
Open source Software
오픈소스 소프트웨어
14. 브랜치 개요와 관리

학습 개요

1. 브랜치 필요성
2. 브랜치 생성과 이동
3. 브랜치 목록 확인과 삭제

학습 목표

1. 브랜치를 이해하고 활용할 수 있다.
2. 브랜치 목록을 파악하고 필요하면 생성할 수 있다.
3. 생성된 브랜치로 이동해 커밋할 수 있다.
4. 필요없는 브랜치를 삭제할 수 있다.

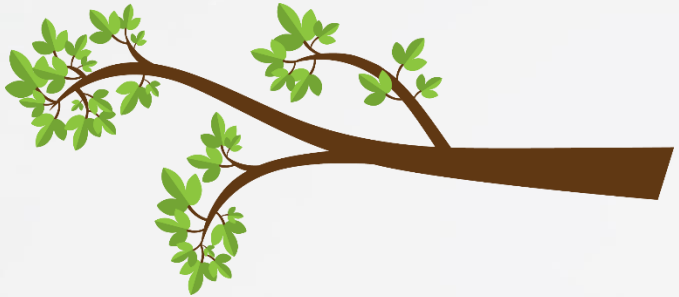
LESSON 01

브랜치 개요

1 브랜치 개요

브랜치

실 행활에서 용어



- Branch
  - 나무의 가지, 지점

깃 브랜치

- 파일 작성 작업을 하다 보면 여러 파일을 관리하는 폴더를 통째로 복사해 활용하는 일이 자주 발생
- 버전 관리를 수행하던 일련의 파일 집합을 통째로 복사해 독립적으로 다시 개발을 진행하는 개념
  - 여러 개발자가 타인을 신경 쓰지 않고 동시에 다양한 작업을 할 수 있게 만들어 주는 기능
  - 브랜치를 통해 하나의 프로젝트를 여러 갈래로 나누어서 관리
- 브랜치 병합(merge)
  - 독립된 브랜치에서 마음대로 소스 코드를 변경하여 작업한 후 원래 버전과 합칠 수 있음

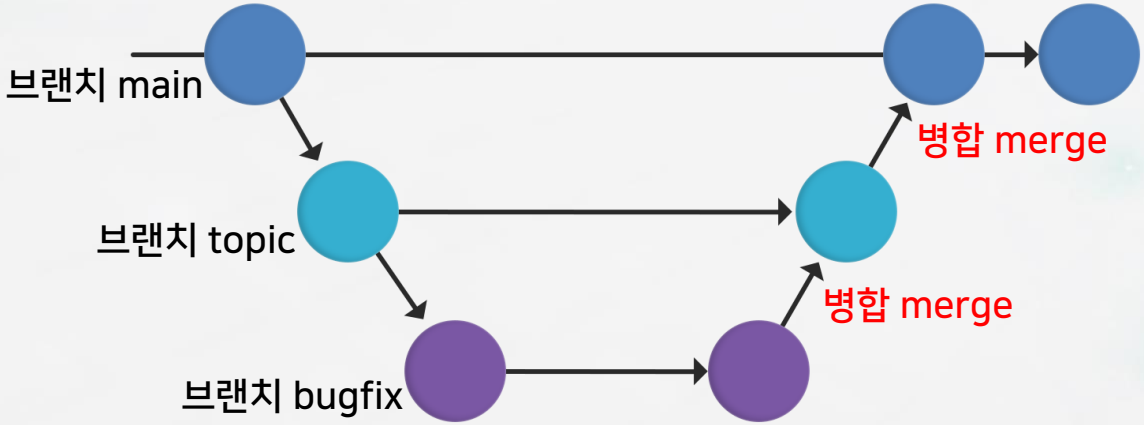
브랜치와 병합

브랜치 사용 장점

- 저장소에서 다른 브랜치에는 영향 없이
  - 새로운 기능을 개발하거나 버그를 수정
  - 새로운 아이디어를 안전하게 실험 가능

브랜치 병합

- 브랜치와 브랜치를 합치는 수행



1 브랜치개요

기본 브랜치

기본 브랜치

- 저장소 생성 시 처음 만들어지는 브랜치
  - main
    - ➡ 예전에는 master

기본 브랜치 이름 설정

- 기본 설정으로 수정
  - $ git config --global init.defaultBranch main

이미 생성된 저장소의 브랜치 이름 수정

- $ git branch -M newBname

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (main)
$ git branch -M master
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (master)
$
```

브랜치와 HEAD

브랜치

- 커밋 사이를 가볍게 이동할 수 있는 포인터

HEAD

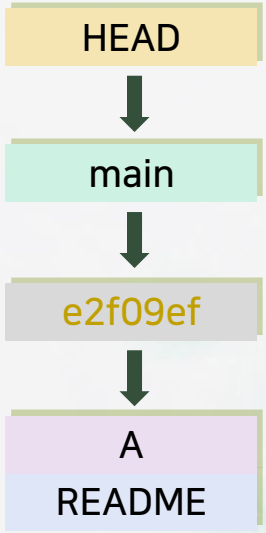
- 작업 중인 브랜치의 최신 커밋을 가리키는 포인터
  - 필요에 따라 특정 커밋이나 브랜치를 가리키도록 헤드 이동 가능
    - ➡ checkout, switch 명령 사용

결과 표시 (HEAD -> main)

- ‘main은 마지막 커밋을 가리키고,
  HEAD는 현재 작업 브랜치인 main을 가리킨다’라는 의미

처음 커밋

- main 브랜치는 생성된 커밋을 가리킴
- 커밋이 계속 발생
  - main 브랜치는 자동으로 가장 마지막 커밋을 가리킴



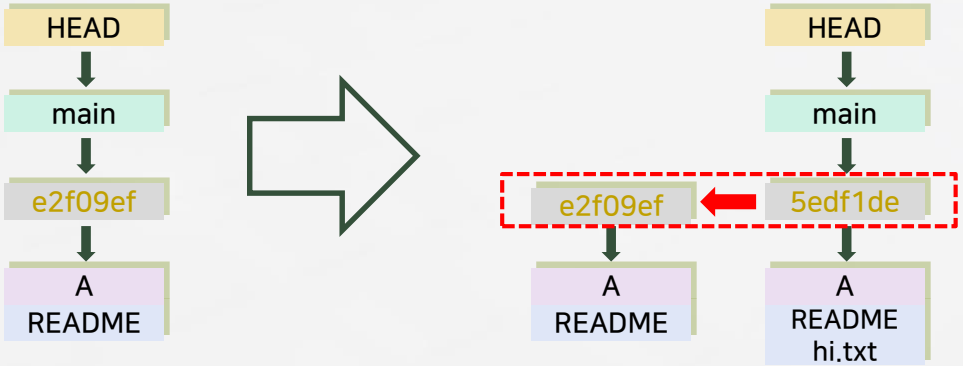
두 번째 커밋 이후의 로그 이력

브랜치 이름 main

- 마지막 커밋을 가리키고
- HEAD
  - 그 main을 가리킴

커밋 이력

- 커밋 번호인 커밋 ID(16진수 40개)
  - 다시 그 이전 커밋을 부모로 가리킴

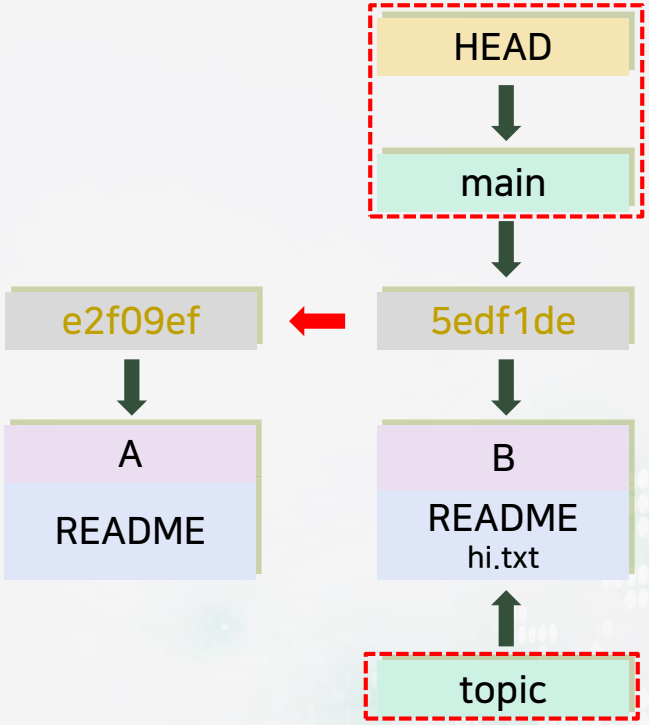


새로운 브랜치 생성

생성

- 단순히 생성하고 HEAD가 이동은 안함
  - $ git branch bname
- 생성하고 새 브랜치로 HEAD 이동도 수행
  - $ git switch -c bname
  - $ git checkout -b bname

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (main)
$ git branch topic
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (main)
$ git branch
* main
  topic
```



브랜치 확인

명령 git branch로 현재 저장소의 브랜치 목록을 확인

주요 명령	
$ git branch	커밋이 발생한 브랜치 목록 보이기
$ git branch -v	브랜치마다 마지막 커밋 ID와 메시지도 함께 표시

결과 표시 * main

- *은 현재 작업하고 있는 브랜치를 표시

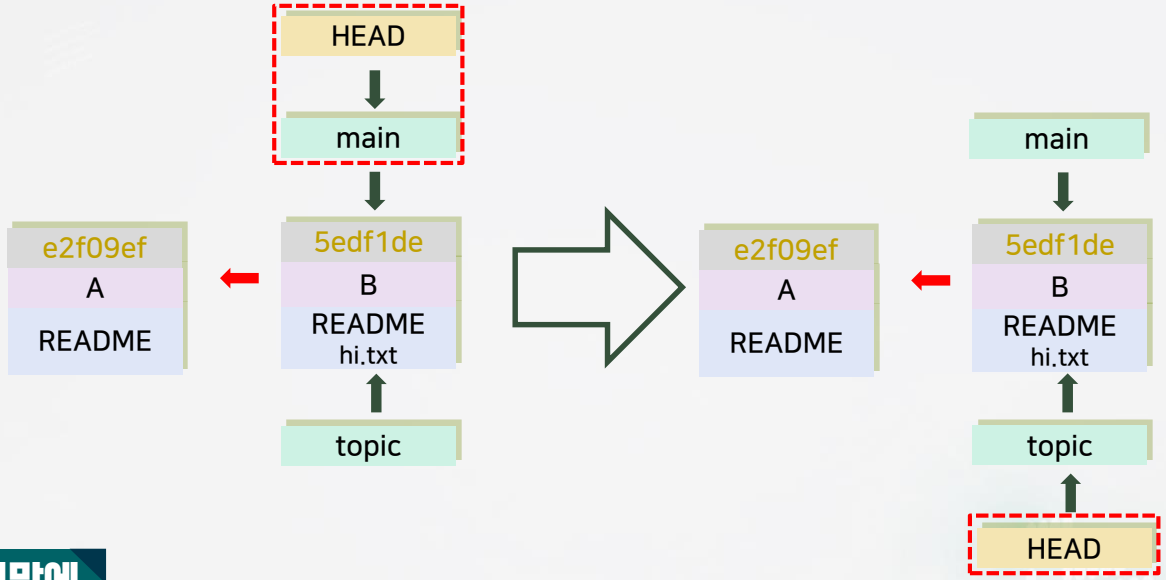
```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (main)
$ git branch topic
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (main)
$ git branch
* main
  topic
```

1 브랜치개요

생성된 새로운 브랜치로 이동

이동

- HEAD를 지정한 브랜치로 이동
  - $ git switch [bname]
  - $ git checkout [bname]
- HEAD를 이전 브랜치로 이동
  - $ git switch -
  - $ git checkout -



항상 첫 줄의 마지막에 브랜치이름이 표시

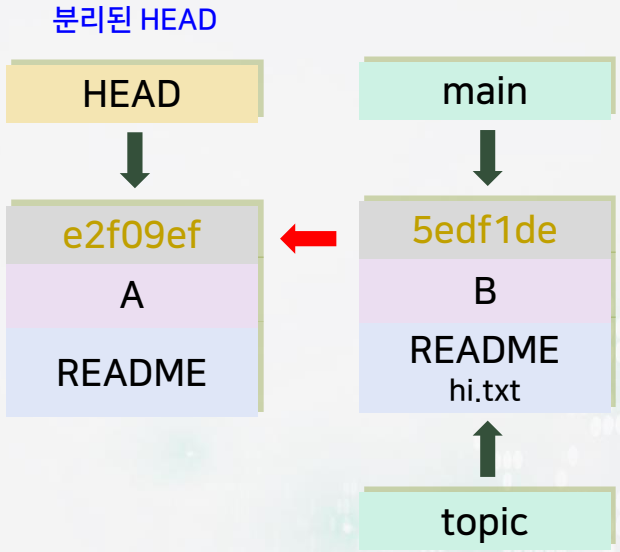
```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (main)
$ git switch topic
Switched to branch 'topic'
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (topic)
$
```

분리된(detached) HEAD

- HEAD가 현재 브랜치(마지막 커밋)가 아닌 그 이전 커밋을 가리키는 상태

- $ git checkout HEAD~
  - 현재 브랜치에서 마지막 커밋 이전 커밋으로 이동

- 이동 커밋
  - 상대적인 HEAD~ 또는 HEAD^^, commitID를 사용
    - 물결 ~: tilde(틸드)
    - 삿갓, 모자 ^; caret(커렛)



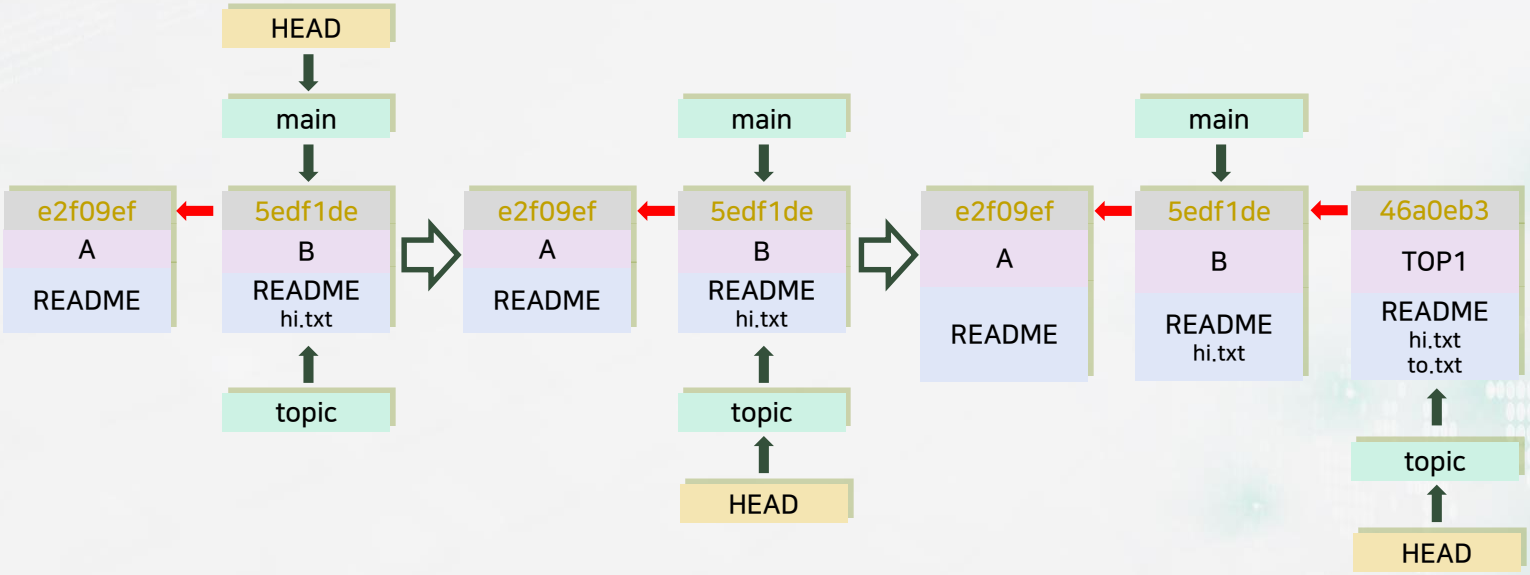
LESSON 02

브랜치 관리

2 브랜치 관리

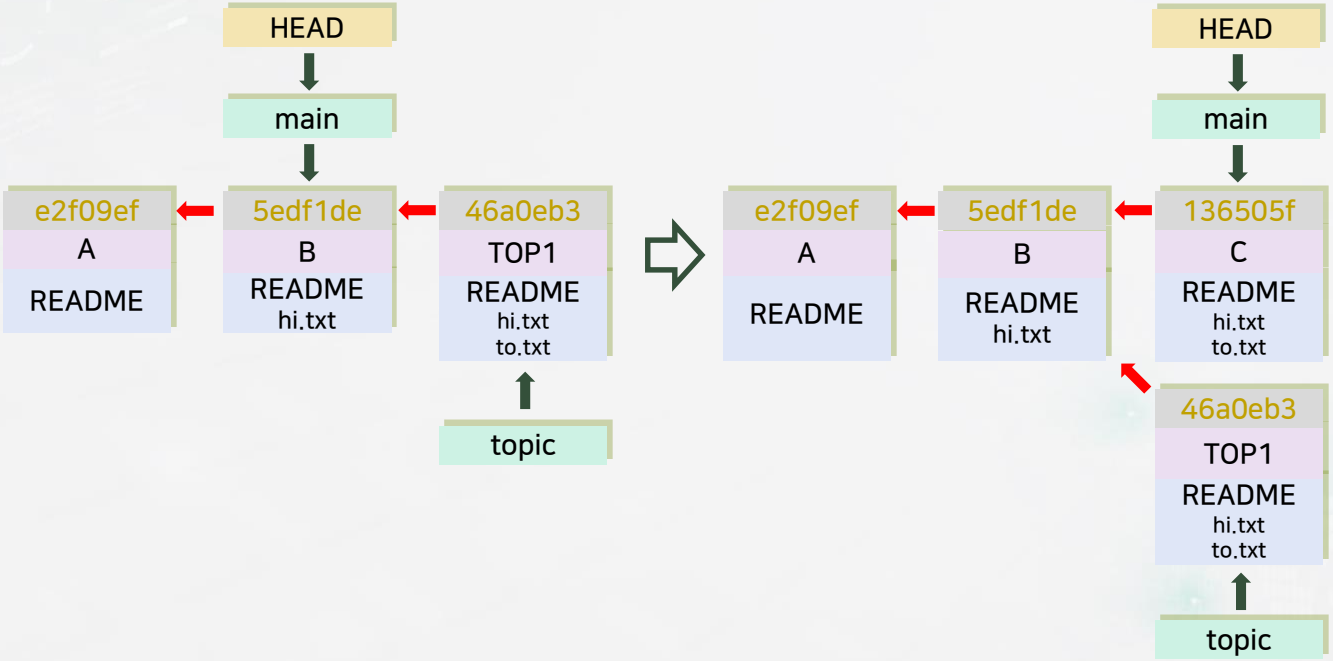
새 브랜치에서 커밋

- 브랜치 main에서 topic 브랜치 생성
- 브랜치 topic으로 이동
- 브랜치 topic에서 커밋



기본 브랜치로 이동 후 다시 커밋

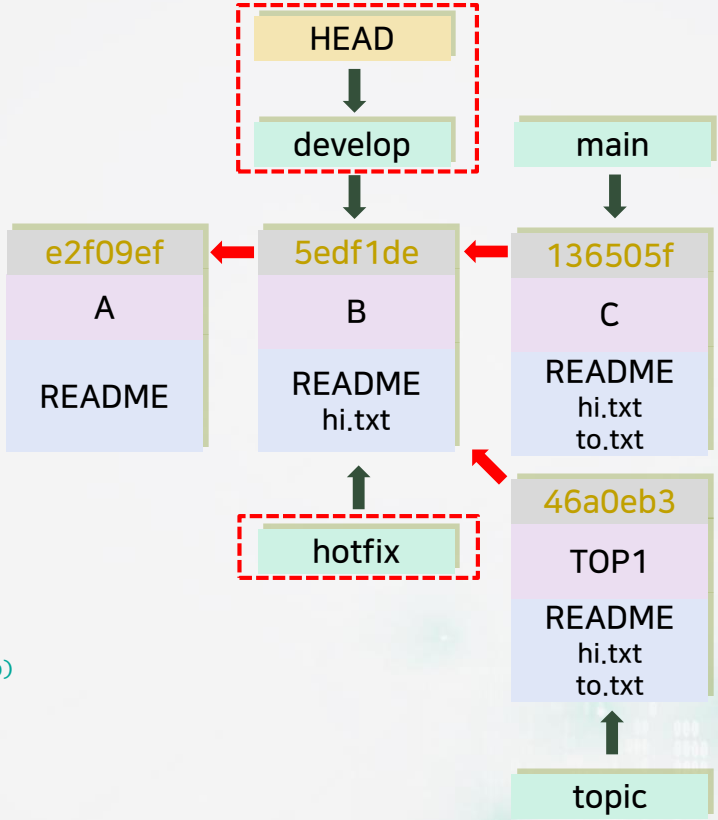
- 브랜치 main으로 이동
- 브랜치 main에서 커밋



2 브랜치 관리

최신 커밋 이전에서 두 브랜치 생성

- $ git switch main
  - HEAD를 main으로 이동
- $ git checkout HEAD~
  - HEAD를 하나 이전으로 이동
- $ git switch -c hotfix
  - 브랜치 hotfix 생성하고 HEAD 이동
- $ git checkout -b develop
  - 브랜치 develop 생성하고 HEAD 이동




```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (develop)
$ git branch
* develop
hotfix
main
topic
```

최신 커밋 이전에서 두 브랜치 생성

명령 branch에서 옵션 -D 또는 -d를 사용

- 아직 병합되지 않은 브랜치라면 강제로 삭제하기 위해 옵션 -D를 사용

주요 명령	
`$ git branch [-d | --delete] [branchName]`	지정한 branchName(이미 병합된)을 삭제
`$ git branch -D [branchName]`	지정한 branchName(병합되지 않더라도)을 삭제

브랜치 목록 보기

- 브랜치의 병합 여부를 옵션 --merged와 --no-merged로 확인 가능
  - 현재 작업 브랜치는 일반적으로 --merged 브랜치

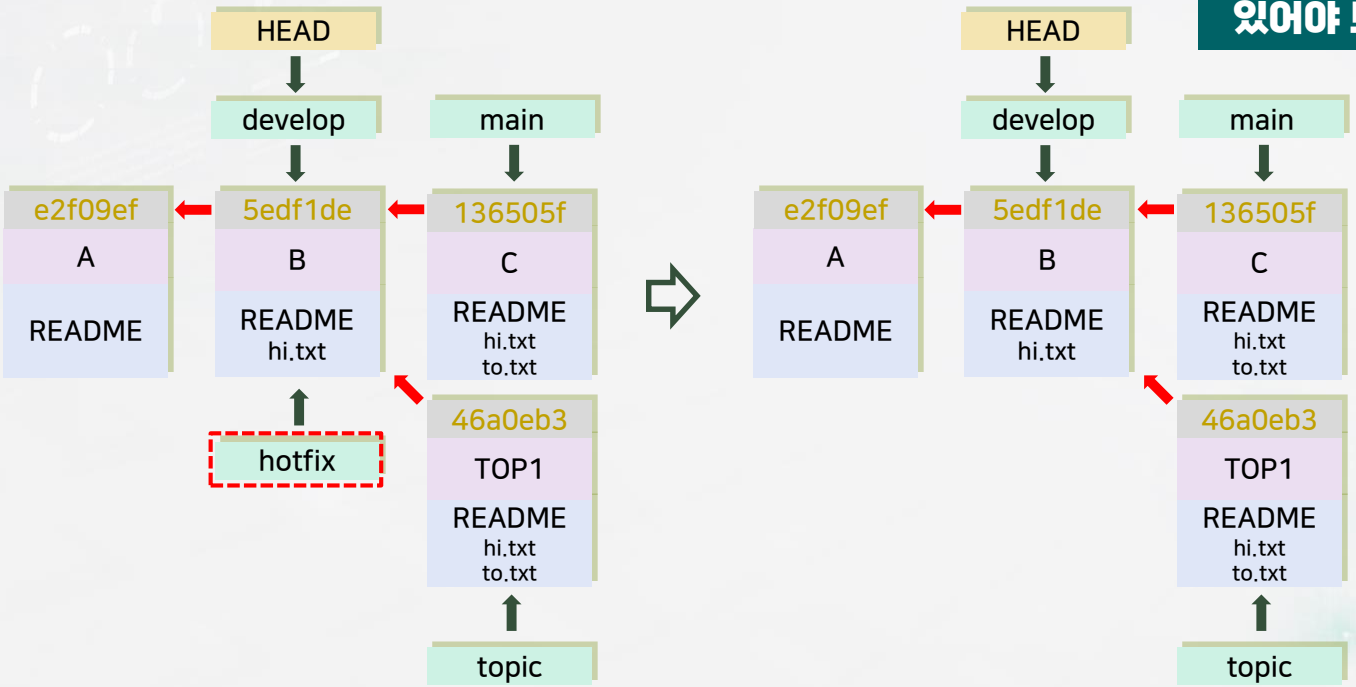
주요 명령	
`$ git branch --merged`	현재 작업 브랜치를 기준으로 병합된 브랜치 목록 표시
`$ git branch --no-merged`	현재 작업 브랜치를 기준으로 아직 병합되지 않은 브랜치 목록 표시
`$ git branch --merged branchName`	인자인 branchName 브랜치를 기준으로 병합된 브랜치 목록 표시
`$ git branch --no-merged branchName`	인자인 branchName 브랜치를 기준으로 아직 병합되지 않은 브랜치 목록 표시

브랜치 삭제 수행

- 현재 branch에서 다른 브랜치 삭제

```
PC@DESKTOP-482NOAB MINGW64 /c/[smart Git]/gbrch (develop)
$ git branch -d hotfix
Deleted branch hotfix (was 5edf1de).
```

다른 브랜치 develop에 위치하고 있어야 브랜치 hotfix를 삭제 가능



브랜치 관련 명령 정리

- $ git checkout bname
  - 전환, 이동
- $ git switch bname
  - 전환, 이동
- $ git checkout -
  - 이전 브랜치로 전환, 이동

브랜치 관련 명령 정리

$ git switch -

- 이전 브랜치로 전환, 이동

$ git branch –h

- 도움말 보기

설명	git checkout	git switch
	⬇	⬇
이전 커밋으로 이동	$ git checkout [이전커밋]	$ git switch -d [이전커밋]
다른 브랜치로 이동	$ git checkout [branch]	$ git switch [branch]
	⬇	⬇
새로운 브랜치를 생성하고 이동	$ git checkout -b [newBranch]	$ git switch -c [newBranch]

Summary

- $ git branch
  - 저장소 목록 보기
- $ git branch ⟨new-branch⟩
  - 저장소 생성만
- $ git checkout -b ⟨new-branch⟩
  - 저장소 생성하고 이동
- git switch –c ⟨new-branch⟩
  - 저장소 생성하고 이동

Summary

- git branch –d branch-name
  - 저장소 삭제
- $ git branch –D branch-name
  - 저장소 삭제, 강제 삭제