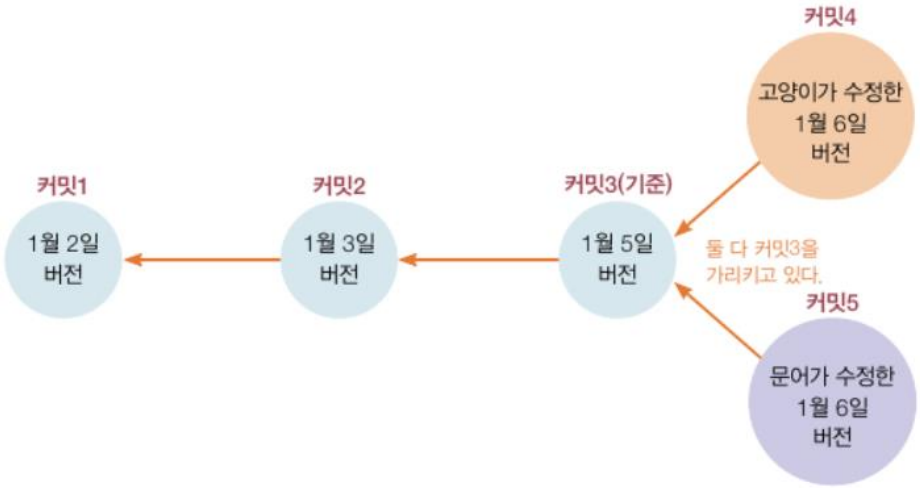
CLI 환경에서의 버전관리2

브랜치

- 아래와 같이 특정 기준에서 줄기를 나누어 작업할 수 있는 기능이 브랜치
- 브랜치의 정체는 커밋을 가리키는 포인터



브랜치 용도

- 새로운 기능 추가
  - 가장 대표적으로 브랜치를 사용하는 경우
  - master 브랜치에서 정상적으로 동작하는 안정적인 버전의 프로젝트가 저장되며, 새 기능을 추가할 때는 master 브랜치의 최신 커밋으로부터 브랜치를 생성해서 개발함
  - 개발, 코드 리뷰, 테스트까지 모두 완료해서 이상이 없으면 master 브랜치로 병합

브랜치 용도

- 버그 수정
  - 오류와 버그는 항상 발생함
  - 버그가 발생하면 master 브랜치로부터 새 브랜치를 생성해서 작업함
  - 이 때, 브랜치 이름은 hotfix 혹은 bugfix 등의 이름을 사용
  - 버그 수정이 끝나면 당연히 master 브랜치로 병합
  - 이 경우 이후에 새로 개발한 내용을 다시 master 브랜치에 병합할 때 버그 수정으로 인해 충돌이 생기므로 주의해야함

브랜치 용도

- 병합과 리베이스 테스트
  - 병합과 리베이스는 까다로운 일
  - 이 때, 임시 브랜치를 만들어서 병합과 리베이스 테스트를 해보면 편리함
  - 잘못되었을 경우 그냥 브랜치를 삭제하면 됨

브랜치 용도

- 이전 코드 개선
  - 이미 기능을 구현완료되었는데 코드가 마음에 들지 않아 개선하고 싶은 경우
  - 많은 사람이 기존 코드를 주석 처리하고, 그 아래 새로운 개선 코드를 작성하는데, 이 방법보다 브랜치 사용이 추천됨
  - 다른 브랜치에는 코드가 여전히 남아있으므로 걱정할 필요 없음

브랜치 용도

- 특정 커밋으로 돌아가고 싶을 때
  - 이전 커밋으로 돌아갈 때 hard reset 혹은 revert를 사용할 수 있음
  - 새 브랜치를 만들어 작업하고, 이후 리베이스나 병합을 사용하는 것이 더 좋을 수 있음

브랜치 목록

- 브랜치 목록 확인
  - git branch
    - 앞에 *가 붙어있으면 HEAD 브랜치임

```
for@Eunsang MINGW64 /c/GitDirectory1 (master)
$ git branch
  branch1
* master
```

  - git branch –v
    - 마지막 커밋도 함께 표시됨

```
for@Eunsang MINGW64 /c/GitDirectory1 (master)
$ git branch -v
  branch1 59bef41 Commit1
* master  59bef41 Commit1
```

브랜치 생성

- git branch [브랜치이름]
  - 브랜치를 생성함. HEAD 위치에서 브랜치를 생성
  - 아직, 그 브랜치로 체크아웃하지는 않음
  - ex) git branch branch1
- git branch [브랜치이름] [커밋체크섬]
  - 해당 커밋 체크섬으로부터 브랜치를 생성
- git branch –f [브랜치이름] [커밋체크섬]
  - 이미 있는 브랜치를 다른 커밋으로 옮기고 싶을 때에는 –f 옵션을 주어야함

브랜치 생성

- git checkout –b [브랜치이름]
  - 브랜치를 생성하면서 그 브랜치로 이동함
- git checkout –b [브랜치이름] [기준브랜치명]
  - 기준브랜치로부터 브랜치를 새로 생성
- git checkout –b [브랜치이름] [커밋체크섬]
  - 브랜치를 생성하면서 그 브랜치로 이동
  - 해당 커밋체크섬 커밋에서 브랜치를 생성함

브랜치 삭제

- 현재 작업하고 있는 브랜치(HEAD)는 삭제할 수 없음
- 따라서 먼저 삭제하지 않을 다른 브랜치로 체크아웃함
- 그 후, 다음 브랜치 삭제 명령어 입력
  - git branch –d [브랜치명]
  - ex) git branch –d branch1
  - -d 옵션 대신 --delete로 써도 무방
- 만약 브랜치에 병합되지 않은 변경사항 및 푸시되지 않은 커밋이 있을 경우 –d 옵션으로 삭제할 수 없음
- 이 경우 아래 명령어로 강제 브랜치 삭제 가능
  - git branch –D [브랜치명]
  - ex) git branch –D branch1

브랜치 삭제

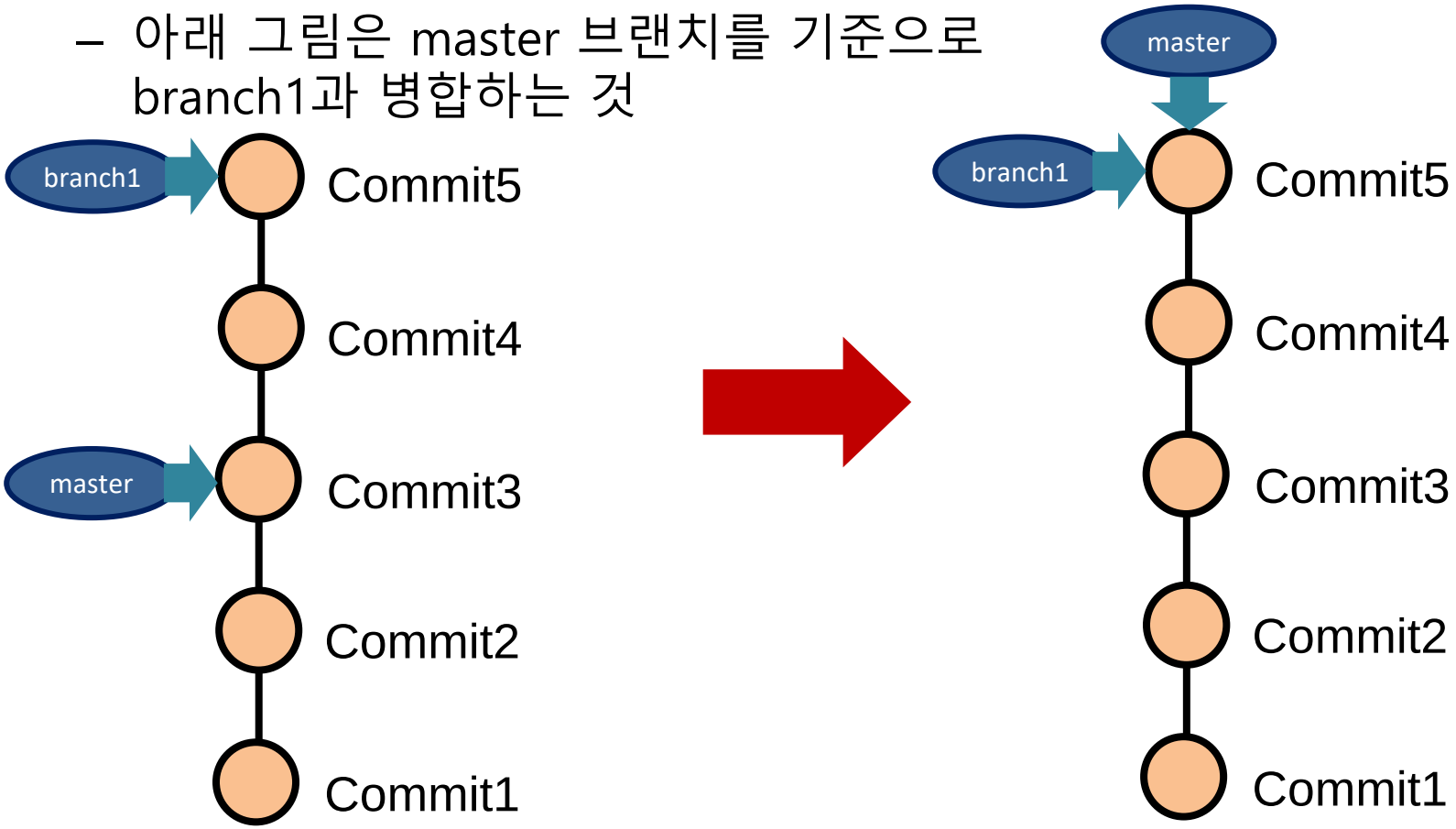
- 원격브랜치와 로컬브랜치는 독립적인 개체
- 원격에서 브랜치를 삭제하려면 다음과 같이 가능
  - git push origin –d [브랜치명]
  - ex) git push origin --delete origin/branch1
- 혹은 다음 명령어로도 가능
  - git push origin :[브랜치명]
  - ex) git push origin :origin/branch1

병합

- branch1 브랜치를 기준으로 branch2를 병합한다고 함
- 이 때, branch1 브랜치가 기준이라는 것은 병합한 결과물을 branch1 브랜치에 반영한다는 의미
- branch1 브랜치로 체크아웃한 후 아래 명령어 입력
  - git merge branch2

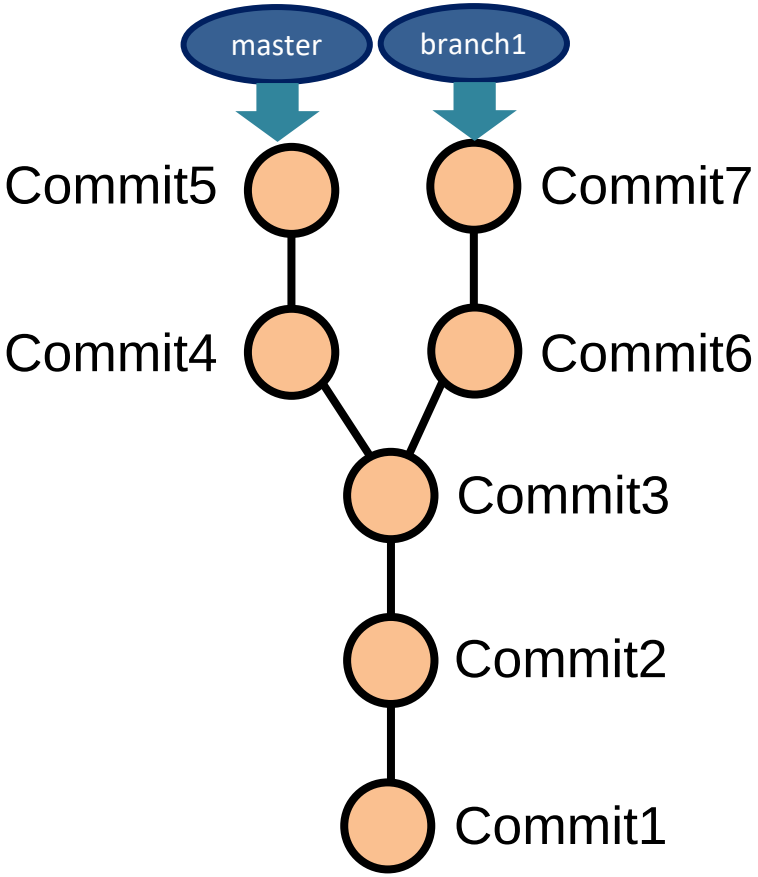
병합 유형

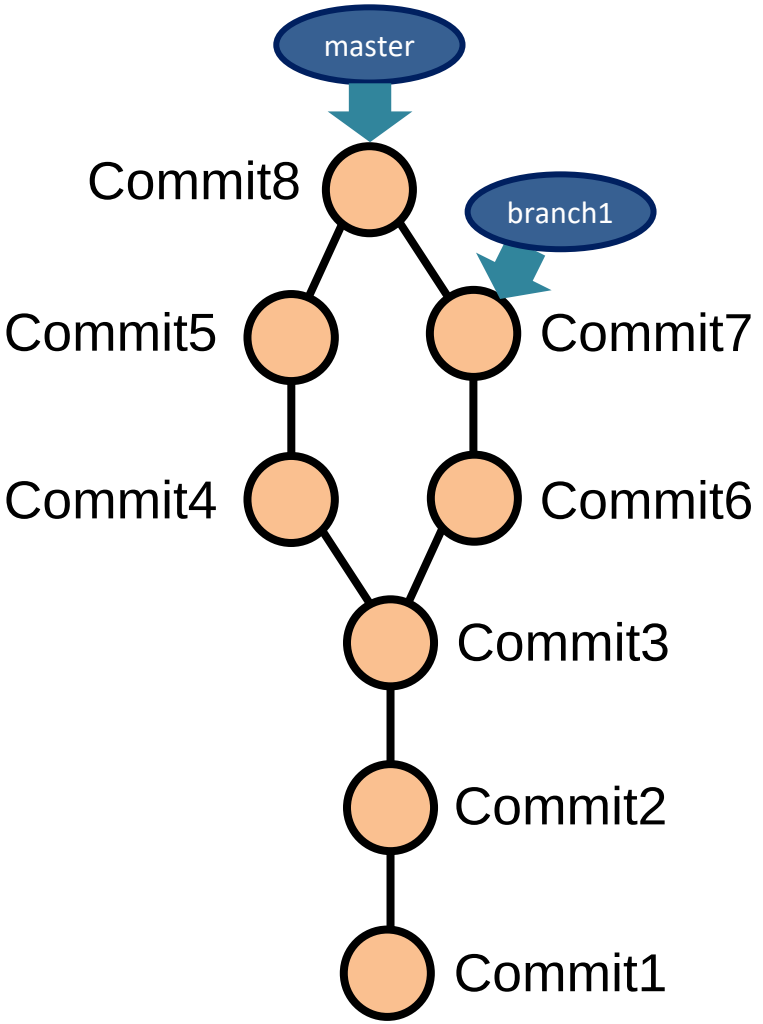
- 빨리감기 병합
  - 합친 결과물이 뒷 상태와 동일한 것
  - fast-forward라고 함
  - 아래 그림은 master 브랜치를 기준으로 branch1과 병합하는 것



병합 유형

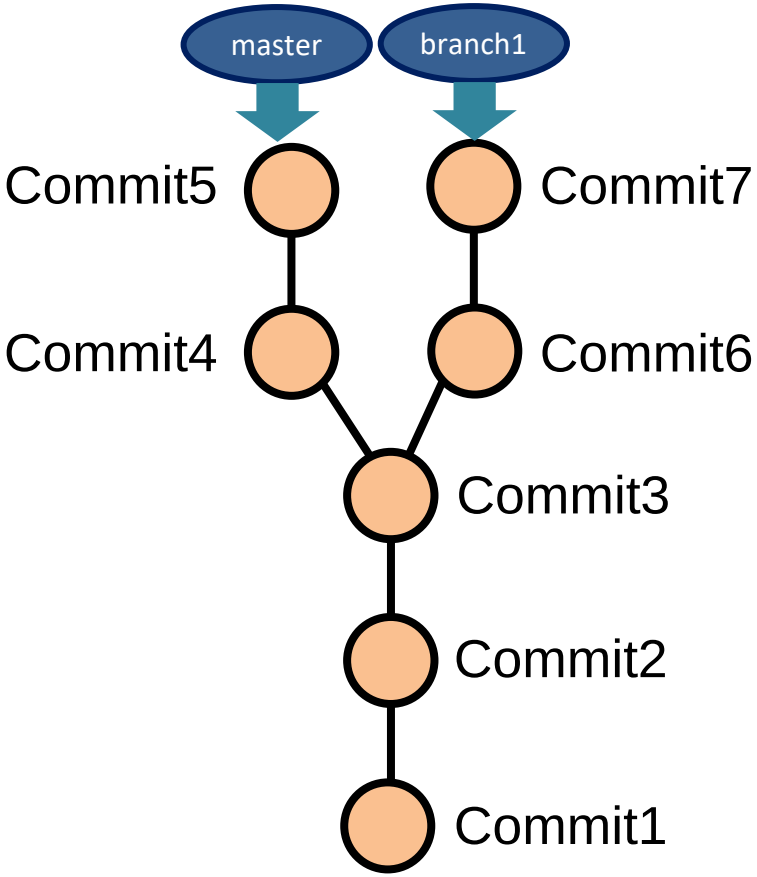
- 일반 병합

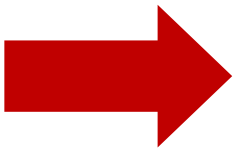

master
branch1
Commit5
Commit7
Commit4
Commit6
Commit3
Commit2
Commit1


master
Commit8
branch1
Commit5
Commit7
Commit4
Commit6
Commit3
Commit2
Commit1

병합 유형

- 충돌

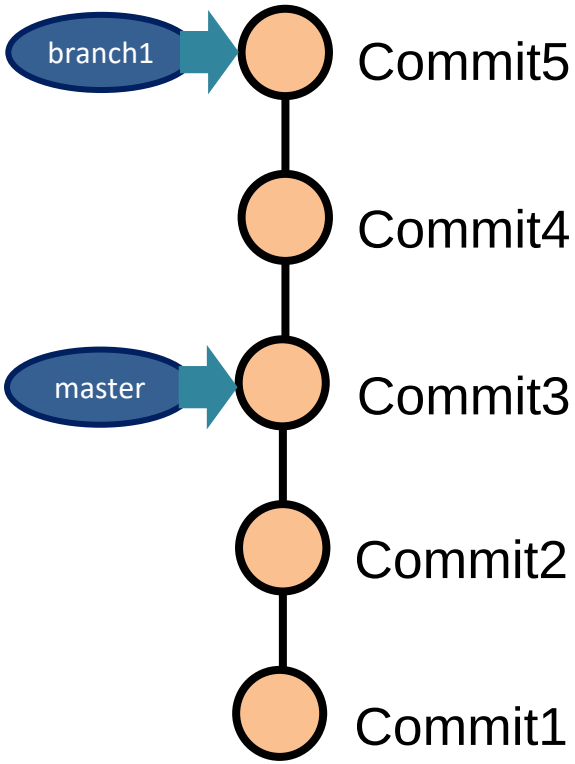




충돌로 병합 불가능

병합 실습 – 빨리감기 병합

- 아래와 같은 상황에서 master 브랜치로 체크아웃함
- 그 후 git merge branch1 입력



파일내용	GitFile1.txt
Commit5	Contents5
Commit4	Contents4
Commit3	Contents3
Commit2	Contents2
Commit1	Contents1

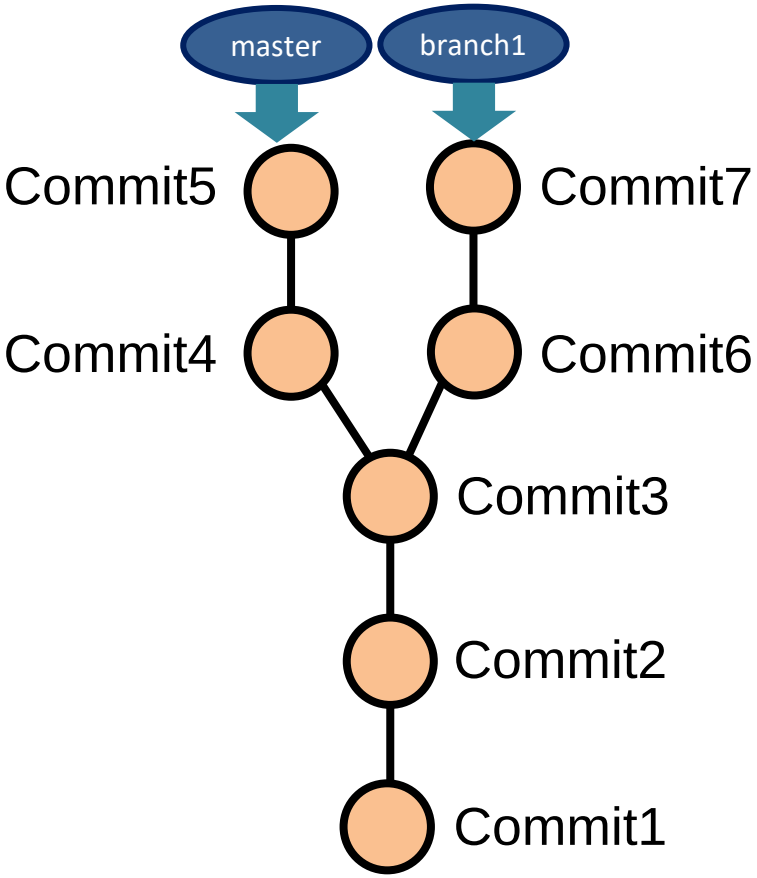
병합 실습 – 빨리감기 병합

```
for@Eunsang MINGW64 /c/GitDirectory1 (branch1)
$ git log
commit d2bc01a8c905b5fccd33e4df6721c8c2d2fd08ec (HEAD -> branch1)
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 13:09:54 2023 +0900

    Commit5

commit e4765a0187e83ea1bc7de057c2f9f6dd677f9245
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 13:09:07 2023 +0900

    Commit4

commit fb3f4c116b8149707c164bb7386ecc33cc6c038a (master)
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 13:08:29 2023 +0900

    Commit3

commit 6f44c39a162939bf8590f924981146a9428c1b0a
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 13:08:11 2023 +0900

    Commit2

commit 59bef4121a8ddf85d71fac6734569db568ab6363
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 11:50:51 2023 +0900

    Commit1
```

```
for@Eunsang MINGW64 /c/GitDirectory1 (master)
$ git log
commit d2bc01a8c905b5fccd33e4df6721c8c2d2fd08ec (HEAD -> master, branch1)
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 13:09:54 2023 +0900

    Commit5

commit e4765a0187e83ea1bc7de057c2f9f6dd677f9245
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 13:09:07 2023 +0900

    Commit4

commit fb3f4c116b8149707c164bb7386ecc33cc6c038a
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 13:08:29 2023 +0900

    Commit3

commit 6f44c39a162939bf8590f924981146a9428c1b0a
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 13:08:11 2023 +0900

    Commit2

commit 59bef4121a8ddf85d71fac6734569db568ab6363
Author: eslee3209 <eslee3209@sejong.ac.kr>
Date:   Wed Mar 15 11:50:51 2023 +0900

    Commit1
```

병합 실습 – 일반 병합

- 아래와 같은 상황에서 master 브랜치로 이동
- master 브랜치를 기준으로 branch1과 병합
  - git merge branch1



파일내용	GitFile1.txt	GitFile2.txt
Commit7	Contents3	Contents2
Commit6	Contents3	Contents1
Commit5	Contents5	
Commit4	Contents4	
Commit3	Contents3	
Commit2	Contents2	
Commit1	Contents1	

병합 실습 – 일반 병합

- 병합할 때 메시지를 적을 수 있는데 Commit8이라 적고 저장

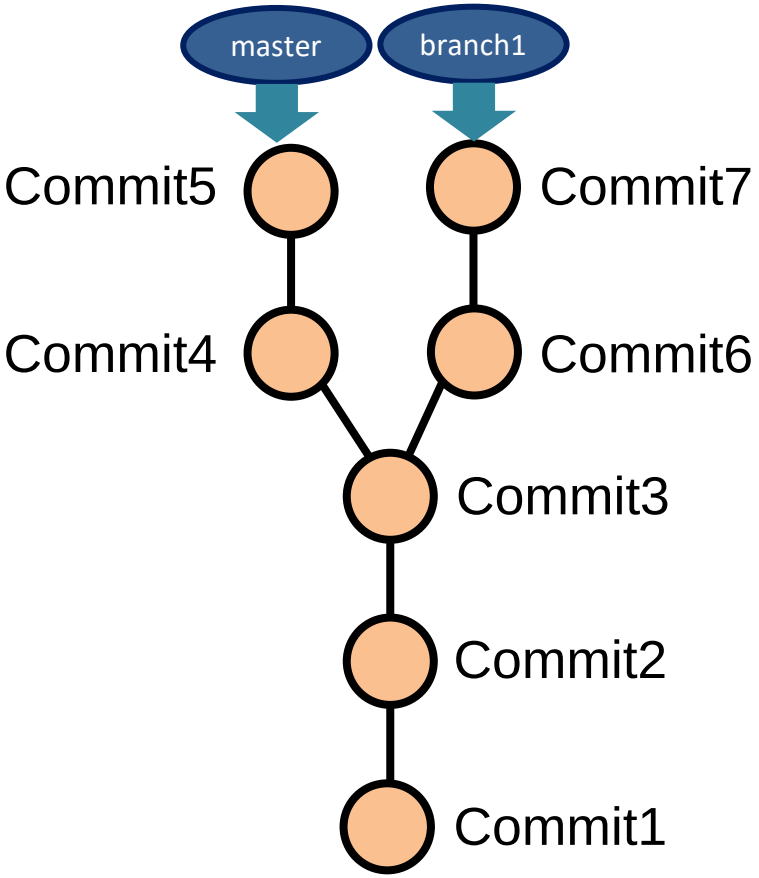
```
for@Eunsang MINGW64 /c/GitDirectory1 (master)
$ git log --oneline --graph --all
* 2400fc2 (branch1) Commit7
* 341898c Commit6
| * d2bc01a (HEAD -> master) Commit5
| * e4765a0 Commit4
|/
* fb3f4c1 Commit3
* 6f44c39 Commit2
* 59bef41 Commit1
```

```
for@Eunsang MINGW64 /c/GitDirectory1 (master)
$ git log --oneline --graph --all
*   3557c1b (HEAD -> master) Commit8
|\
| * 2400fc2 (branch1) Commit7
| * 341898c Commit6
* | d2bc01a Commit5
* | e4765a0 Commit4
|/
* fb3f4c1 Commit3
* 6f44c39 Commit2
* 59bef41 Commit1
```

파일내용	GitFile1.txt	GitFile2.txt
Commit8	Contents5	Contents2
Commit7	Contents3	Contents2
Commit6	Contents3	Contents1
Commit5	Contents5	
Commit4	Contents4	
Commit3	Contents3	
Commit2	Contents2	
Commit1	Contents1	

병합 실습 - 충돌 해결

- 아래와 같은 상황에서 master 브랜치로 이동
- master 브랜치를 기준으로 branch1과 병합
  - git merge branch1



파일내용	GitFile1.txt
Commit7	Contents7
Commit6	Contents6
Commit5	Contents5
Commit4	Contents4
Commit3	Contents3
Commit2	Contents2
Commit1	Contents1

병합 실습 - 충돌 해결

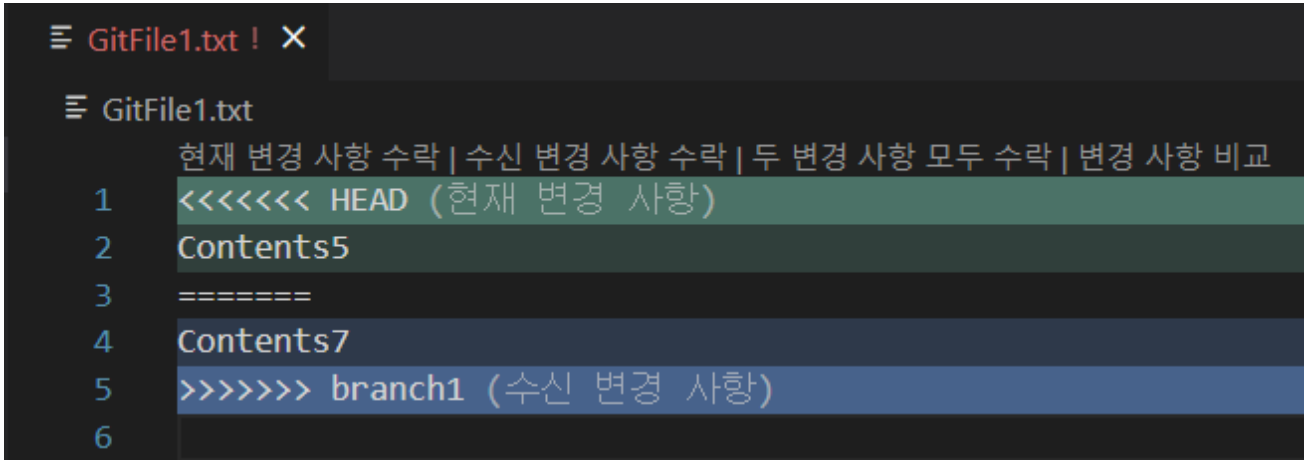
```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --oneline --graph --all
* becce7f (origin/branch1, branch1) Commit7
* bfbf235 Commit6
| * 6fa555c (HEAD -> master, origin/master) Commit5
| * 2426a63 Commit4
|/
* 0ee4a91 Commit3
* 046bb48 Commit2
* 78d6900 Commit1
```

- git merge 명령어 입력 시 충돌 발생

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git merge branch1
Auto-merging GitFile1.txt
CONFLICT (content): Merge conflict in GitFile1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

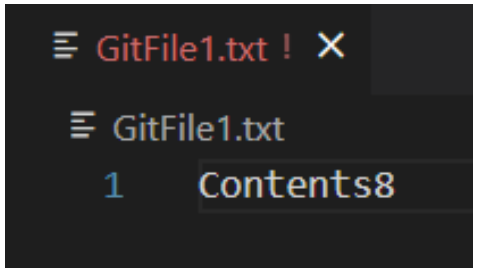
병합 실습 - 충돌 해결

- VSCode에서 GitFile1.txt 파일을 열면 git이 충돌이 난 코드를 자동으로 마크함
- "현재 변경 사항"은 기준으로 한 브랜치(master)에서의 변경사항
- "수신 변경 사항"은 기준 브랜치에 합치려는 브랜치(branch1)에서의 변경사항



병합 실습 - 충돌 해결

- 충돌한 코드를 "Contents8" 내용으로 수정
- 아래와 같이 '<<<<<<<HEAD', '>>>>>>master', '========='와 같은 불필요한 라인 다 지우고 "Contents8"로 수정



병합 실습 - 충돌 해결

- CLI 창에서 들어가 수정된 파일을 스테이지에 올리고 커밋/push 진행
  - git add .
  - git commit -m "Commit8"
  - git push origin master
- 그러면 아래와 같이 성공적으로 병합이 완료됨

```
USER@BOOK-1C7BTP524F MINGW64 /c/GitDirectory1 (master)
$ git log --oneline --graph
*   006d1b2 (HEAD -> master, origin/master) Commit8
|\
| * becce7f (origin/branch1, branch1) Commit7
| * bfbf235 Commit6
* | 6fa555c Commit5
* | 2426a63 Commit4
|/
* 0ee4a91 Commit3
* 046bb48 Commit2
* 78d6900 Commit1
```

명령어 요약

- 1. 체크아웃
  - 브랜치A로 checkout
    - git checkout A
- 2. 브랜치 생성
  - 브랜치A 생성 (checkout x)
    - git branch A
    - ex) git branch master
  - 브랜치A 생성 및 그 브랜치로 checkout
    - git checkout –b A
    - ex) git checkout –b master

명령어 요약

- 3. 브랜치 삭제
  - 브랜치A 삭제
    - 삭제하지 않을 다른 브랜치로 checkout
    - git branch –d A (이걸로 삭제 안 되는 경우 git branch –D A)
- 4. 원격브랜치 삭제
  - 원격저장소 브랜치A 삭제
    - git push origin –d origin/A
    - ex) git push origin –d origin/branch1

명령어 요약

- 5. 병합
  - 브랜치A를 기준으로 브랜치B를 병합
    - git checkout A
    - git merge B