GUI 환경에서의 브랜치, 빨리감기 병합

세종대학교 이은상

커밋들의 줄기

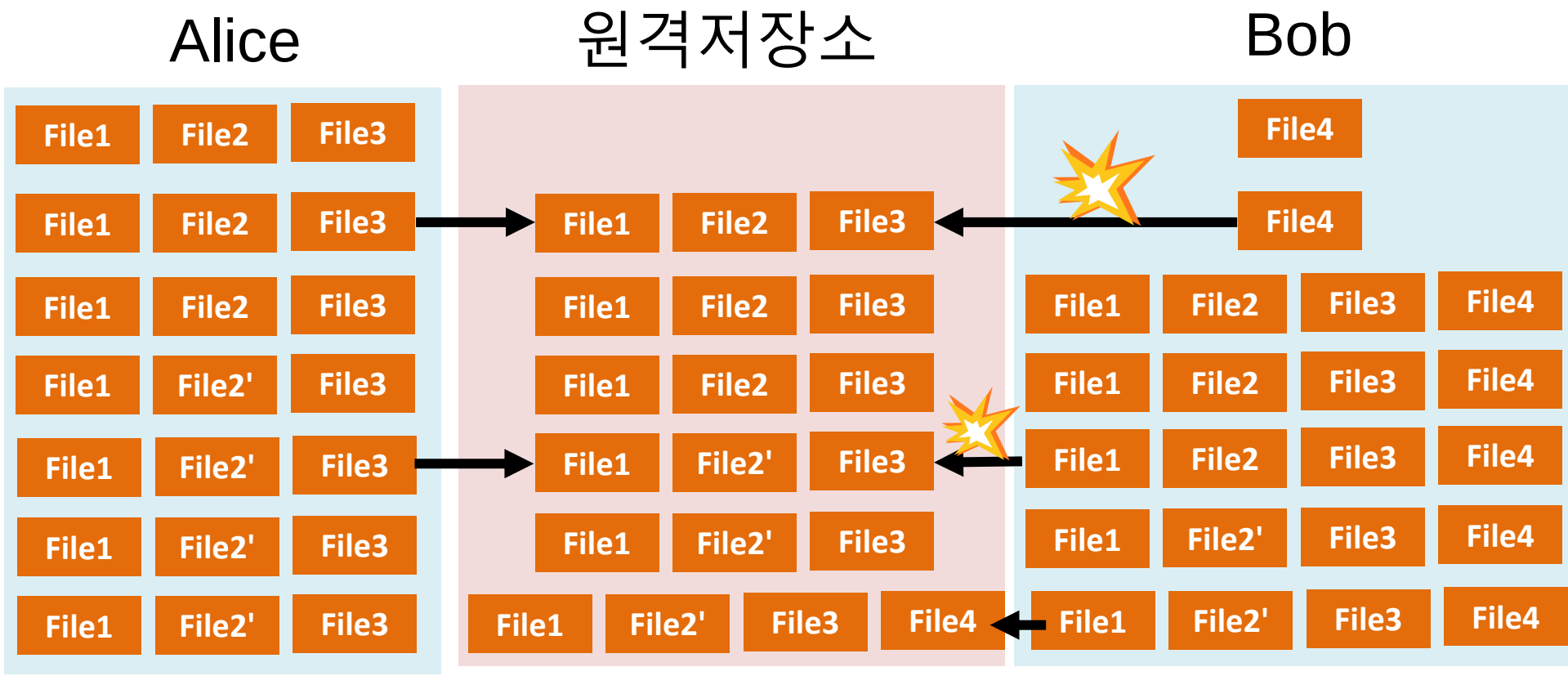
- 커밋은 줄줄이 기차처럼 연결되어있음
- 새로 만든 커밋은 기존 커밋 다음에 시간순으로 쌓이며 이전 커밋을 가리킴
- 커밋들은 하나의 줄기(브랜치)를 이룸

하나의 줄기에서의 협업 시 이슈

- 동시에 작업을 진행해야하는 경우 있음. 그런데 내가 작업하는 사이에 다른 사람이 수정한 부분이 있으면 합쳐야할 수 있음.
- Git은 내가 원할 때 작업해서 올리고, 원할 때 다른 사람이 새로 작업한 것과 합치는 것을 가능하게 함

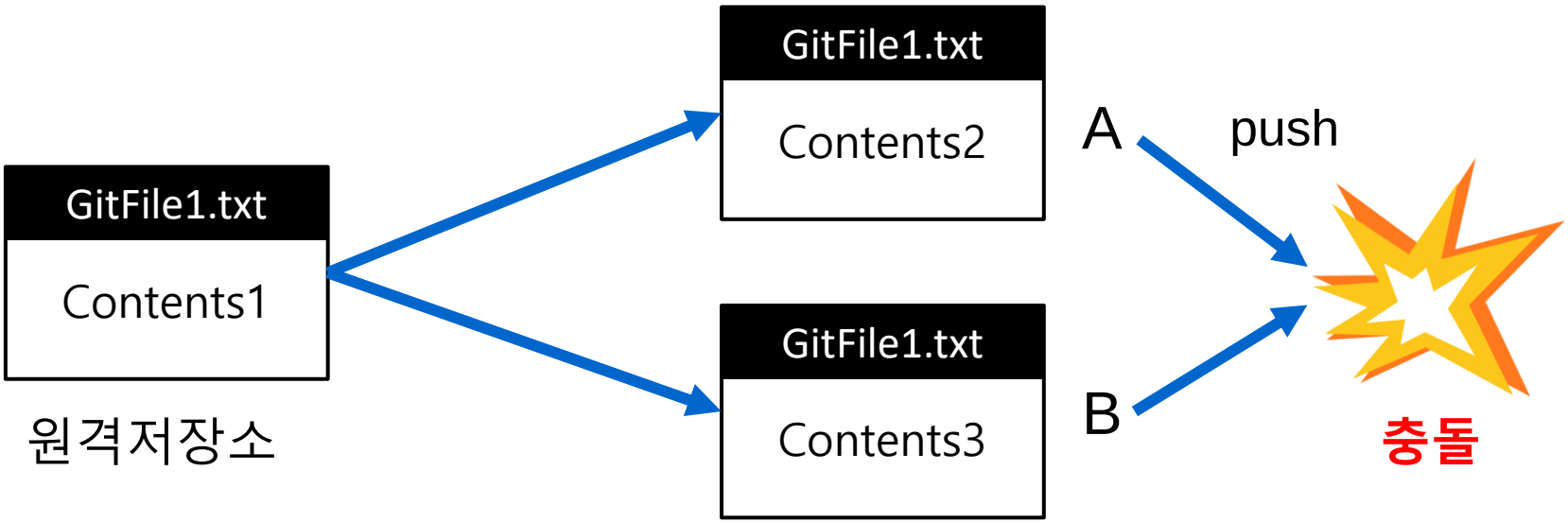
하나의 줄기에서의 협업 시 이슈

- 같은 원격저장소 공유 시 자주 충돌 발생



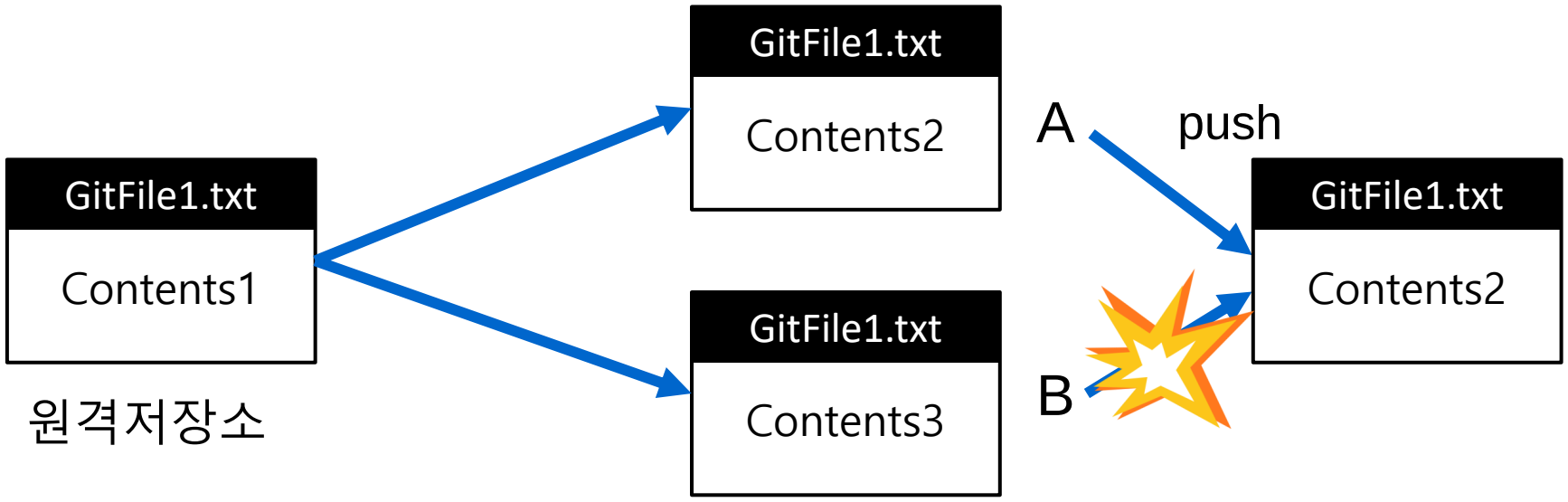
브랜치 개념

- 두 사람이 각자 수정한 내용을 원격저장소에 저장하는 경우? 두 수정 사항이 충돌함
- A가 작업한 Contents2로 결정할 경우 Contents3가 무시됨
- B가 작업한 Contents3로 결정할 경우 Contents2가 무시됨



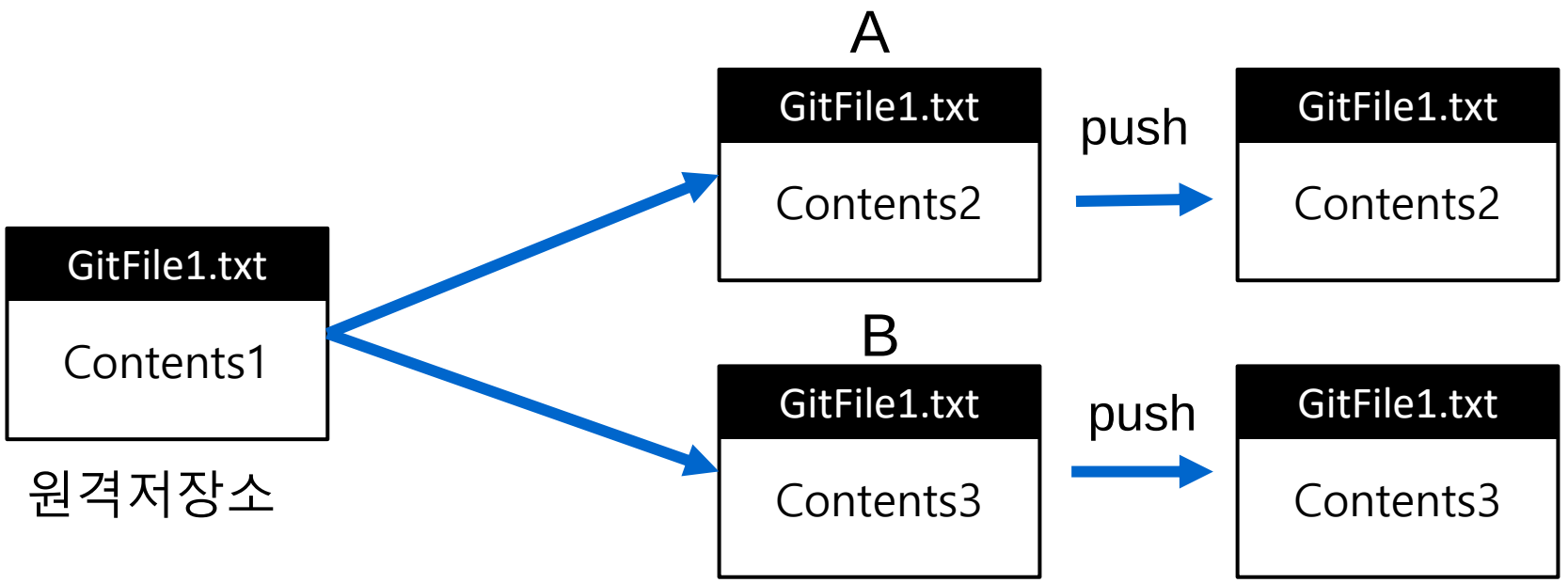
브랜치 개념

- 실제 git에서 브랜치를 안 만들고 동일한 브랜치에서 둘 다 충돌하는 커밋을 한다면? 원격저장소에 먼저 푸시한 커밋은 정상적으로 되고, 뒤늦게 푸시한 커밋은 "예전 코드가 아닌 최신 코드에 푸시하라 " 는 오류가 남



브랜치 개념

- 따라서 아예 두 갈래로 나눈 다음에 두 사람이 각자의 줄기에서 작업하면 충돌 걱정 없이 동시에 작업 가능함
- 이렇게 특정 기준에서 줄기를 나누어 작업할 수 있는 기능을 브랜치(branch)라고 함. 새로운 가지로 커밋을 만들기 위해 브랜치를 먼저 만들어야함

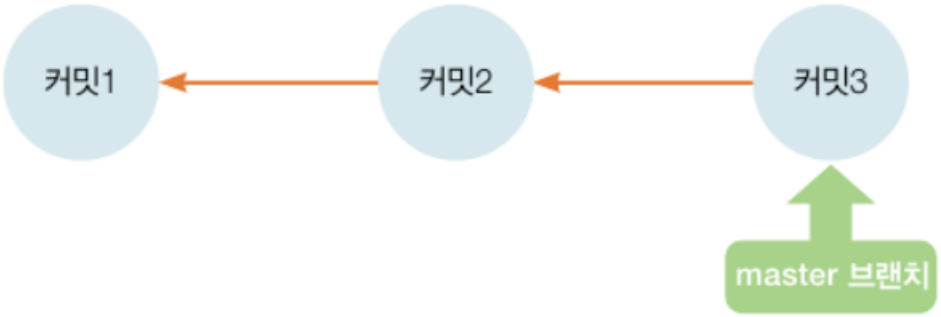


브랜치 정체

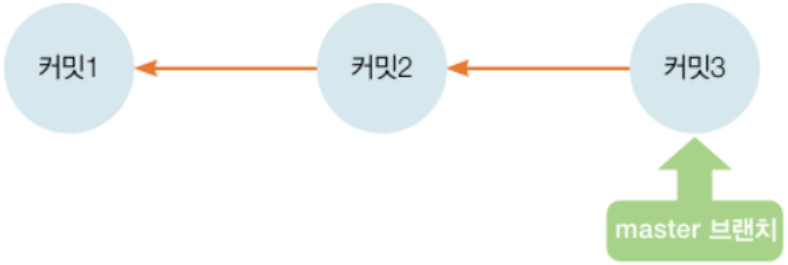
- 브랜치는 논리적으로는 어떤 커밋과 그 조상들을 묶어서 뜻하지만, 사실은 단순히 객체 하나를 가리킬 뿐임
- master 브랜치는 git이 제공하는 기본 브랜치임
- 첫 번째 커밋을 하면 자동으로 master라는 이름의 브랜치가 커밋을 가리킴
- 새로 커밋을 하면 master 브랜치가 두 번째 커밋을 가리킴

브랜치 정체

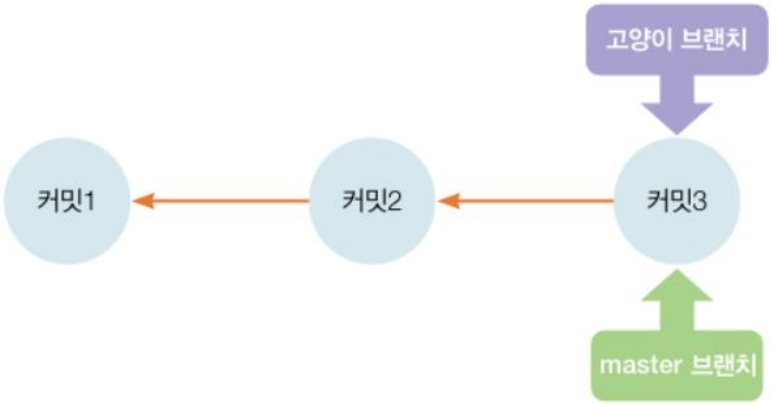
- 브랜치의 정체는 단순히 포인터
- 따라서, 브랜치가 커밋을 가리킨다고 표현함
- 새로 커밋을 하면 브랜치는 새로운 커밋을 가리키게 됨
- 예) 아래와 같이 새로 커밋을 한다고 할 때 커밋2를 가리키던 master 브랜치는 커밋3를 가리키게 됨



브랜치 생성 개념

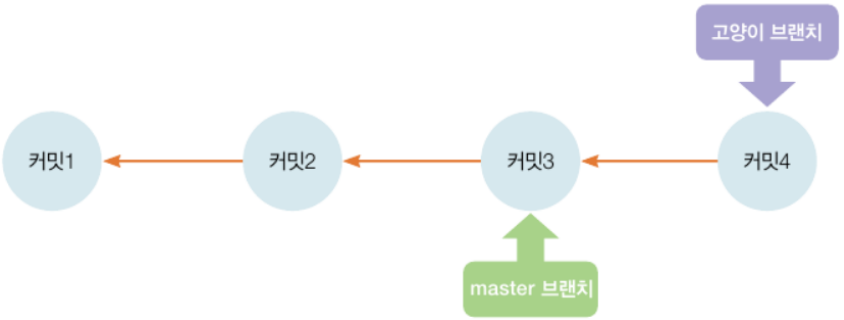


- 이 상태에서 [고양이] 브랜치를 새로 만든다고 함. 그러면 아래와 같이 [고양이] 브랜치도 master와 동일하게 커밋3를 가리키게 됨

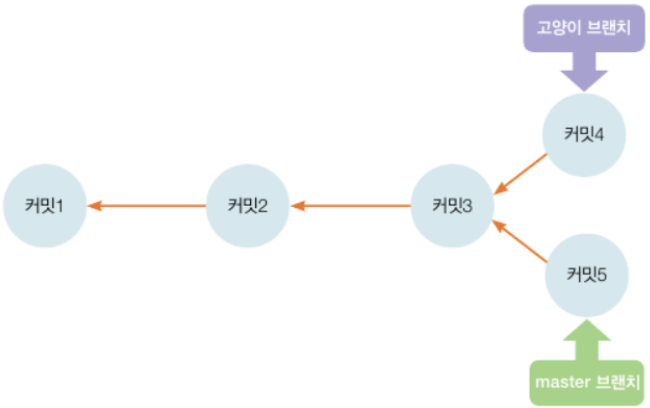


브랜치 갈라짐 개념

- 고양이 브랜치에 커밋을 하나 추가하면 고양이 브랜치는 master보다 하나 앞서게 됨

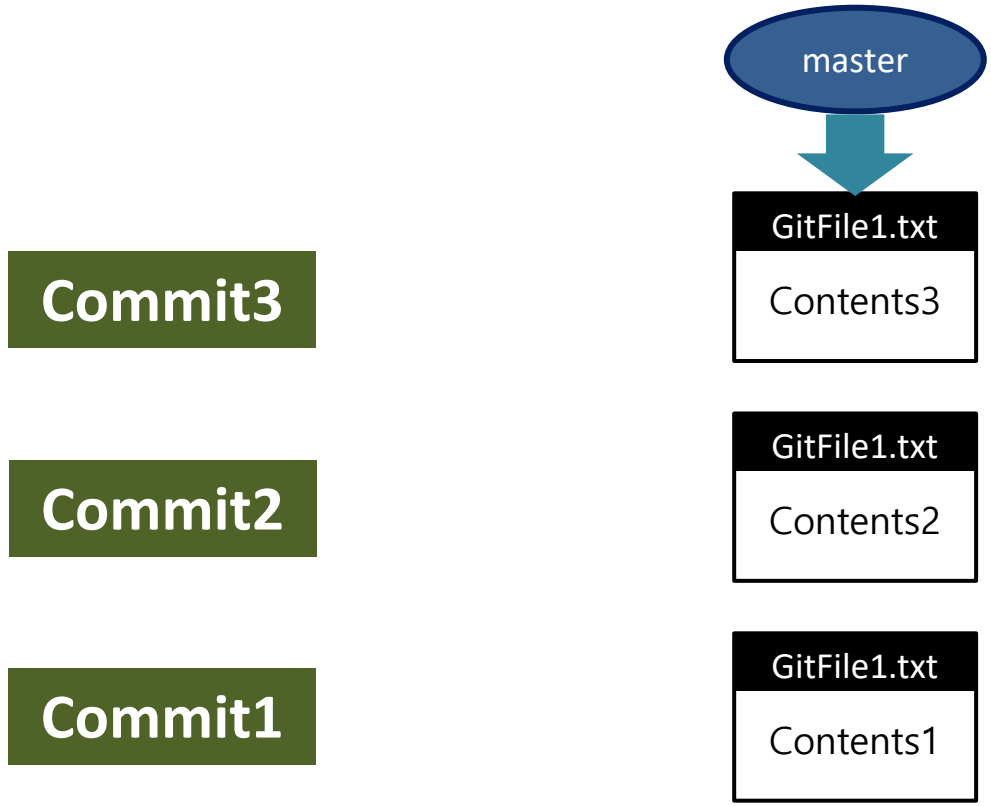


- 이 상태에서 master 브랜치로 이동해 커밋을 하나 더 하면 이제 두 브랜치가 눈에 띄게 갈라지게 됨. 커밋3을 기준(base)로 두 가지 버전이 생긴 것임



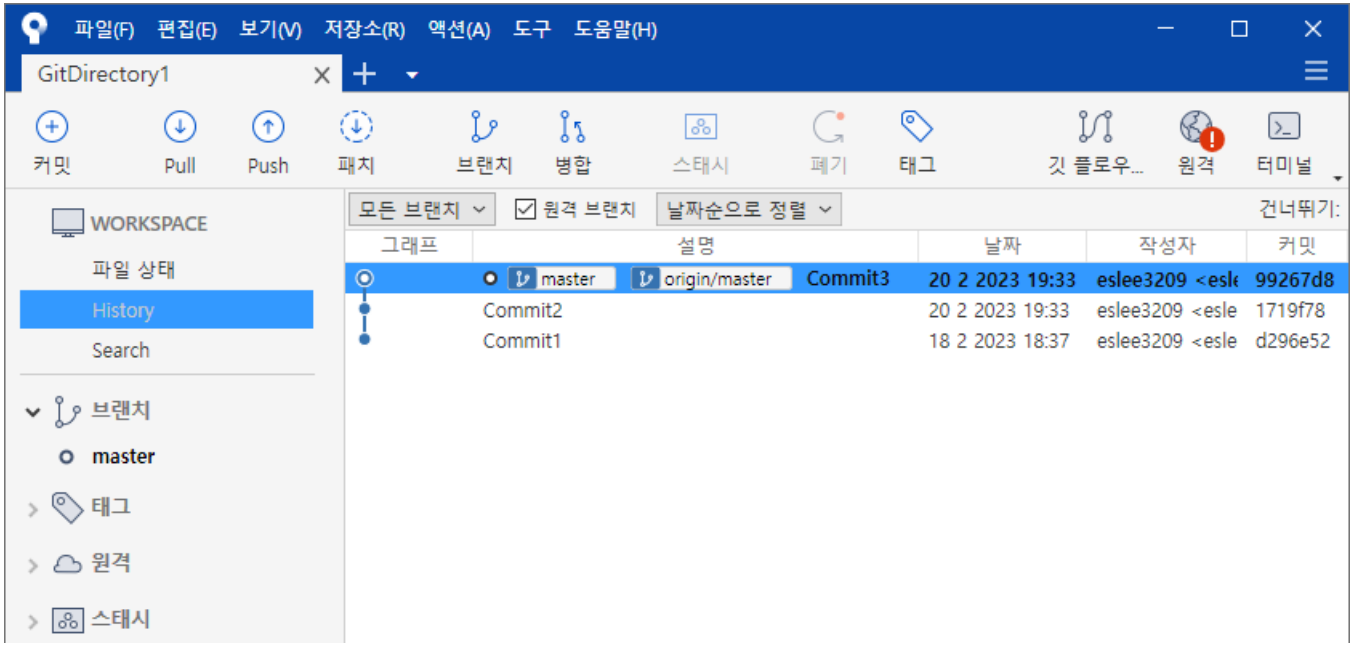
브랜치 실습

- 상황
  - 아래와 같은 상황에서 시작한다고 함



브랜치 실습

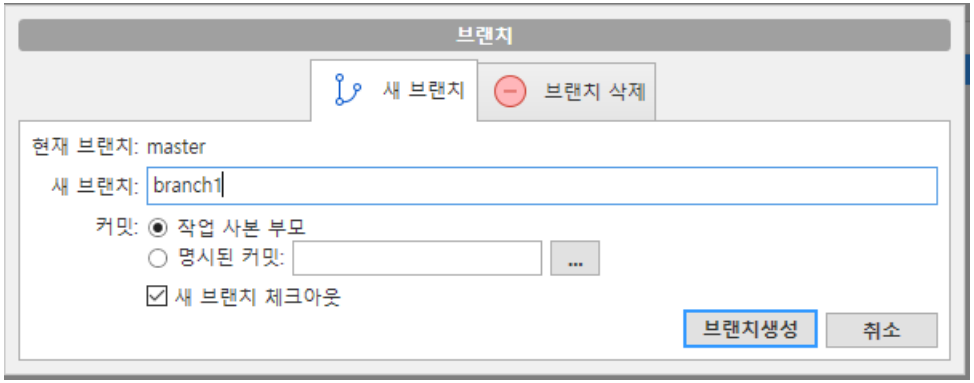
- 아래와 같이 master, origin/master가 모두 Commit3 커밋을 가리키고 있다고 가정
- 이제, [branch1]이라는 이름의 브랜치를 만들 것임



브랜치 실습

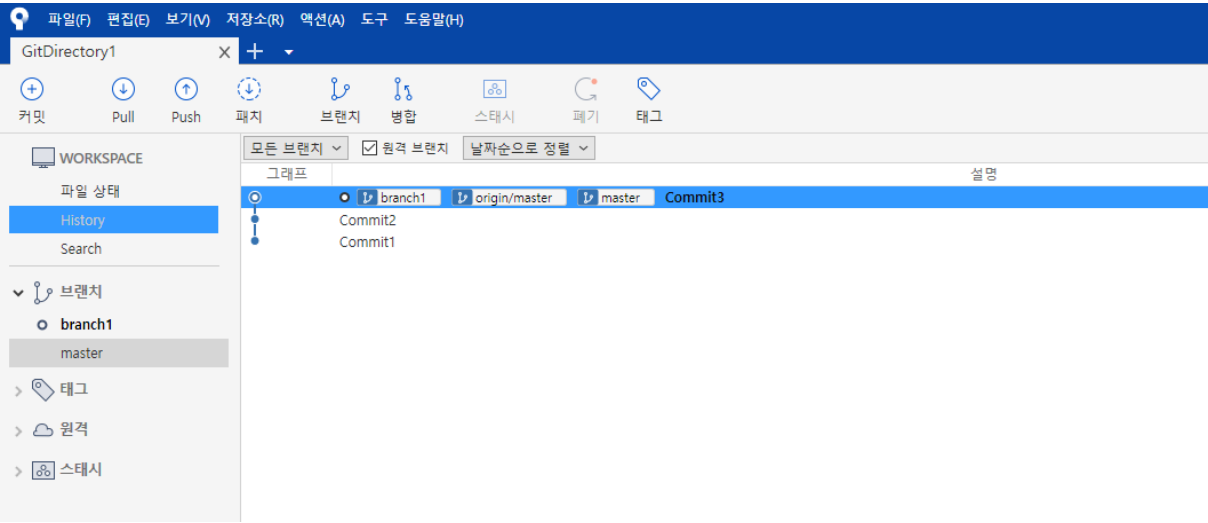
- 소스트리 상단에 [브랜치] 아이콘 클릭
- [새 브랜치] 이름에 "branch1"라고 적어줌. 현재 브랜치가 master임을 확인해야함
- [새 브랜치 체크아웃] 체크박스를 선택하고 [브랜치생성]을 클릭함

*** 체크아웃은 브랜치 이동 명령어임. [새 브랜치 체크아웃] 체크박스를 선택하면 브랜치를 만듦과 동시에 그 브랜치로 이동함



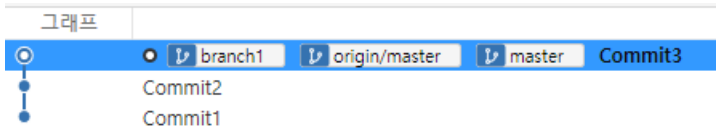
브랜치 실습

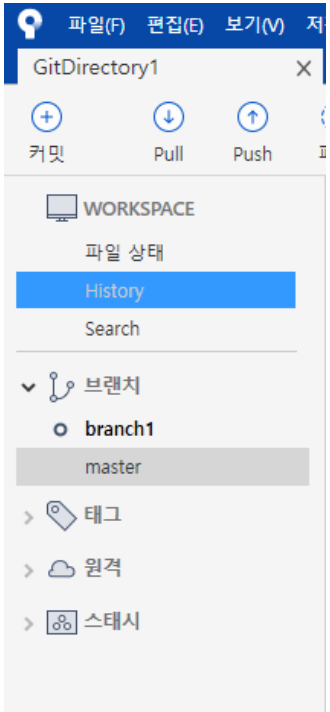
- 그러면 branch1이 생성된 것을 알 수 있음



브랜치 실습 - HEAD

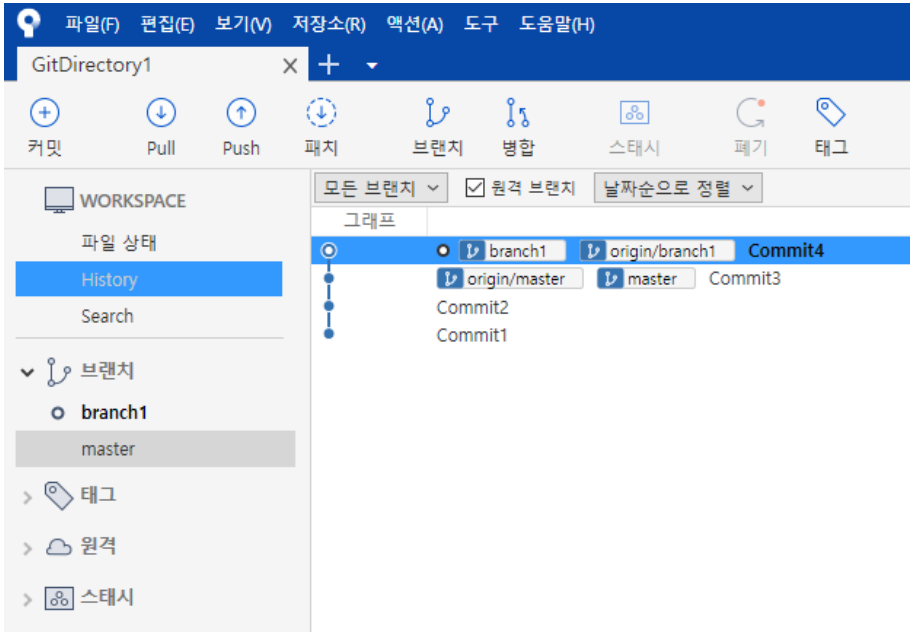
- HEAD
  - 현재 있는 브랜치의 가장 마지막 커밋을 뜻함
  - HEAD를 보면 현재 내가 어느 브랜치에서 작업하고 있는지 알 수 있음
  - 소스트리에서는 그림처럼 동그라미 표시가 된 것이 HEAD임
  - 이 그림들은 모두 HEAD가 branch1 브랜치를 가리키고 있음을 의미함





브랜치 실습

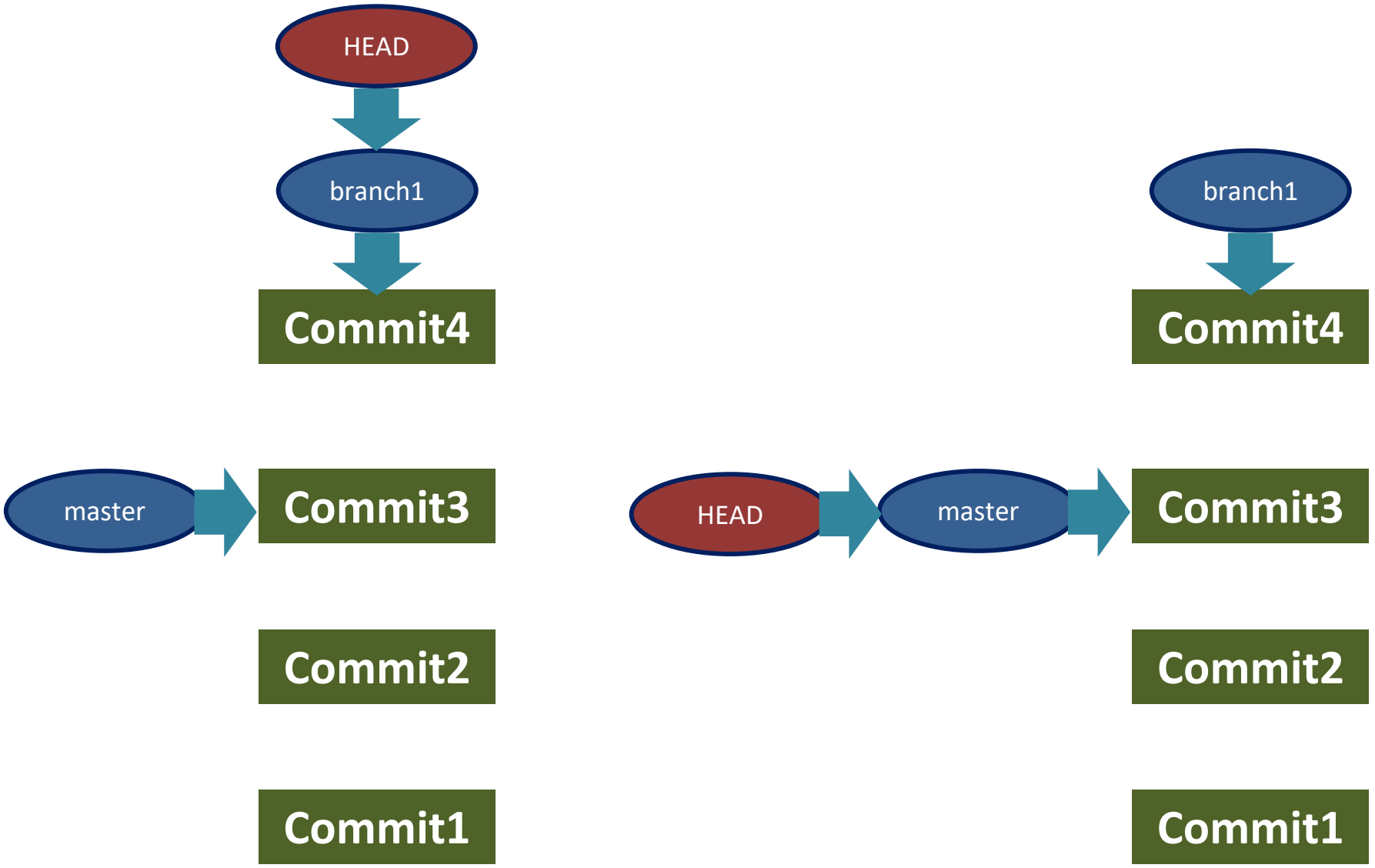
- 현재 브랜치(branch1)에서 파일을 새로 생성함. VSCode로 돌아와 새 파일을 생성하여 편집하면 됨
  - 예) GitFile2.txt 파일. 내용은 "Contents1"
- 그 후 커밋을 진행하고 푸시까지 동시에 진행함. [-에 바뀐 내용 즉시 푸시] 체크하면 push도 동시에 진행됨.
  - 예) 커밋 메시지: Commit4



브랜치 실습 - Checkout

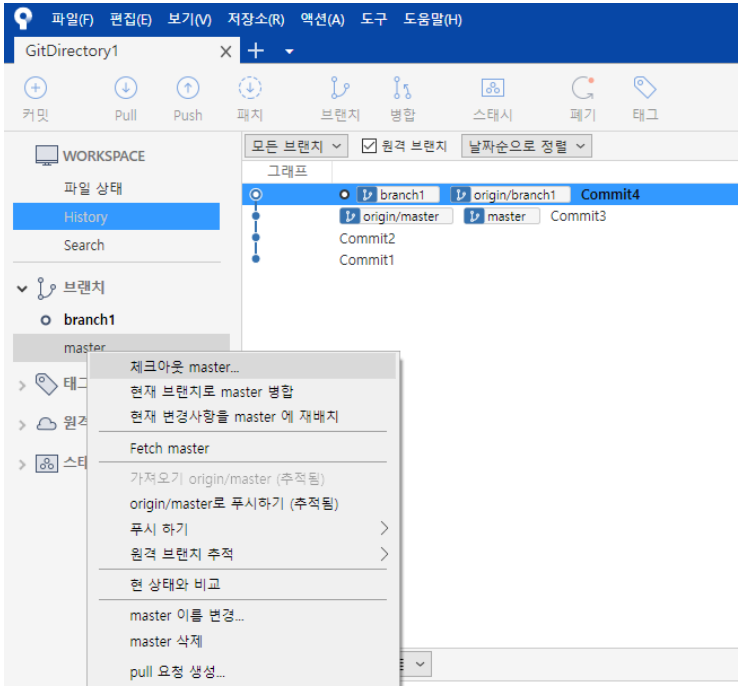
- 체크아웃(checkout)이란 다른 브랜치(혹은 커밋)로 이동하는 것임.
- HEAD 포인터가 이동하게 됨. 구체적으로 다른 브랜치로 이동하기도 하고 혹은 브랜치의 최신 커밋이 아닌 과거 커밋으로도 이동할 수 있음

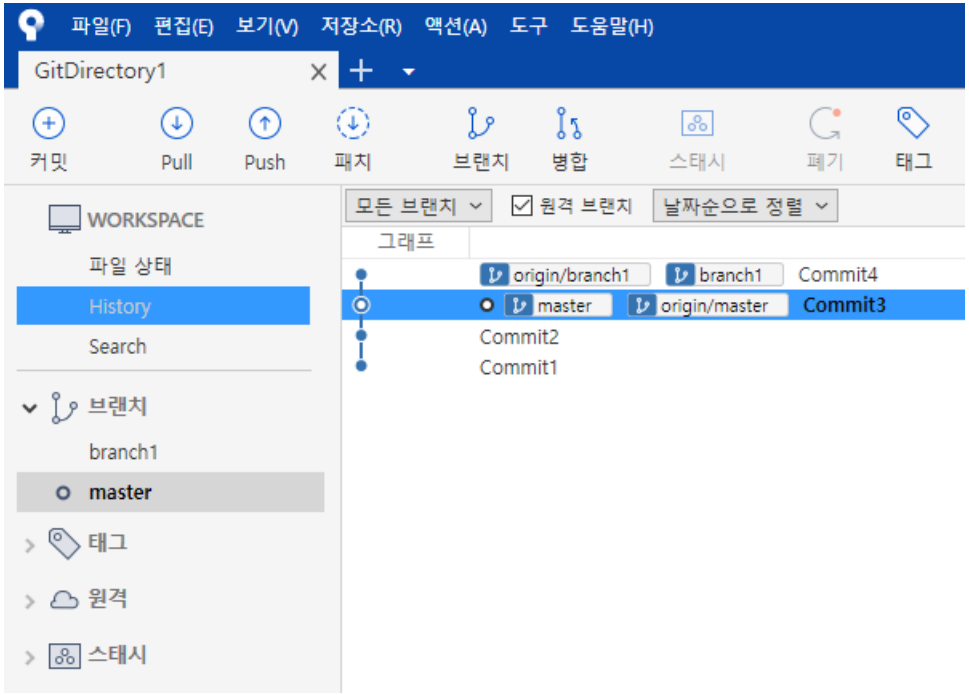
브랜치 실습


HEAD
branch1
Commit4
master
Commit3
Commit2
Commit1
branch1
Commit4
HEAD
master
Commit3
Commit2
Commit1

브랜치 실습 - 체크아웃 방법

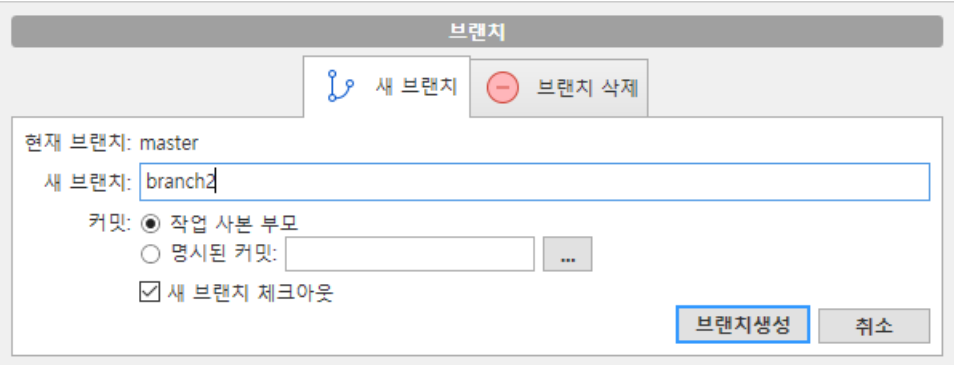
- 소스트리 좌측 탭에서 체크아웃하려는 브랜치에서 마우스 오른쪽 버튼을 클릭하고 [체크아웃 master]를 클릭함. 혹은 브랜치 이름을 더블클릭해주어도 됨

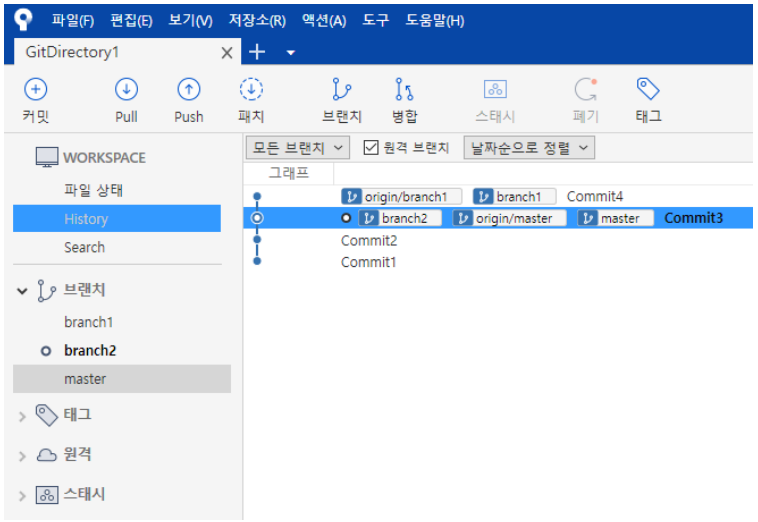




브랜치 실습 - 브랜치 갈라짐

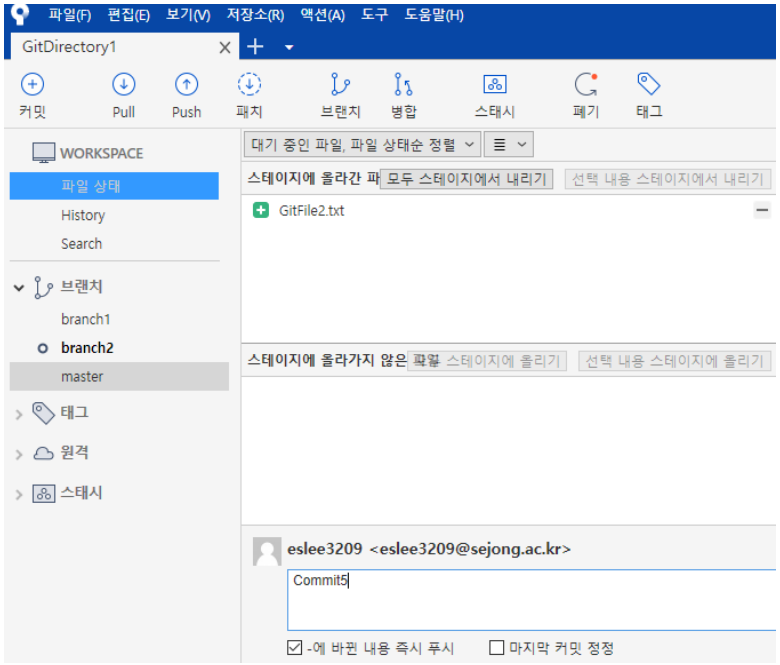
- 소스트리 상단의 [브랜치] 아이콘 눌러 새 브랜치 생성함 . 현재 브랜치가 [master]인지 다시 확인
- 새 브랜치 이름은 branch2로 입력하고 [브랜치 생성] 버튼을 클릭함





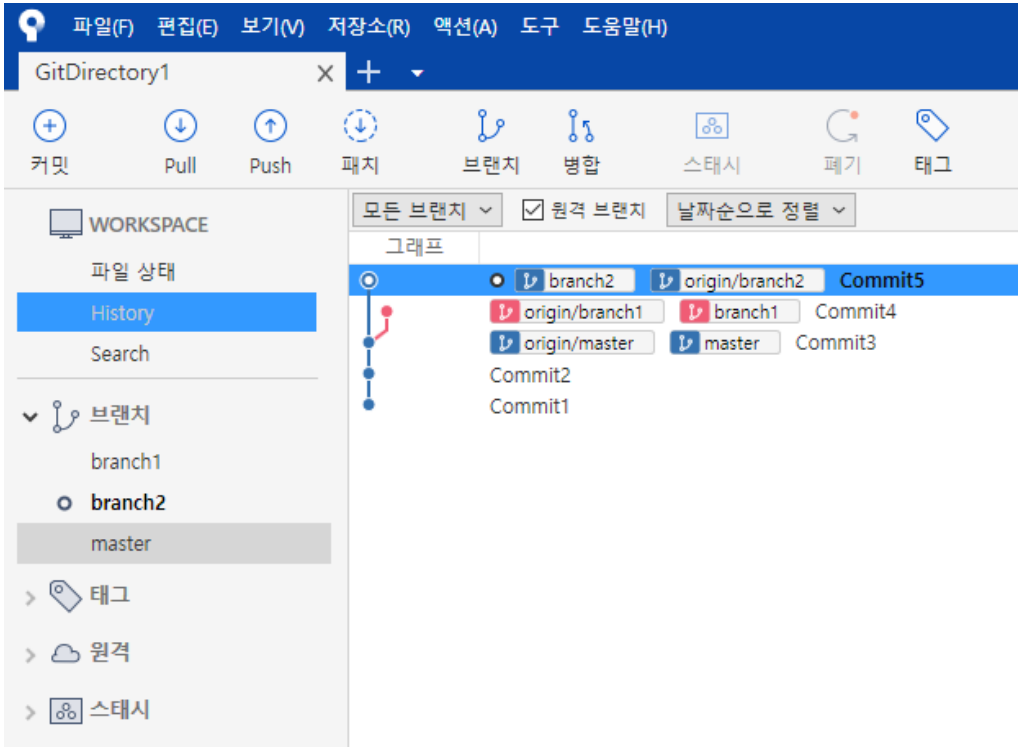
브랜치 실습 - 브랜치 갈라짐

- 이제 VSCode를 통해 GitFile2.txt를 생성하고 내용은 "Contents2"로 생성함
- 해당 내용을 커밋함. [-에 바뀐 내용 즉시 푸시] 체크하여 푸시까지 한번에 해줌
  - 예) 커밋 메시지: Commit5

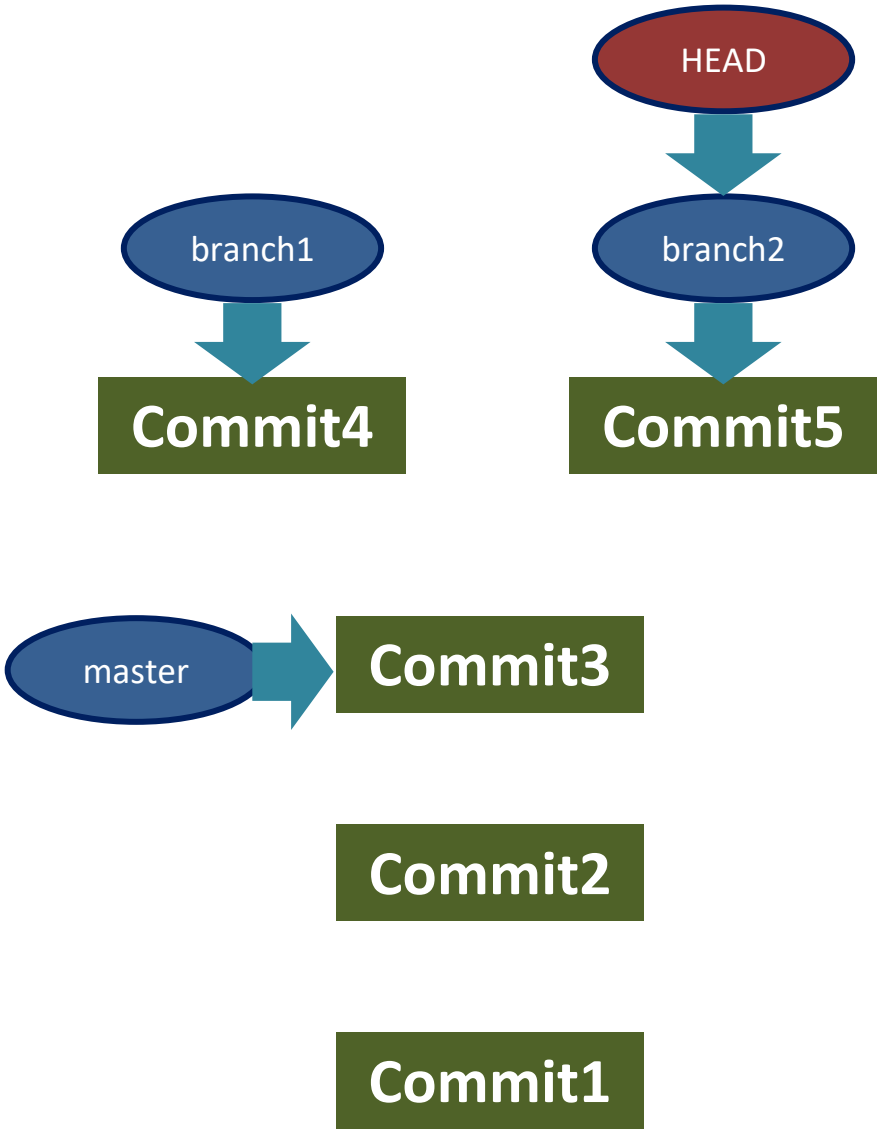


브랜치 실습 - 브랜치 갈라짐

- 아래와 같이 히스토리 그래프에서 두 가지가 뻗어나온 것을 확인할 수 있음

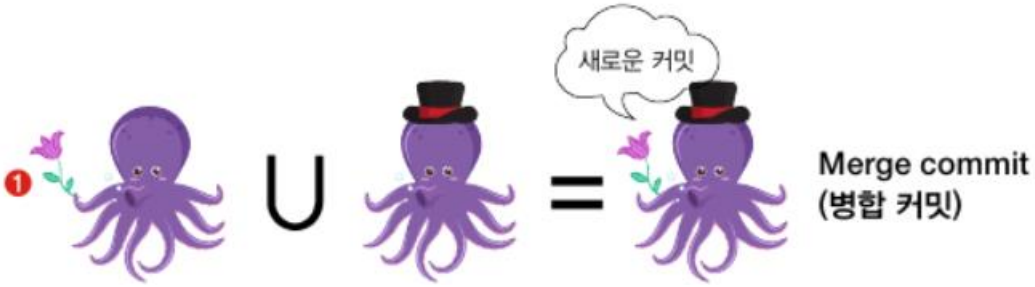


브랜치 실습 - 브랜치 갈라짐



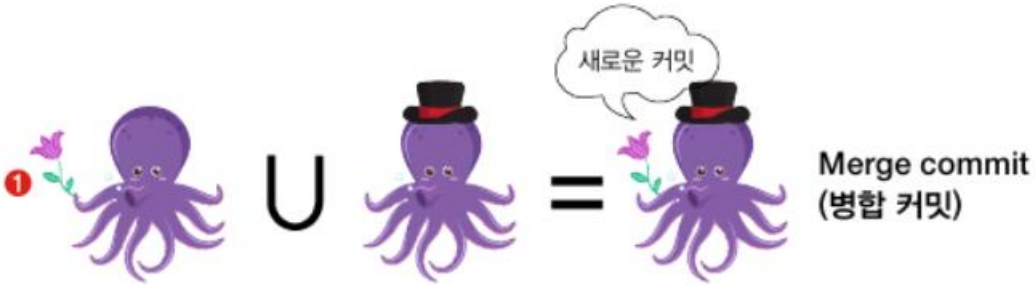
병합

- 병합은 간단히 말해 두 버전의 합집합을 구하는 것임. 다음 세 가지 경우가 있음
- 1) 일반 병합
  - 꽃을 든 문어, 모자를 쓴 문어를 합치면 꽃과 모자를 든 문어가 나옴



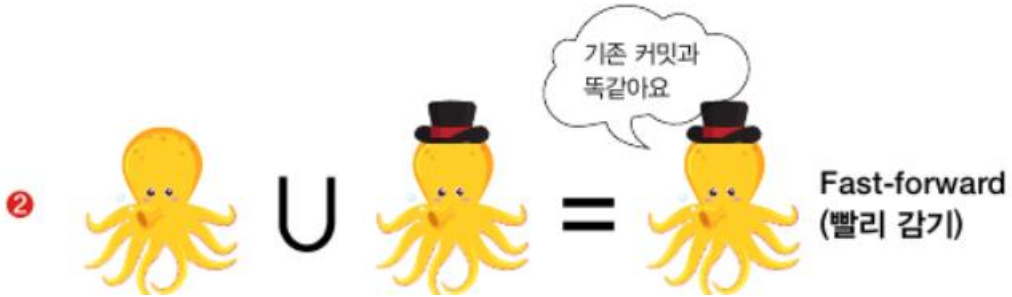
병합

- 1) 일반 병합
  - 두 사람이 서로 다른 파일을 수정한 경우
    - ex) Alice는 GitFile1.txt를, Bob은 GitFile2.txt를 수정한 경우
  - 두 사람이 같은 파일을 수정하되 수정한 부분이 다른 경우
    - ex) Alice는 GitFile1.txt의 1~200번째 줄까지 수정하고, Bob은 GitFile1.txt의 400~600번째 줄까지 수정



병합

- 2) 빨리 감기
  - 합친 결과물이 뒷 상태와 동일한 것을 의미함. 새로 상태를 만들어줄 필요 없이 뒷 상태로 바꾸면 되는데 이를 fast-forward라고 함

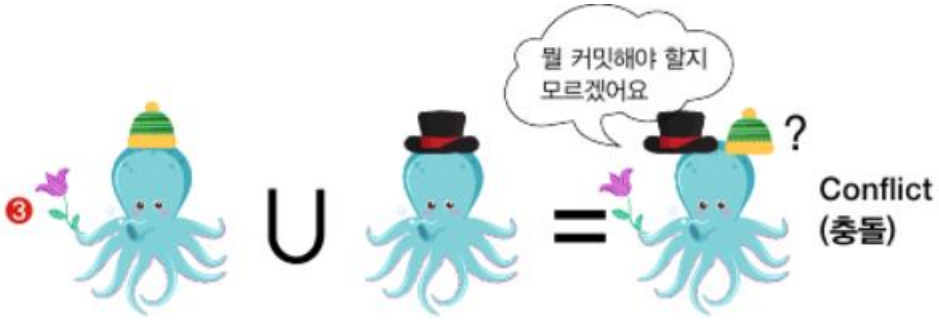


병합

- 2) 빨리 감기
  - Bob의 최신 버전이 단순히 Alice의 최신 버전에서 추가로 수정한 경우
  - 병합하면 단순히 Bob의 최신 버전이 됨

병합

- 3) 충돌
  - 모자가 다른 경우라 충돌이 생김

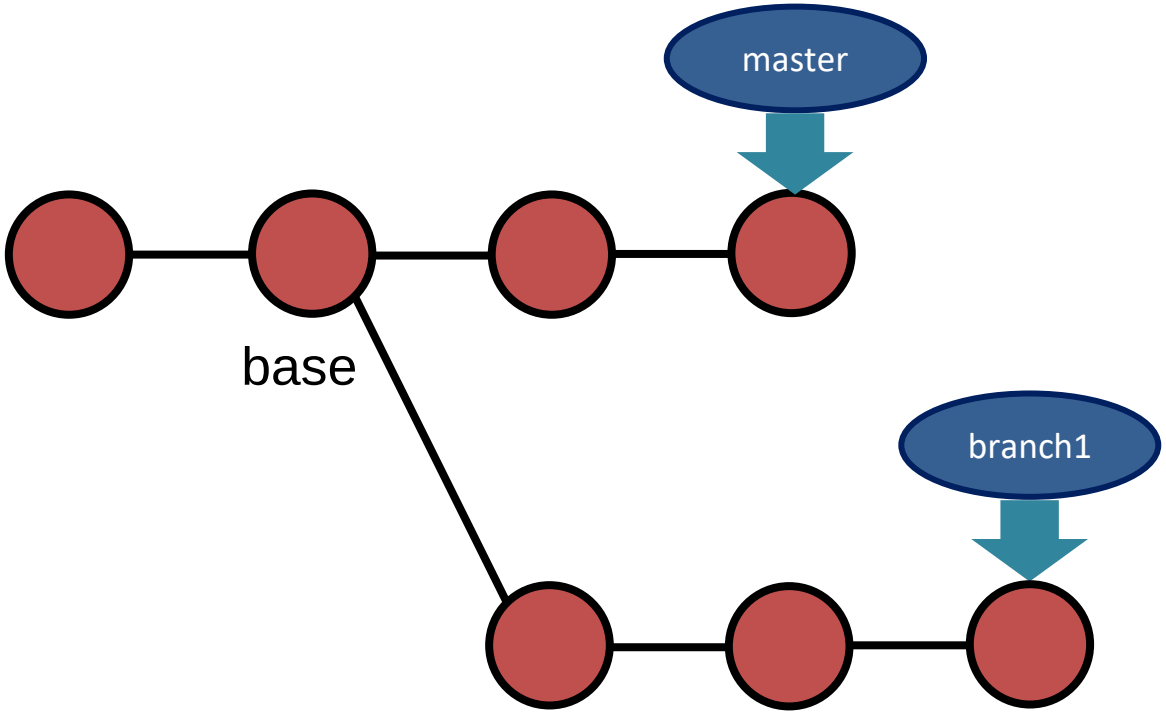


병합

- 3) 충돌
  - 두 사람이 같은 파일을 수정하고 수정한 부분이 겹칠 때
  - ex) Alice가 GitFile1.txt에서 100번째 줄을 수정했고, 동시에 Bob은 GitFile1.txt에서 100번째 줄을 다르게 수정

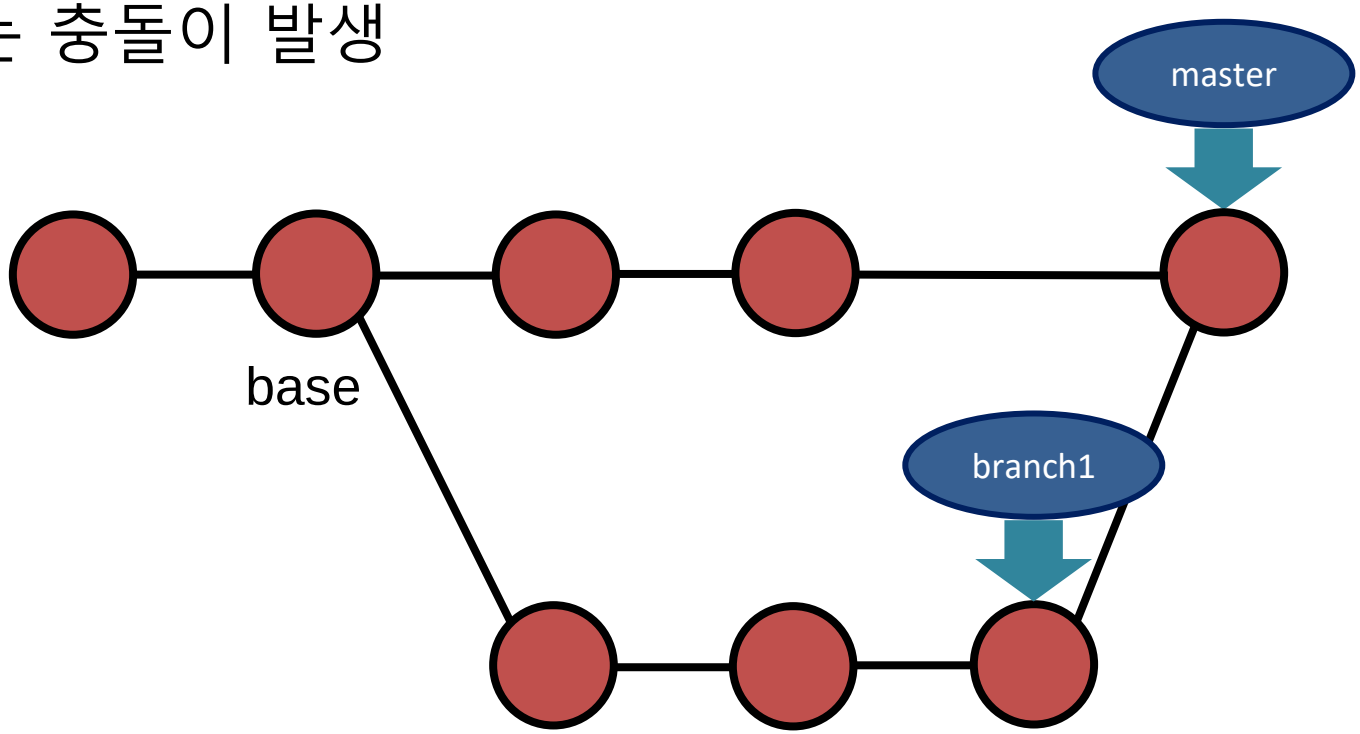
3-Way Merge

- 1), 3)의 경우는 3-way merge라 불림
- Base에서 커밋을 진행하여 분기해나간 상태에서 병합
- 두 브랜치 중 어느 것도 base에 위치하지 않는 상황
- 3-way라 불리는 이유는 병합 시 base와 각 브랜치 2개가 참조하는 커밋을 기준으로 병합을 진행하기 때문



3-Way Merge

- 3-way merge 진행 시 base와 각 브랜치가 참조하는 커밋을 고려하여 자동 병합 진행
- base를 기준으로 변경사항이 있는 파일들을 병합 커밋에 반영
- 만약 두 커밋 모두에서 변경사항이 발생한 파일에 대해서는 충돌이 발생

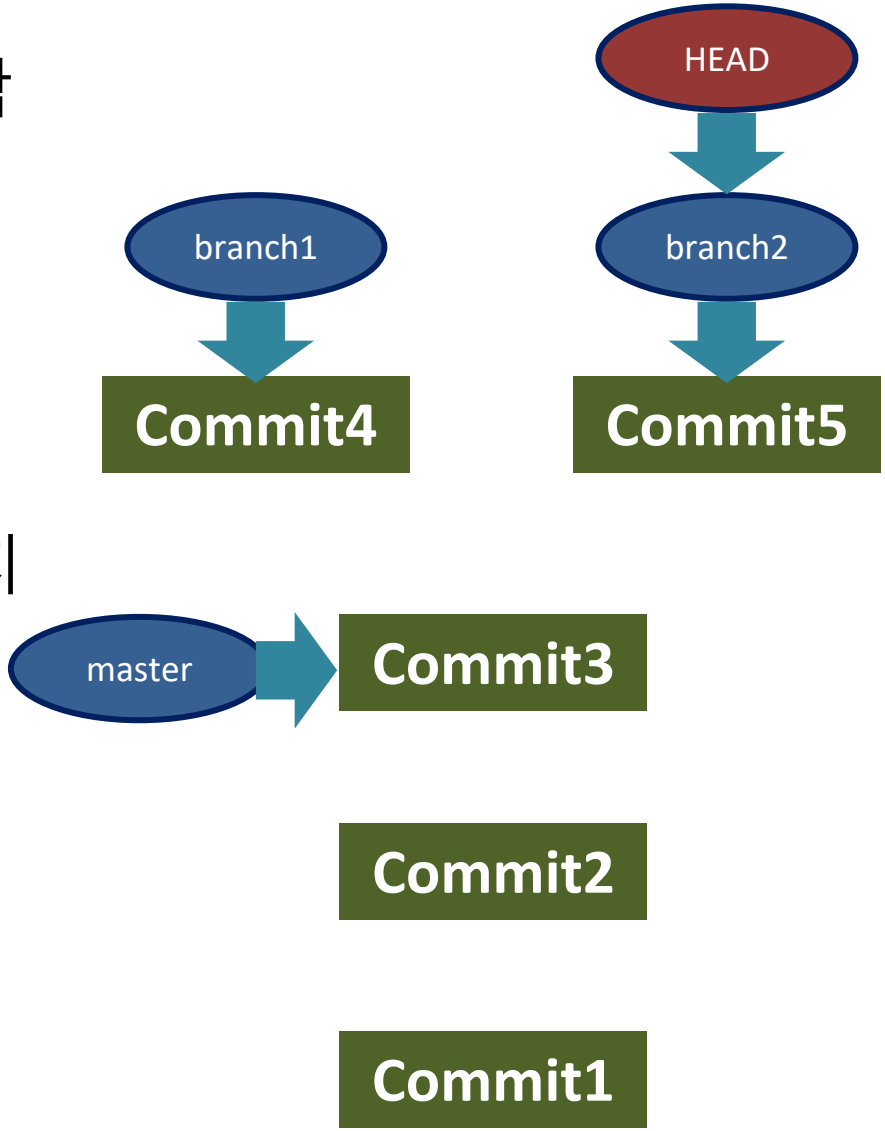


실전에서의 충돌 예방

- 기본적으로 프로젝트의 큰 흐름인 master 브랜치에 수시로 병합하는 것은 바람직하지 못함
  - 자신의 브랜치에서 작업한 후 충분히 검증이 되면 master 브랜치에 병합
- 병합 없이 오랜기간 내 브랜치에서 개발하면? 나중에 병합 시 충돌 수정이 너무 많아지므로 비효율적
- 따라서, master 브랜치의 변화를 지속적으로 자신의 브랜치로 가져와 병합해야함

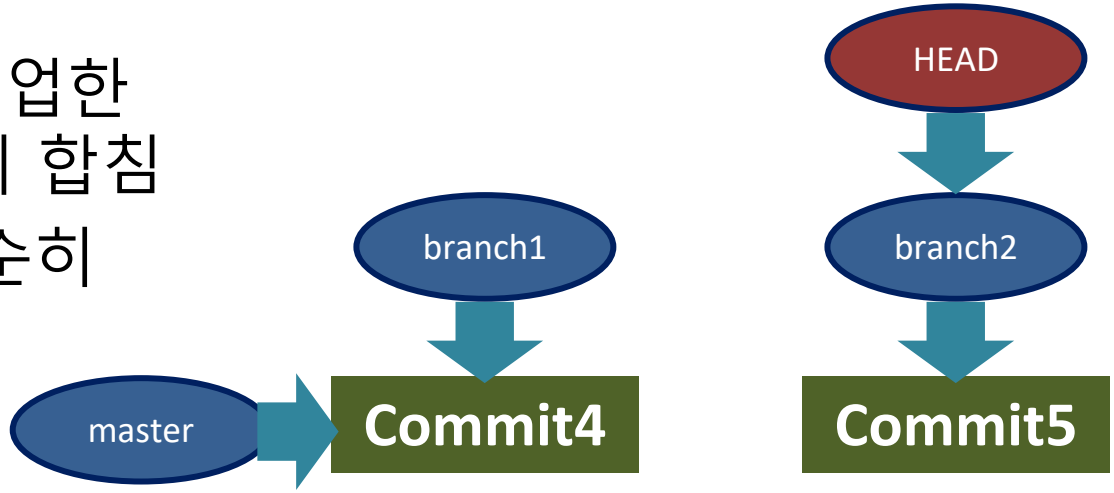
병합 - 두 브랜치를 합치는 과정

- Git에서 브랜치와 브랜치를 합치는 명령어는 머지(merge)인데 우리말로는 병합이라고 부름
- 두 명이 각자 만든 브랜치(branch1, branch2)에서 작업한 것을 master 브랜치에 합치는 경우 생각



병합

- 먼저, branch1에서 작업한 것을 master 브랜치에 합침
- 두 상태를 합치면 단순히 Commit4가 됨(fast-forward)
- branch1은 더이상 필요없으니 삭제



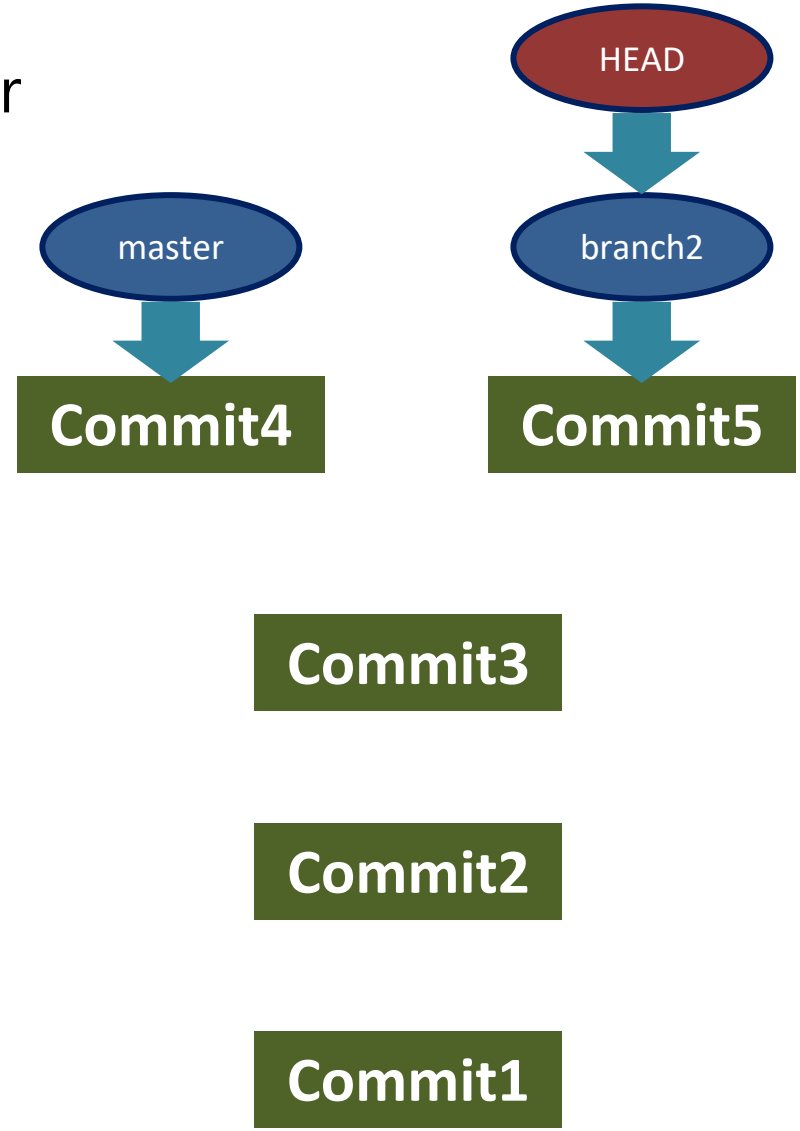
Commit3

Commit2

Commit1

병합

- 이번엔 branch2의 내용을 master 브랜치에 합치려고 함
- 이 경우에는 Commit4와 Commit5가 Commit3 중심으로 상태가 바뀌었음.
- 따라서 빨리감기 병합을 할 수는 없고 3-way 병합을 해야함



병합

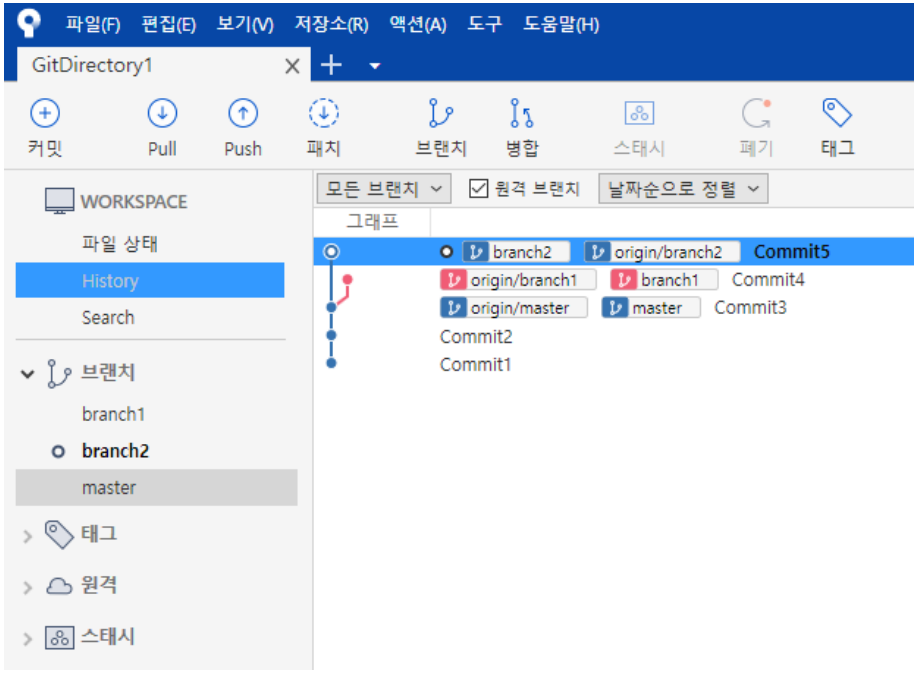
- 커밋4와 커밋5를 합친 병합 커밋을 만든다. 이렇게 두 브랜치를 합치면서 새로 병합된 커밋을 올릴 브랜치로는 [master] 및 [branch2] 중 선택할 수 있음
- master 브랜치에 올린다고 함

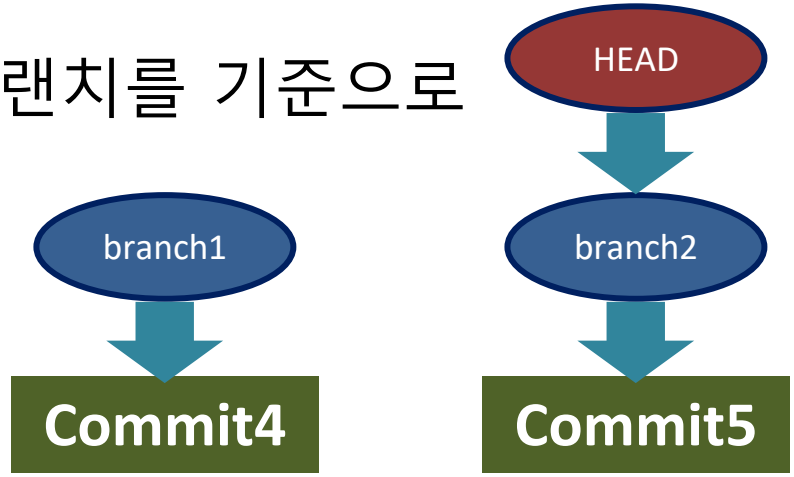
병합 시 기준 브랜치

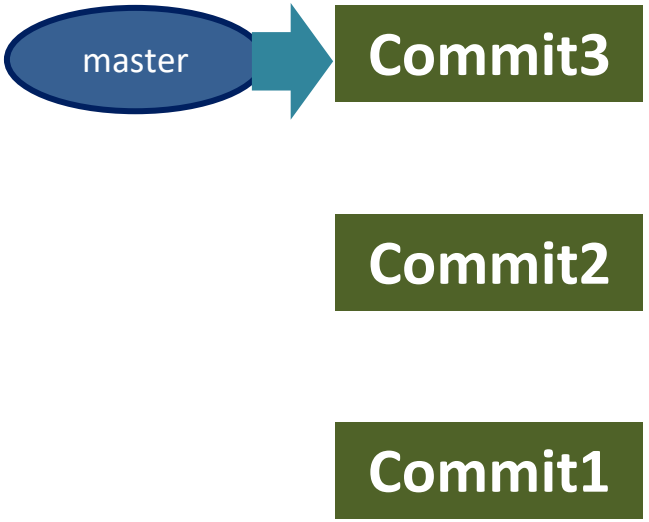
- A 브랜치와 B 브랜치 두 개를 합칠 때 합친 결과를 A 브랜치에 올릴지 B 브랜치에 올릴지 정해야함
- A 브랜치를 베이스로 병합
  - A브랜치에만 반영되고 B 브랜치에는 반영 안 됨
- B 브랜치를 베이스로 병합
  - B 브랜치에만 반영되고 A 브랜치에는 반영 안 됨

빨리감기 병합 실습

- 아래와 같은 상황에서 master 브랜치를 기준으로 branch1 브랜치를 합친다고 함





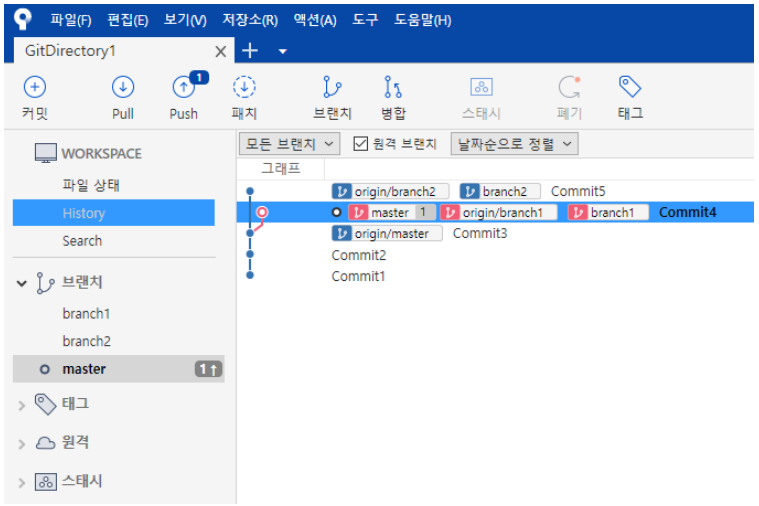


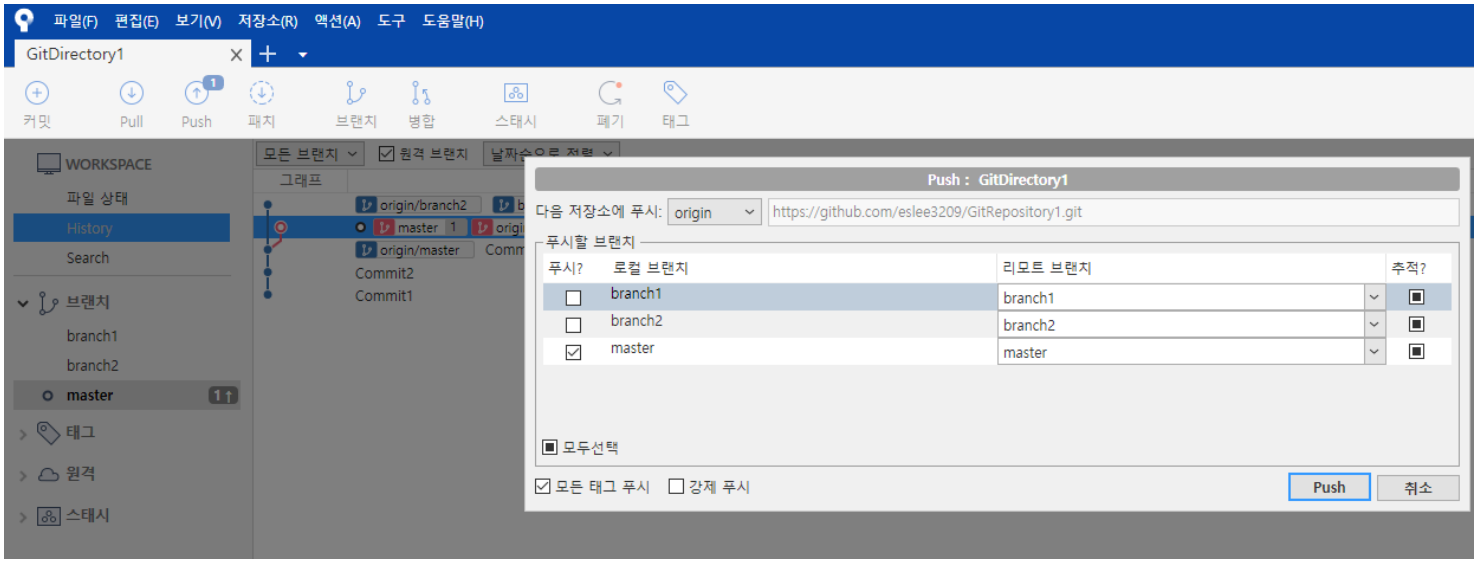
빨리감기 병합 실습

- 먼저 master 브랜치로 체크아웃함
- 병합하기를 원하는 커밋(branch1의 최신커밋)에서 마우스 오른쪽 버튼을 눌러 [병합] 메뉴를 선택함
- [fast-forward가 가능해도 새 커밋으로 생성]은 체크하지 않음

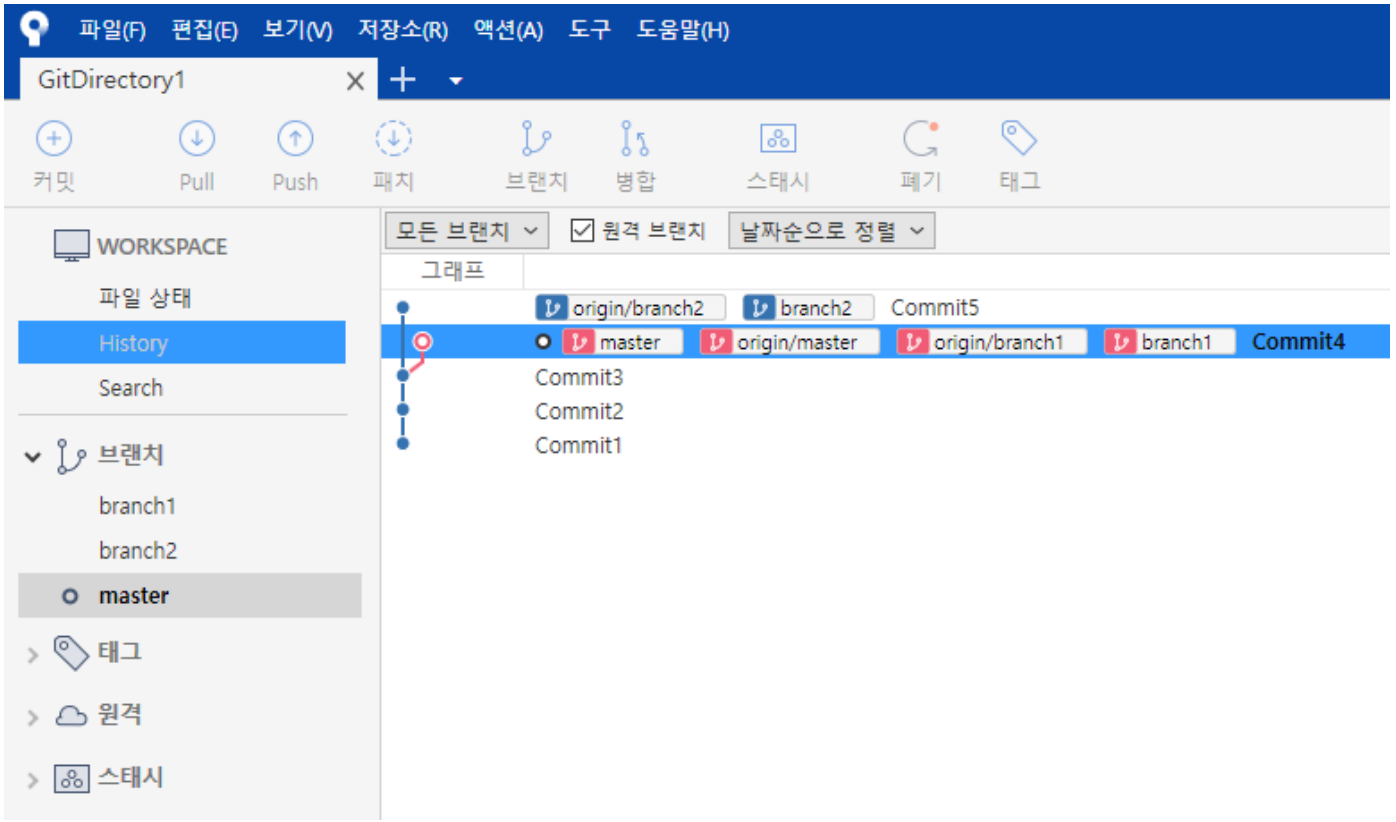
빨리감기 병합 실습

- 두 브랜치가 합쳐짐. master 브랜치가 branch1 브랜치와 같은 커밋을 가리키게 됨. 빨리감기 방식임
- Push 버튼을 눌러 푸시까지 진행하면 원격저장소에도 적용이 됨





빨리감기 병합 실습


파일(F) 편집(E) 보기(V) 저장소(R) 액션(A) 도구 도움말(H)
GitDirectory1
커밋
Pull
Push
패치
브랜치
병합
스태시
폐기
태그
WORKSPACE
파일 상태
History
Search
브랜치
branch1
branch2
master
태그
원격
스태시
모든 브랜치
원격 브랜치
날짜순으로 정렬
그래프
origin/branch2
branch2
Commit5
master
origin/master
origin/branch1
branch1
Commit4
Commit3
Commit2
Commit1

브랜치 삭제

- 소스트리 상단 [브랜치] 아이콘을 클릭
- [브랜치 삭제]를 클릭
- 로컬/원격저장소 모두 브랜치 삭제 가능함. 로컬은 branch1, 원격저장소는 origin/branch1 과 같은 이름