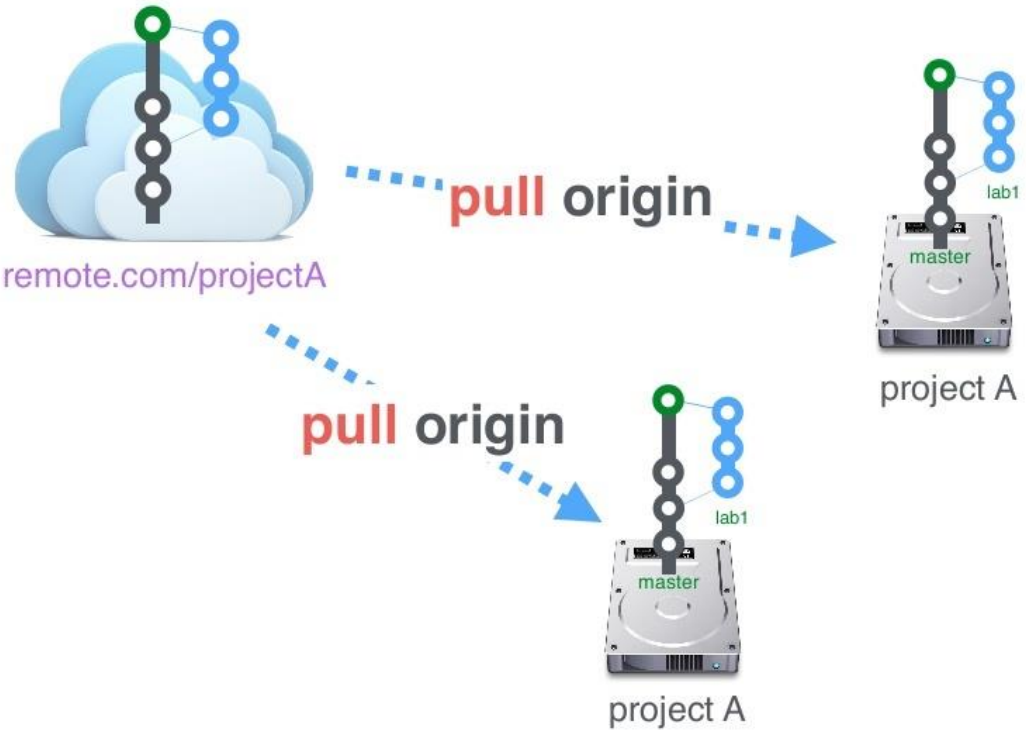
리모트 저장소에서 저장소 다운로드

리모트 저장소에서 처음으로 git 저장소를 다운로드 받는 것을
복사본을 만든다는 의미로 clone 이라 한다.



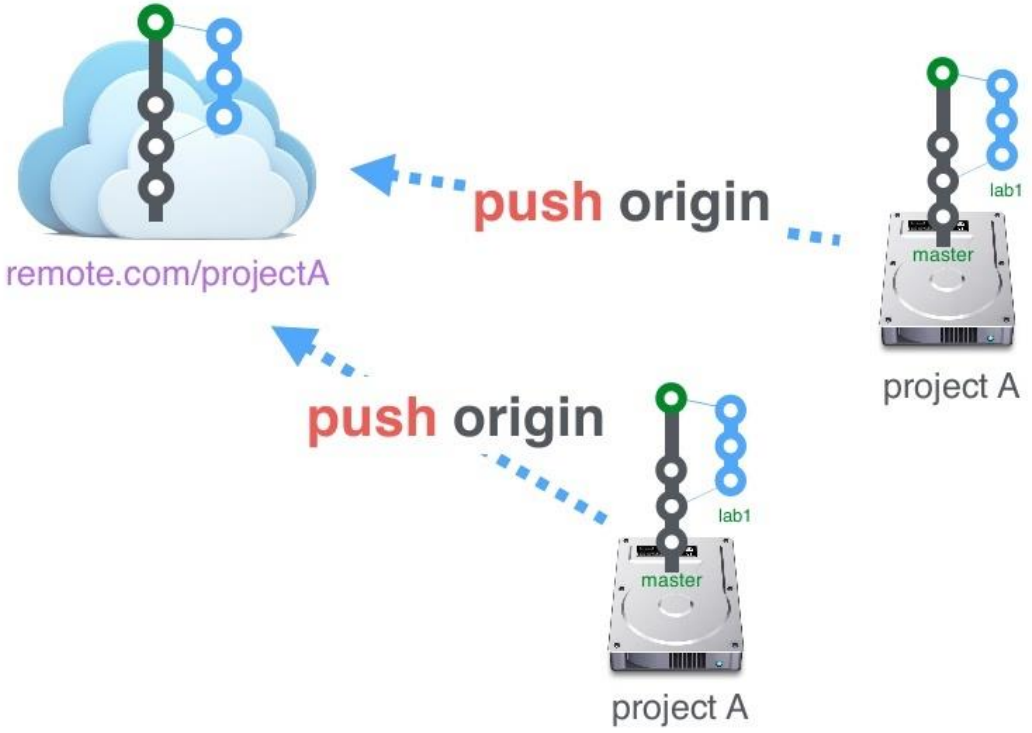
리모트 저장소의 변경 내용 업데이트

리모트 저장소의 변경된 내용을 로컬(내 컴퓨터) 저장소에 적용하는 작업을 Pull 이라 한다. 이때 브랜치 병합과 같은 병합이 발생한다.



내 저장소(로컬 저장소)의 변경 내용 리모트로 전송하기

로컬(내 컴퓨터) 저장소에서 작업한 내용을 리모트 저장소로 보내는 작업을 Push라 한다. 함께 작업하는 동료에서 변경사항을 전송하기 위해선 리모트 저장소를 경유해야 한다는 것을 알 수 있다.



git 사용법을 학습할 수 있는 다양한 콘텐츠

Git 메뉴얼 (한글)
http://git-scm.com/book/ko

Learn Git Branching
http://pcottle.github.io/learnGitBranching/

Code School - Try git
https://try.github.io/levels/1/challenges/1

Atlassian git tutorial
https://www.atlassian.com/git/tutorial