3-4. 브랜치 병합하기

만들어진 **각 branch에서 작업을 하다가 어느 시점에서는 branch 작업을 마무리하고 기존 branch와 합해야 한다.** 이것을 **‘branch merge’**라고 한다. **branch와 branch를 merge하다 보면 여러 상황이 생길 수 있는데 각 상황마다 병합하는 방법을 알아보자.** 그리고 병합하면서 branch사이에 충돌이 있을 때 해결하는 방법도 살펴보자.

**\* 서로 다른 파일 병합하기**

branch merge는 처음에 공부하기 까다롭기 때문에 새로운 저장소를 만들어서 필요한 branch와 commit만 사용해보자.

1. 터미널 창을 열고 홈 디렉터리로 이동한다. manual-2라는 깃 저장소를 만들어보자. **git init 다음에 디렉터리 이름을 입력**하면 **새로운 디렉터리를 만들고 저장소를 초기화하는 과정을 한꺼번에 처리할 수 있다.** manual-2 디렉터리로 이동한 후 **ls -al 명령**을 사용하면 ‘.git/’ 디렉터리가 만들어진 것을 확인할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

2. 빔에서 work.txt 파일을 만들고 ‘1’이라는 내용을 추가한 후 **commit(commit message: work1)**을 수행하자.

3. **o2라는 branch를 만들어보자.**

텍스트이(가) 표시된 사진

자동 생성된 설명

4. 현재 master branch에 master.txt라는 파일을 하나 더 만들자. 빔에서 ‘master 2’라는 내용을 입력하고 **commit(commit message: master work2)**을 수행하자.

5. 이제 **o2 branch로 checkout**하자.

텍스트이(가) 표시된 사진

자동 생성된 설명

6. o2 branch에서 o2.txt 파일을 만들고 ‘o2 2’라는 내용을 저장하자. 그리고 **commit(commit message: o2 work2’)**을 수행하자.

7. **git log 명령**을 사용해 현재 **commit 상태를 확인**해 보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

**‘work1’ commit은 master branch와 o2 branch가 똑같이 가지고 있다.** 그 다음으로 **master branch에는 ‘master work2’ commit이 생겼고, o2 branch에는 ‘o2 work2’ commit이 생겼다.**

8. o2 branch에서 작업이 다 끝났다고 가정하고, **o2 branch의 내용을 master branch로 병합**해보자. **branch를 merge하려면 master branch로 checkout해야한다.**

텍스트이(가) 표시된 사진

자동 생성된 설명

9. branch를 merge하려면 **git merge 명령** 뒤에 가져올 branch 이름을 적는다. **master branch에 o2 branch를 가져와 merge해보자.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**(참고) branch를 merge할 때 편집기 창 열리게 하기**

branch를 merge할 때 편집기를 사용하려면 아래와 같이 **--edit 옵션**을 추가하면 된다.

**\* git merge o2 --edit**

편집기를 사용하지 않으려면 아래와 같이 입력하면 된다.

**\* git merge o2 --no-edit**

10. **ls -al 명령**을 사용해 확인해 보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

o2 branch에 있던 **o2.txt 파일이 master branch에 합쳐져 있다.**

**git log 명령**을 이용해 branch와 commit들이 어떻게 merge되었는지 확인해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

‘o2 work2’ commit이 master branch에 merge되면서 **‘merge branch o2’라는 새로운 commit이 생겼다.**

**\* 같은 문서의 다른 위치를 수정했을 때 병합하기**

**master branch와 o2 branch 에는 똑같이 work.txt 파일이 있다.** 양쪽 branch에서 work.txt 문서를 수정하되 **서로 다른 위치를 수정한 후 branch를 merge했을 때 어떤 결과가 나오는지 확인**해보자.

1. **홈 디렉터리로 이동한 후 manual-3이라는 깃 저장소를 만들고 해당 디렉터리로 이동**하자.

2. **work.txt 파일을 만들고 내용을 입력**해보자. 나중에 work.txt 문서를 수정하고 merge할 것이기 때문에 내용 사이에 두 줄의 공백을 넣었다.

(work.txt에 입력한 내용)

3. **commit(commit message: work1)을 수행**하자.

4. **o2라는 새로운 branch를 만들어보자.**

텍스트이(가) 표시된 사진

자동 생성된 설명

이제 **master branch와 o2 branch에는 모두 ‘work1’ commit이 있게 된다.**

5. **master branch의 work.txt 내용을 수정**해보자. 공백 사이에 ‘master content 2’ 텍스트를 추가하고 **commit(commit message: master work2)을 수행**한다.

6. **o2 branch의 work.txt 내용도 수정**해보자. 먼저 o2 branch로 checkout하고 work.txt에 두번째 ‘content’ 다음줄에 ‘o2 content 2’라고 입력하자. 그리고 **commit(commit message: o2 work2)**을 수행해주자.

7. **master branch와 o2 branch를 병합**해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

**master branch와 o2 branch 모두 work.txt 파일을 수정했지만 문서 안의 수정 위치는 다르다.** work.txt파일의 내용이 어떻게 바뀌었는지 확인해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

master branch와 o2 branch의 work.txt 파일 내용이 자동으로 합쳐졌다. 이렇게 branch를 자동으로 합쳐주는 기능 때문에 깃은 더욱 강력한 도구가 된다.

**\* 같은 문서의 같은 위치를 수정했을 때 병합하기**

**깃에서는 줄 단위로 변경 여부를 확인한다.** 그래서 **각 branch에 같은 파일 이름을 가지고 있으면서 같은 줄을 수정했을 때 branch를 merge하면 ‘branch conflict’가 발생**한다. master branch와 o2 branch에서 같은 파일의 같은 위치를 수정한 후 merge해 보면서 **어떤 경우에 conflict가 생기는지, 그리고 어떻게 conflict를 해결하는지 알아보자.**

1. 홈 디렉터리로 이동한 후 **manual-4라는 깃 저장소를 만들고 해당 디렉터리로 이동**하자.

2. **work.txt 파일을 만들고 이전과 같이 입력하고 commit(commit message: work1)을 실행**하자. 단, 공백은 이전과 다르게 1줄만 두도록 하자.

3. **o2라는 branch를 만들고 master의 work.txt를 수정**하자. 공백에 ‘master content 2’를 입력한 후 **commit(commit message: master work2)을 실행**하자.

4. **o2의 work.txt도 수정**해주자. 공백에 ‘o2 content 2’를 입력하고 **commit(commit message: o2 work2)을 실행**하자.

5. **master branch에 o2 branch를 merge해보자.**

텍스트이(가) 표시된 사진

자동 생성된 설명

merge 명령을 실행하면 다음과 같은 메시지가 출력된다. 위 메시지는 **work.txt에 Auto-merge를 하는 동안 conflict가 발생했다는 의미**이다.

**충돌이 생긴 문서는 자동으로 merge가 될 수 없으므로 사용자가 직접 conflict 부분을 해결한 후 commit해야 한다.** 충돌이 생긴 work.txt를 열어보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

work.txt 파일의 내용이 조금 특별하게 되어 있다. master branch에 있던 내용과 o2 branch에 있던 내용이 한꺼번에 나타났기 때문이다. **‘<<<<<<< HEAD’와 가운데 가로줄(=======)사이의 내용은 master branch에서 수정한 내용**이다. **가로줄(=======)과 ‘>>>>>>> o2’사이의 내용은 o2 branch에서 수정한 내용**이다. **양쪽 branch의 내용을 참고해 직접 내용을 수정**해야 한다.

6. **내용을 원하는 대로 수정했으면 문서에 나타나 있던 ‘<<<<<<< HEAD’, ‘>>>>>>> o2’, 가로줄(=======)은 모두 삭제**한다. **이제 수정한 work.txt를 stage에 올리고 commit(commit message: merge o2 branch)하면 된다.** 이렇게 해서 o2 branch에서 merge한 work.txt의 conflict를 해결하고 commit할 수 있다.

7. git log를 이용해 지금까지 만든 branch와 commit의 관계를 확인해보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

**\* merge가 끝난 branch 삭제하기**

**branch를 병합한 후 더 이상 사용하지 않는 branch는 깃에서 삭제할 수 있다.** **단, 이렇게 branch를 삭제해도 이 branch가 완전히 지워지는 것이 아니라 다시 같은 이름의 branch를 만들면 예전 내용을 다시 볼 수 있다.**

1. **git branch 명령**을 사용하면 현재 저장소에 **master와 o2 branch가 있는 것을 확인**할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

2. **저장소의 기본 branch는 master이므로 branch를 삭제하려면 master branch에서 해야 한다.** branch를 삭제할 때는 **git branch 명령에 -d 옵션**을 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위처럼 **Deleted branch o2 메시지**가 나타나면 **성공적으로 branch를 삭제한 것**이다. 하지만 **삭제한 branch는 같은 이름으로 다시 branch를 만들면 예전에 작업했던 내용이 그대로 나타난다. 즉, branch를 삭제한다는 것은 완전히 저장소에서 없애는 것이 아니라 깃의 흐름 속에서 감추는 것이라고 생각하면 된다.**