브랜치

ACKANG

브랜치(Branch)

나뭇가지, 지사, 분점 등 줄기 하나에서 뻗어 나온 갈림길을 의미

저장 공간 하나에서 가상의 또는 저장 공간을 만드는 것

브랜치 작업

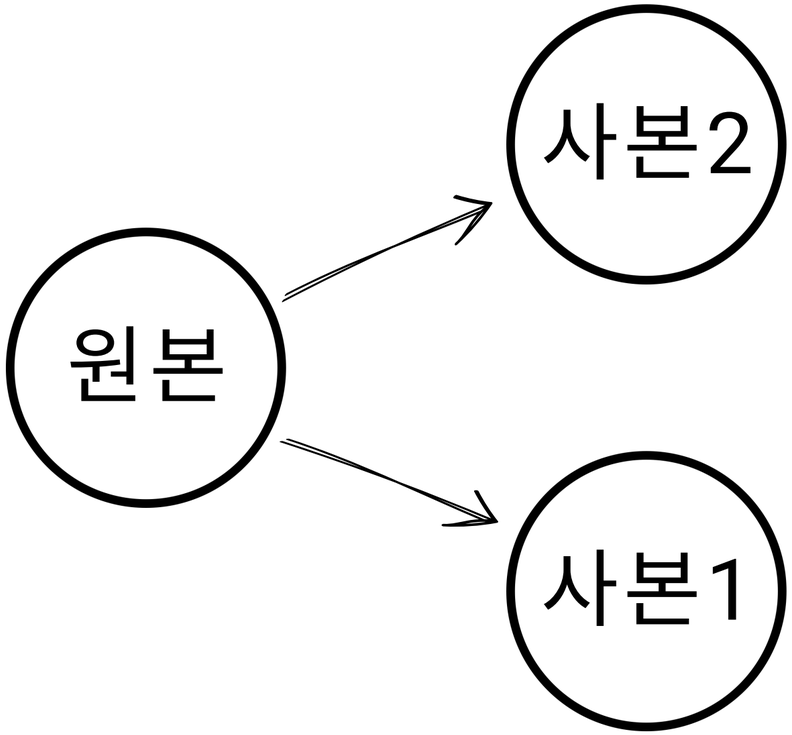
original

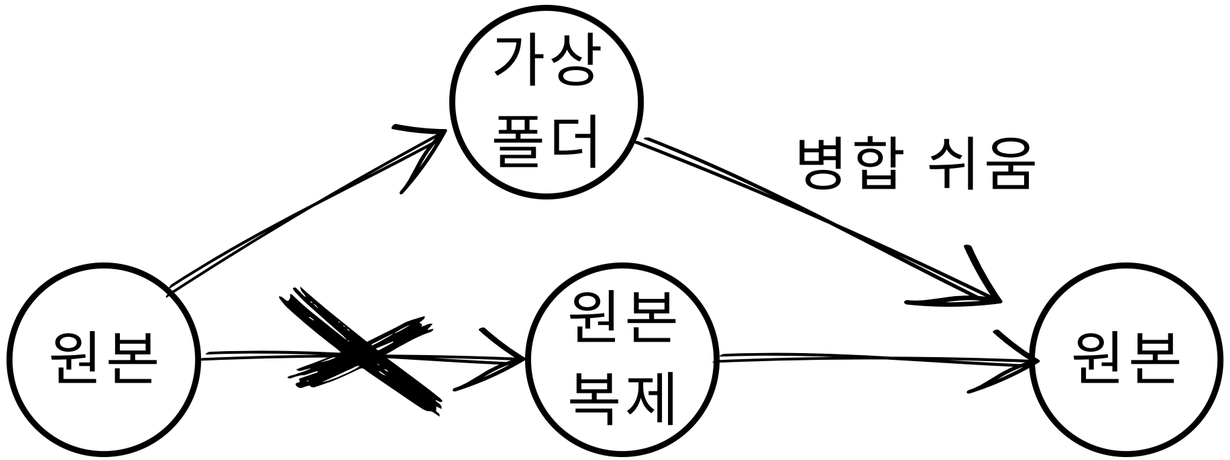
보통 새로운 기능을 추가하거나 많은 변경이 예상될 때는
작업 폴더를 통째로 복사하고, 폴더 이름을 변경

copy

안정적인 기존 코드는 남겨두고, 복제된 작업 폴더에서
도전적인 작업들을 하기 위해 코드를 분리

깃을 사용하면 프로젝트 작업 폴더를 복사하지 않고 기존
코드와 분리해서 작업 할 수 있다





깃 브랜치의 특징

깃 브랜치는 기존 폴더를 복제하는 것과 다르게 가상 폴더를 사용하여 개발 작업을 구분

가상폴더

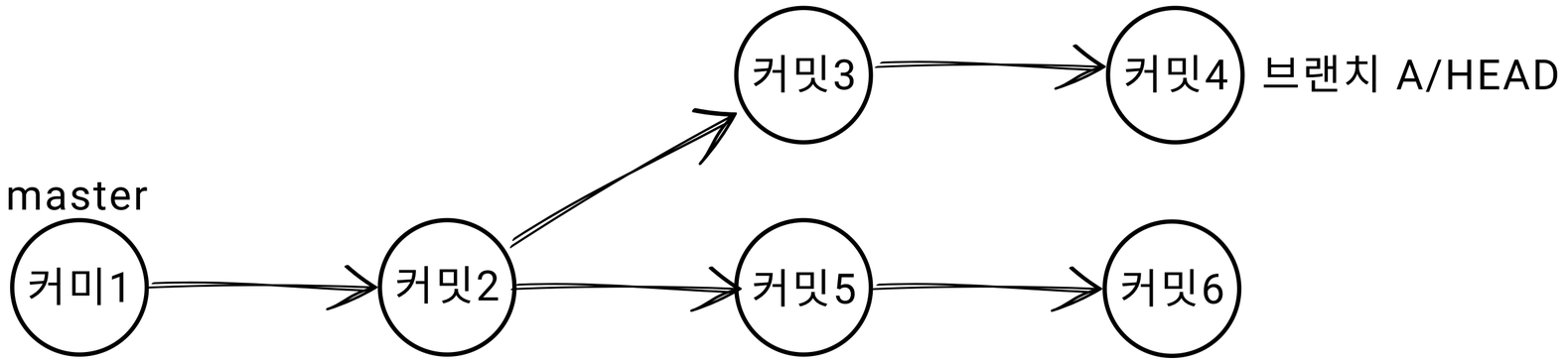
깃의 브랜치는 작업 폴더를 복사하지 않고, 가상 폴더를 생성
외부적으로는 물리적인 파일 하나만 있는 것으로 보임
가상폴더는 물리적으로 복제된 구조보다 유연하게 처리할 수 있음

독립적인 동작

브랜치를 이용해 가상폴더로 독립적인 개발 작업을 수행 가능
깃의 브랜치는 규모가 큰 코드 수정이나 병합을 처리할 때 유용

빠른 동작

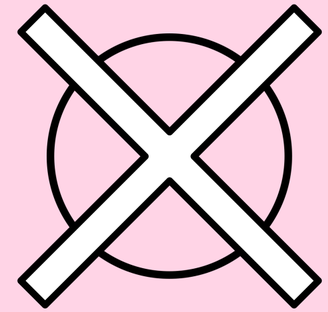
깃의 브랜치는 다른 버전 관리 도구보다 가볍고, 브랜치 전환이 빠른 것이 특징
깃은 Blod 개념을 도입하여 내부를 구조화 해 브랜치를 변경할때 포인터를 이동하여 빠르게 전환 가능
다른 번전 관리 시스템은 폴더 전체를 복사하는 반면 깃은 해시 파일 하나만 만들면 됨



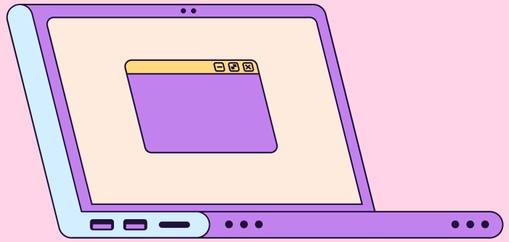
브랜치 생성

- 브랜치는 공통된 커밋을 가리키는 지점
- 브랜치를 생성한다는 의미는 기존 브랜치 또는 커밋에 새로운 연결 고리를 하나 더 만드는 것과 같다
- 새 브랜치를 생성하면 포인터만 있는 브랜치가 생성
- 일반적으로 브랜치를 생성하면 현재 커밋을 가리키는 head를 기준으로 생성
- 이것으로 기존 브랜치의 영향을 주지 않고 새로운 작업을 할 수 있다

브랜치를 생성하면 기본 main 브랜치 외에 사용가자 직접 정의한 사용자 브랜치를 생성함
브랜치는 깃에서 또 하나의 개발 분기점을 의미
새로운 개발 분기점이 필요할 때는 브랜치를 추가로 생성 할 수 있다



기본 main 브랜치

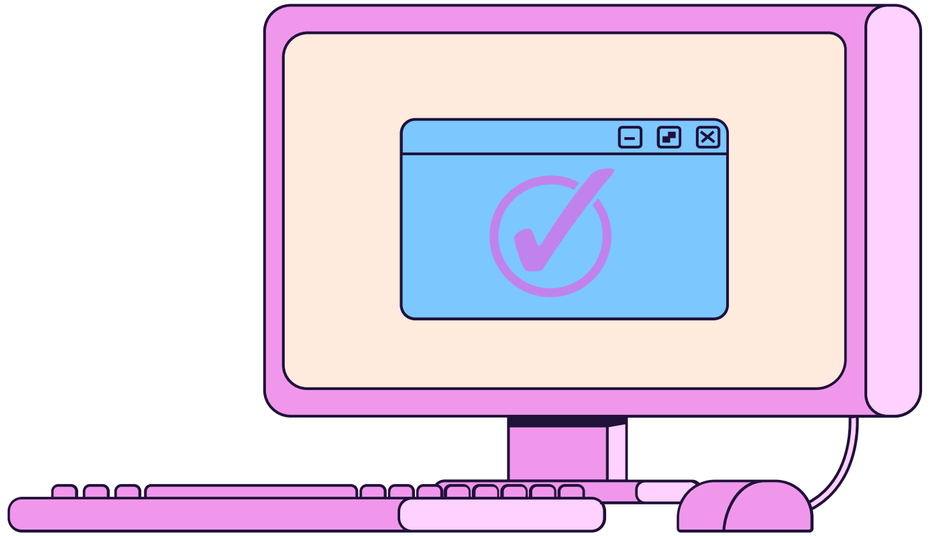


브랜치 생성

https://github.com/perneean/OSS_6/blob/main/6%EC%9E%A5%20%EB%B8%8C%EB%9E%9C%EC%B9%98%201~4(%EC%97%BC%EC%A7%84%EC%84%AD).md

체크아웃(checkout)

- 현재 브랜치를 떠나 새로운 브랜치로 들어간다는 의미
- 깃은 하나의 워킹 디렉터리만 갖고 있기 때문에 한 브랜치에서만 작업과 커밋을 할 수 있다.
- 다른 브랜치에서 작업하려면 반드시 브랜치를 변경하여 워킹 디렉터리를 재설정 해야 함

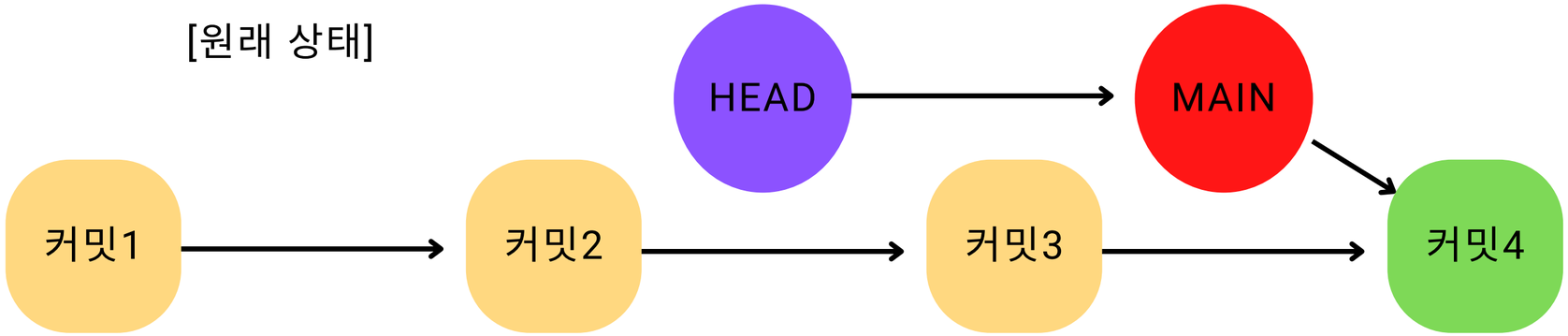


체크아웃

https://github.com/perneean/OSS_6/blob/main/6%EC%9E%A5%20%EB%B8%8C%EB%9E%9C%EC%B9%985%2C6%2C7(%EC%9A%B0%EC%8A%B9%EC%9B%90).md

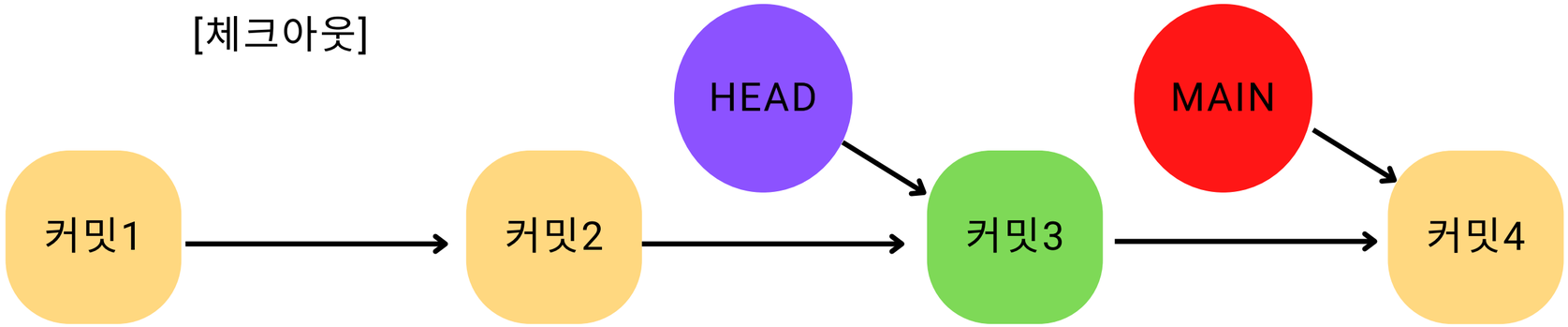
브랜치 동작 원리

[원래 상태]



HEAD는 브랜치의 마지막 커밋을 의미. 브랜치가 이동하면 HEAD 포인터도 함께 이동.

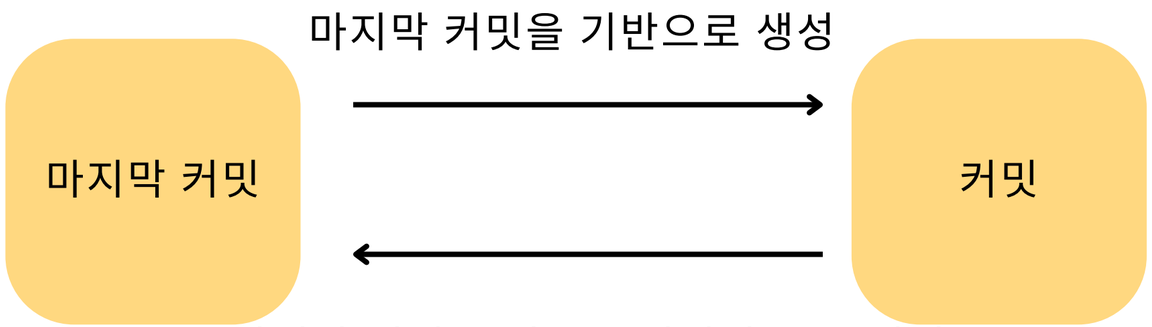
[체크아웃]



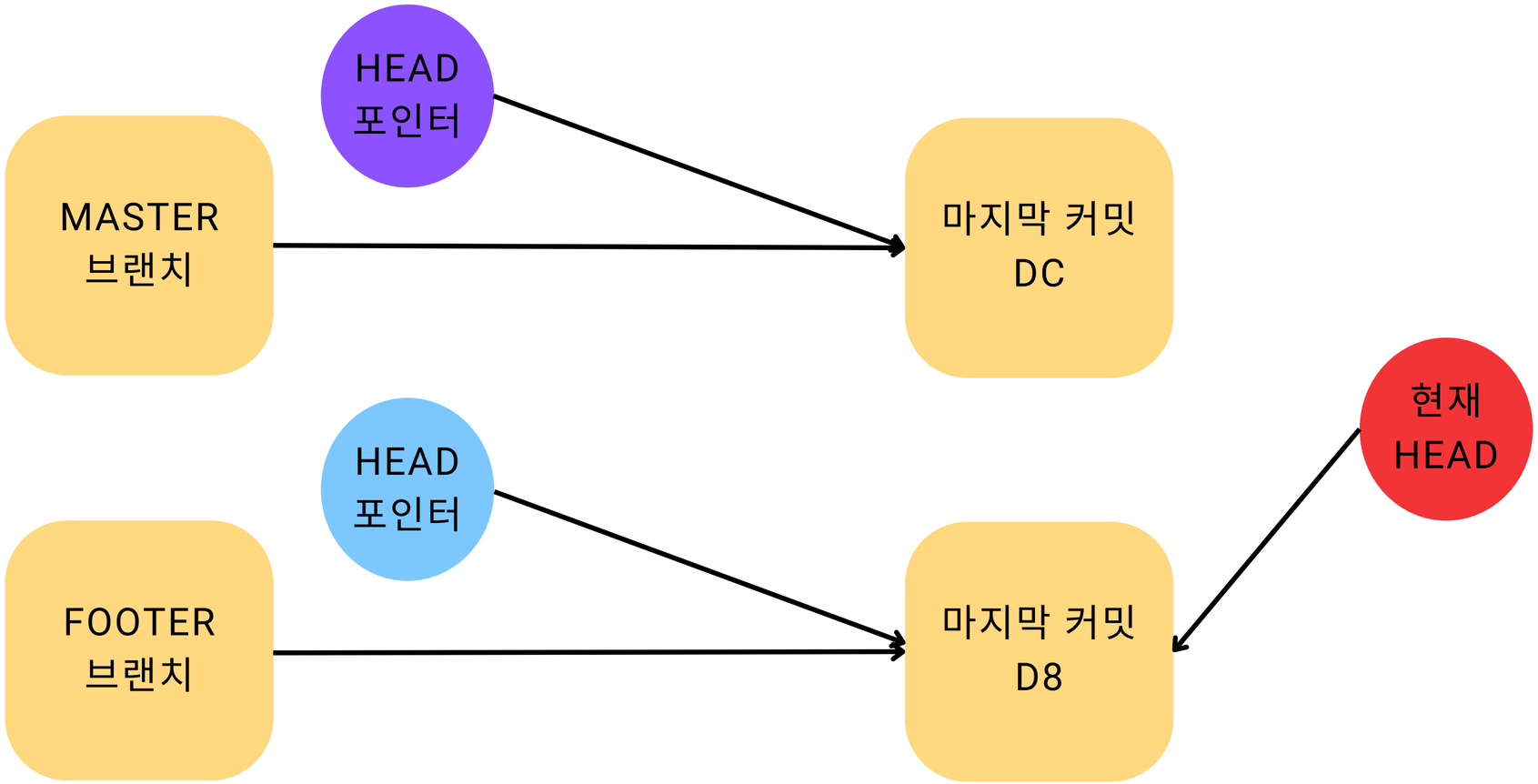
브랜치를 변경하려면 기존 브랜치의 워킹 디렉터리를 정리

HEAD 포인터

깃은 마지막 커밋 정보를 기반으로 새로운 커밋을 생성.
마지막 커밋은 새로운 커밋의 부모 커밋이다.
HEAD는 작업 중인 브랜치의 마지막 커밋 ID를 가르키는 참조 커밋이다.
HEAD 포인터를 부모 커밋으로 대체하여 사용함.
HEAD 포인터를 사용하여 빠르게 스냅샷을 생성 가능



HEAD 포인터


HEAD
포인터
MASTER
브랜치
마지막 커밋
DC
현재
HEAD
HEAD
포인터
FOOTER
브랜치
마지막 커밋
D8

깃의 HEAD 포인터

내부적으로 커밋을 생성하고 브랜치를 관리하는 데 유용, 명령어를 입력할 때도 기준점으로 사용

HEAD^ / HEAD~

HEAD^3 / HEAD~3

포인터 바로 이전 커밋

HEAD 기준으로 이전 3개 위치 지정

AHEAD, BHEAD

- 원격 저장소와 연동하여 깃을 관리한다면 브랜치마다 HEAD가 2개 존재.
- 원격저장소와 로컬저장소는 물리적으로 서로 다른 저장소
- , 서로 다른 커밋을 가르키는 HEAD 포인터를 가짐.
- AHEAD와 BHEAD는 서로 다른 저장소간 HEAD 포인터의 위치 차이를 의미.
- 브랜치를 여러개 운영한다면 다수의 AHEAD와 BHEAD가 생길 수 있다.

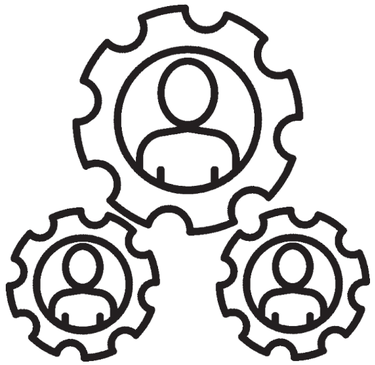
[AHEAD]

- 서버로 전송되지 않은 로컬 커밋이 있는 것.
- EX) 로컬 저장소에 새로운 커밋을 생성하고 새로운 커밋 정보를 서버로 전송하지 않은 상황
- 로컬 저장소의 HEAD 포인터를 기준으로 로컬 브랜치에 있는 커밋이 서버의 커밋 개수보다 많은 경우

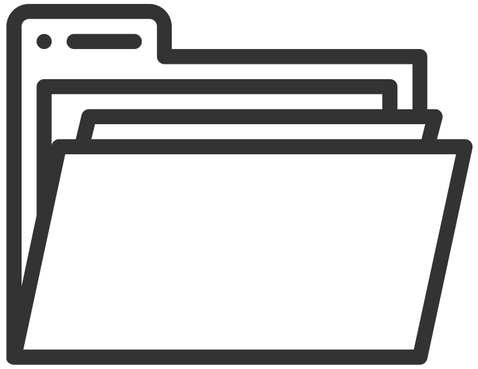
[BHEAD]

- 로컬 저장소로 내려받지 않은 커밋이 있는 것.
- EX) 누군가 서버에 새로운 커밋을 했을때, 아직 로컬 저장소에 서버의 새로운 커밋을 내려받지 않은 경우
- 코드 수정하여 원격 저장소의 커밋이 자신의 로컬 저장소보다 더 최신 상태인 것을 의미.

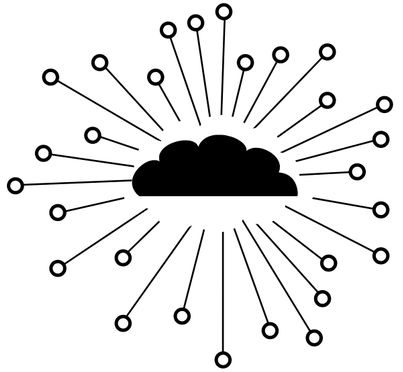
원격 브런치



깃은 다수 개발자와 협업으로 코드 유지 가능

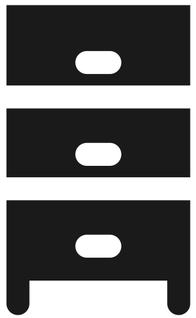


주요 개발 작업은 로컬 저장소에서 하지만 협업은 원격 저장소도 공유함

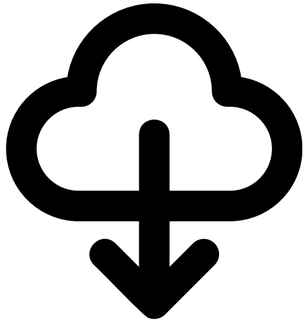


다수의 저장소를 만들어 연결할 수 있다

리모트 브랜치

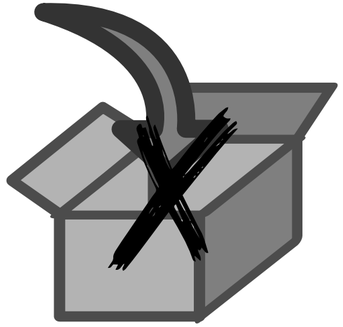


저장소는 각자의 고유한
브랜치를 생성하고 관리함

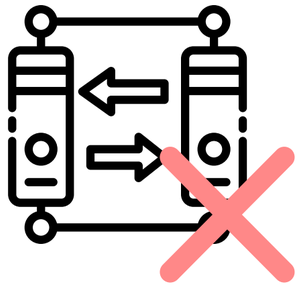


원격 저장소에 생성한 브랜치를
리모트 브랜치라고 함

로컬 저장소에 생성한 브랜치는
서버로 공유 가능



연결된 로컬 저장소에서 브랜치를 생성
한다 해서 자동으로 원격 저장소에도
브랜치가 생성되진 않음

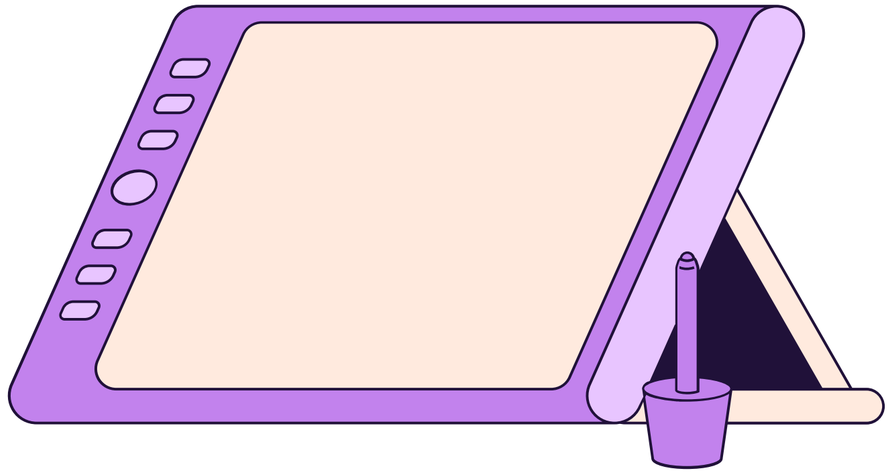


원격 저장소에 등록된 브랜치가 자동으
로 로컬 저장소를 만들지도 않음

로컬 저장소의 브랜치 이름은 보통
같지만 달라도 됨, 서로 다른 이름으
로 브랜치 연결 가능

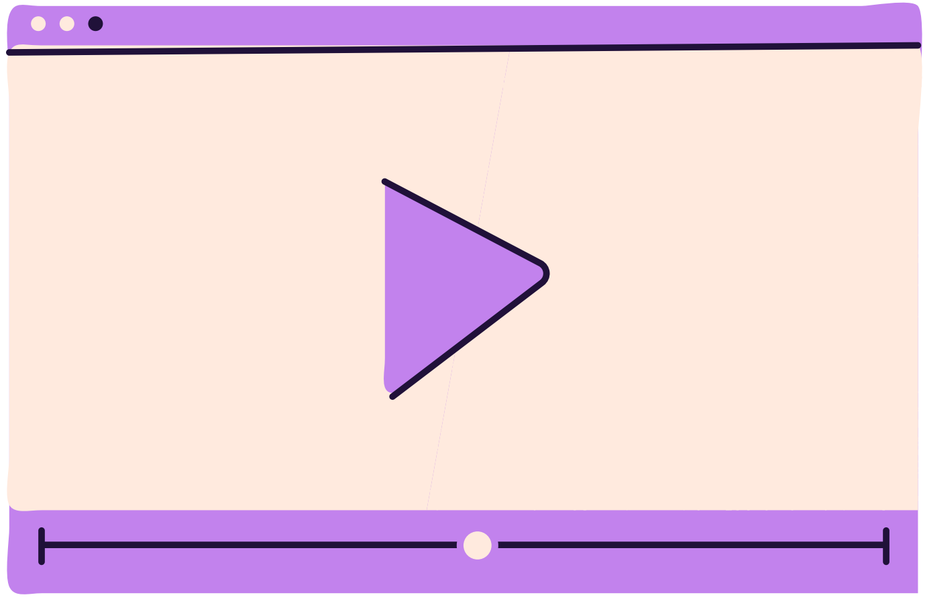
리모트 브랜치

https://github.com/perneean/OSS_6/blob/main/6%EC%9E%A5%20%EB%B8%8C%EB%9E%9C%EC%B9%98%209(%EB%B0%95%EC%83%81%ED%98%81).md



브랜치 전송

https://github.com/perneean/OSS_6



감사합니다