

Branch

Branch란?!

- 특정 시점에서 코드의 버전(상태)을 나누어 별도로 관리할 수 있도록 하는 기능
- 기본 브랜치 (Main, Master)에서 나뉘어가지 처럼 새로운 브랜치를 생성하여 기능 추가, 오류 수정 등을 가능하게 함
- 분기된 가지: 특정 시점에서 갈라져서 독립적인 작업 가능

Branch 명령어

- 브랜치 목록 확인: git branch

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git branch
* main
```

- 브랜치 생성: git branch <브랜치명>

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git branch test
```

- 브랜치 이동: git switch <브랜치명>

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git switch test
Switched to branch 'test'

admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (test)
$
```

Branch 명령어

- 브랜치 생성+이동: `git switch -c <브랜치명>`

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (test)
$ git switch -c hotfix
Switched to a new branch 'hotfix'

admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (hotfix)
$ git branch
* hotfix
  main
  test
```

- 브랜치 이름 수정: `git branch -m <기존 브랜치명> <새 브랜치명>`

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (hotfix)
$ git branch -m test dev

admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (hotfix)
$ git branch
  dev
* hotfix
  main
```

- 브랜치 삭제 : `git branch -d <브랜치명>`

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (hotfix)
$ git branch -d dev
Deleted branch dev (was 08e33a0).
```

실습) main branch

1. Main 브랜치에 dept.yaml 파일 생성
2. 아래의 내용을 작성 한 뒤, add

department: ITSoft

professor: Haeri

subject: IT

member:
- Tam

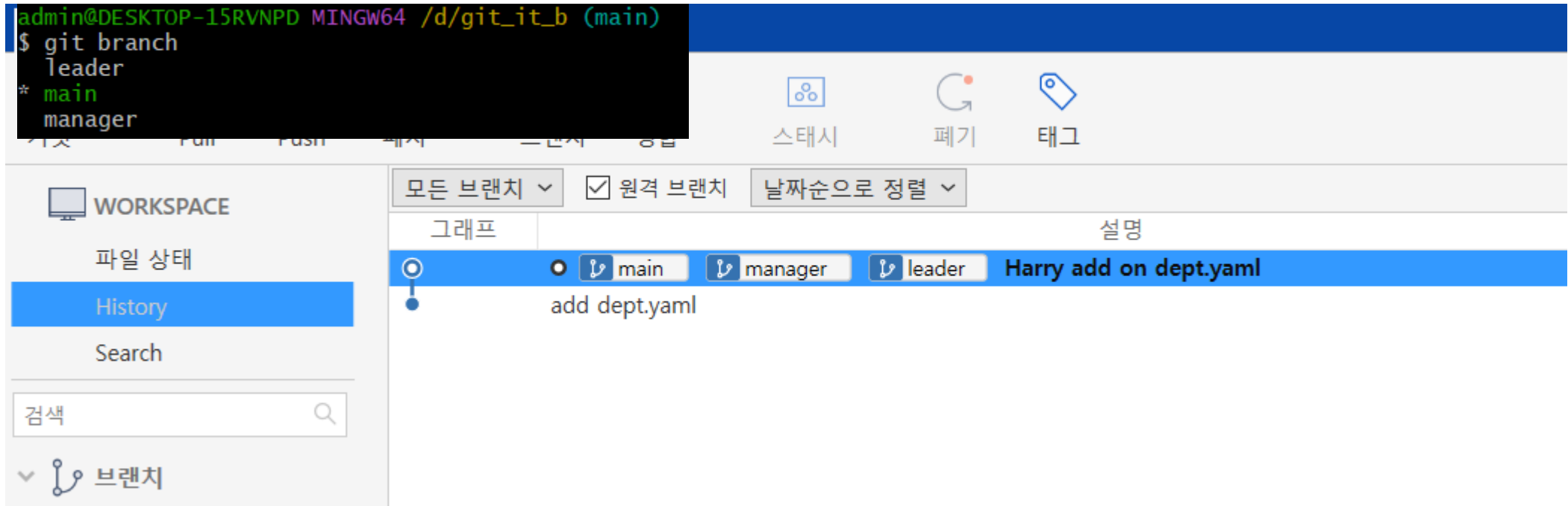
3. commit 하기, commit message : add dept.yaml

실습) main branch

4. member에 Harry 추가 후, add → commit

message : Harry add on dept.yaml

5. branch 생성하기 : leader, manager 브랜치 생성



실습) leader branch

6. leader브랜치로 이동

7. dept.yaml 파일 수정: add → commit -m "add kim for leader on dept.yaml"

```
department: ITSoft
```

```
professor: Haeri
```

```
leader: Kim
```

```
subject: IT
```

```
member:
```

```
- Tam
```

```
- Harry
```

실습) leader branch

8. leader.yaml 파일 생성 후, 아래의 내용 작성

```
leader:  
- kim
```

9. add → commit -m "add leader.yaml"

10. leader.yaml 파일에 Lee추가

: add → commit -m "Lee add on leader.yaml"

11. leader.yaml 파일에 Park추가

: add → commit -m "Park add on leader.yaml"

12. **main 브랜치 이동!** : member에 Sunny 추가

: add → commit -m "add Sunny on dept.yaml"

실습) manager branch

13. manager브랜치로 이동

14. dept.yaml 파일 수정: add → commit -m "choi add for manager on dept.yaml"

```
department: ITSoft
```

```
professor: Haeri
```

```
manager: Choi
```

```
subject: IT
```

```
member:
```

```
- Tam
```

```
- Harry
```

실습) manager branch

8. manager.yaml 파일 생성 후, 아래의 내용 작성

```
manager:  
- Choi
```

9. add → commit -m "add manager.yaml"

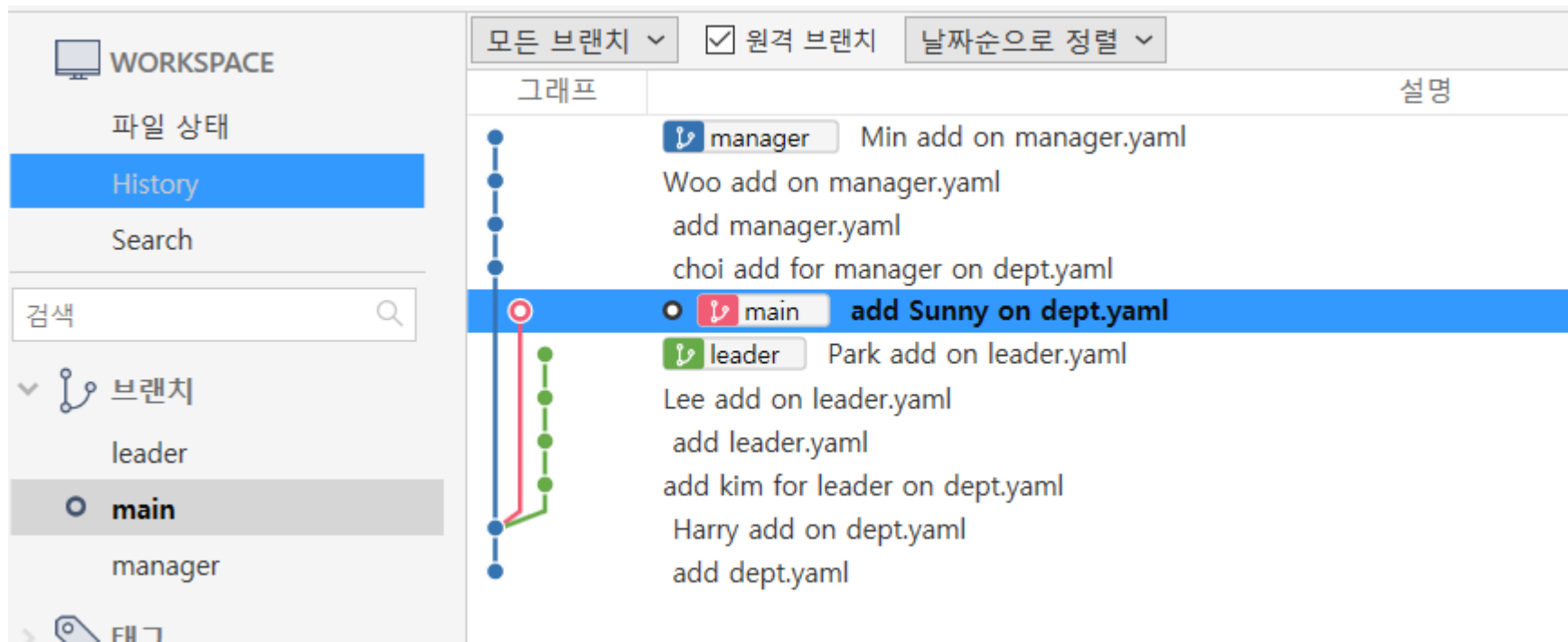
10. manager.yaml 파일에 Woo추가

: add → commit -m "Woo add on manager.yaml"

11. manager.yaml 파일에 Min추가

: add → commit -m "Min add on manager.yaml"

실습) Source Tree 에서 브랜치 확인



실습) cli로 확인

- git log --all --decorate --oneline --graph

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git log --all --decorate --oneline --graph
* 6efdc63 (manager) Min add on manager.yaml
* e0fa29d Woo add on manager.yaml
* dfe009e add manager.yaml
* 0a32436 choi add for manager on dept.yaml
| * 8843459 (HEAD -> main) add Sunny on dept.yaml
|/
| * 826d65e (leader) Park add on leader.yaml
| * 9cc5aa0 Lee add on leader.yaml
| * f89fe4c add leader.yaml
| * 61b6f51 add kim for leader on dept.yaml
|/
* 8bd1b61 Harry add on dept.yaml
* 4882be7 add dept.yaml
```

Branch 병합하기

- leader브랜치를 main브랜치로 merge
 1. main브랜치로 이동
 2. git merge leader

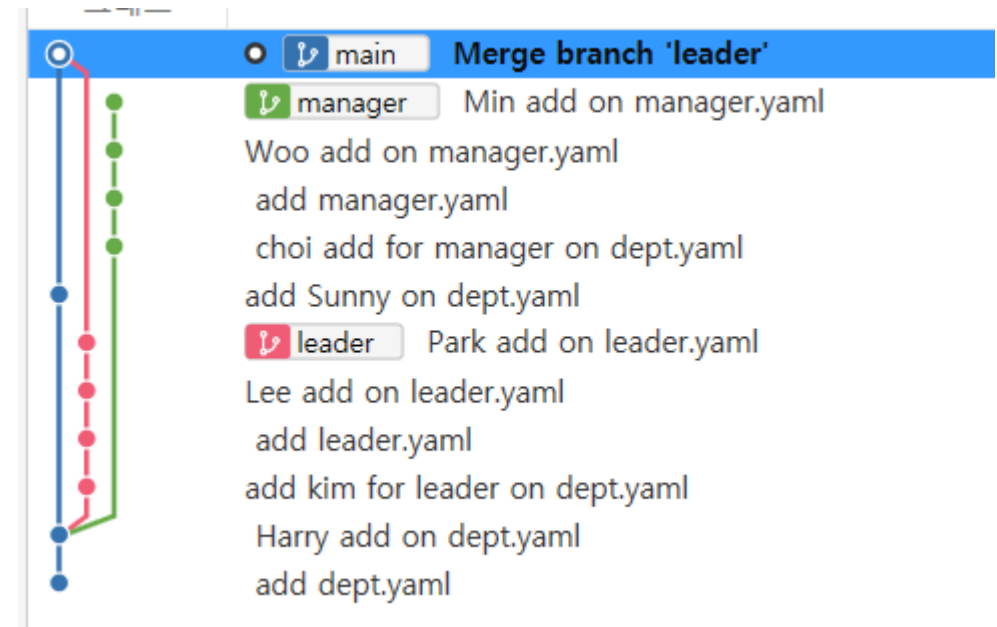
```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git merge leader
Auto-merging dept.yaml
Merge made by the 'ort' strategy.
 dept.yaml | 2 ++
 leader.yaml | 4 ++++
 2 files changed, 6 insertions(+)
 create mode 100644 leader.yaml
```

이름

📁 .git






📄 dept.yaml

📄 leader.yaml



git log로 commit 내역들 확인하기

실습) merge취소하기!

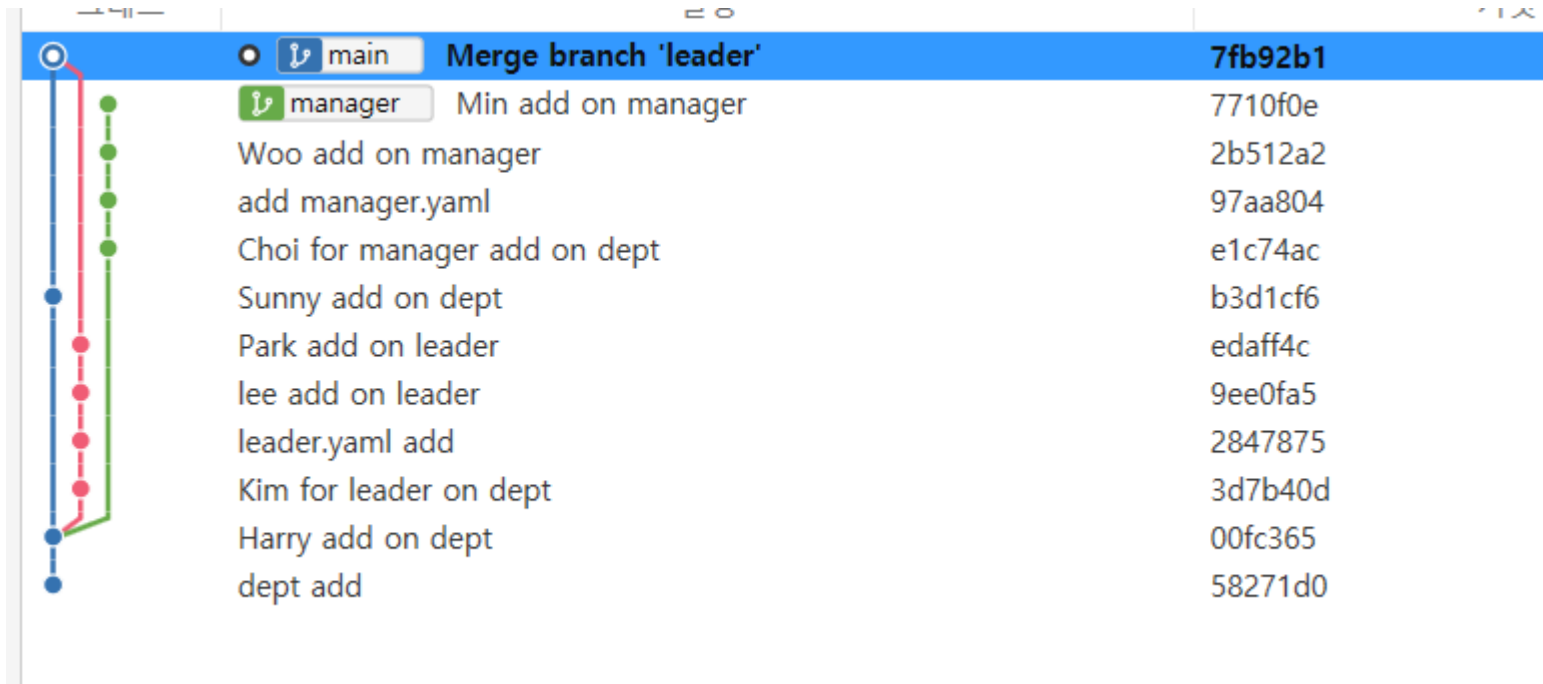
그래프	설명	날짜	작성자	
	 main Merge branch 'leader'	11 11 2024 4:01	anhr <anhr@shingu.ac.kr>	00bf288
	 manager Min add on manager	11 11 2024 4:01	anhr <anhr@shingu.ac.kr>	ab44469
	Woo add on manager	11 11 2024 4:00	anhr <anhr@shingu.ac.kr>	239d2f8
	manager.yaml:	11 11 2024 4:00	anhr <anhr@shingu.ac.kr>	48a2b9b
	Choi for manager on dept	11 11 2024 3:59	anhr <anhr@shingu.ac.kr>	f41ba2b
	 Sunny add on dept	11 11 2024 3:59	anhr <anhr@shingu.ac.kr>	930308f
	 leader Park on leader.yaml	11 11 2024 3:58	anhr <anhr@shingu.ac.kr>	450f634
	Lee on leader.yaml	11 11 2024 3:58	anhr <anhr@shingu.ac.kr>	79a25b9
	add leader.yaml	11 11 2024 3:58	anhr <anhr@shingu.ac.kr>	0e8fdd2
	kim for leader on dept	11 11 2024 3:57	anhr <anhr@shingu.ac.kr>	864797c
	Harry add on dept	11 11 2024 3:56	anhr <anhr@shingu.ac.kr>	a162b7e
	add dept	11 11 2024 3:56	anhr <anhr@shingu.ac.kr>	25cc08b

병합이전 커밋!

```
git reset --hard 930308f43f0fd0002e7f5a56691d50004a50caf4
```

병합된 브랜치 삭제

git branch -d leader



branch 병합하기- rebase

- manager 브랜치를 main브랜치로 rebase
- rebase : 기반을 다시 설정하다
- 즉! manager브랜치를 main브랜치 위로 rebase하면, manager 브랜치 커밋들이 main의 최신 커밋 위에 다시 쌓임

중요! - rebase중 충돌 발생!

```
department: ITSoft
```

```
professor: Haeri
```

현재 변경 사항 수락 | 수신 변경 사항 수락 | 두 변경 사항 모두 수락 | 변경 사항 비교

```
<<<<<<< HEAD (현재 변경 사항)
```

```
leader: Kim
```

```
=====
```

```
manager: Choi
```

```
>>>>>>> e1c74ac (Choi for manager add on dept) (수신 변경 사항)
```

```
subject: IT
```


```
member:
```

- Tam
- Harry

git rebase --abort 를 사용하여 rebase 전으로 복구!

실습)branch 병합하기- rebase

- main 브랜치로 이동
- leader: Kim부분 → manager: Choi로 변경후 add and commit

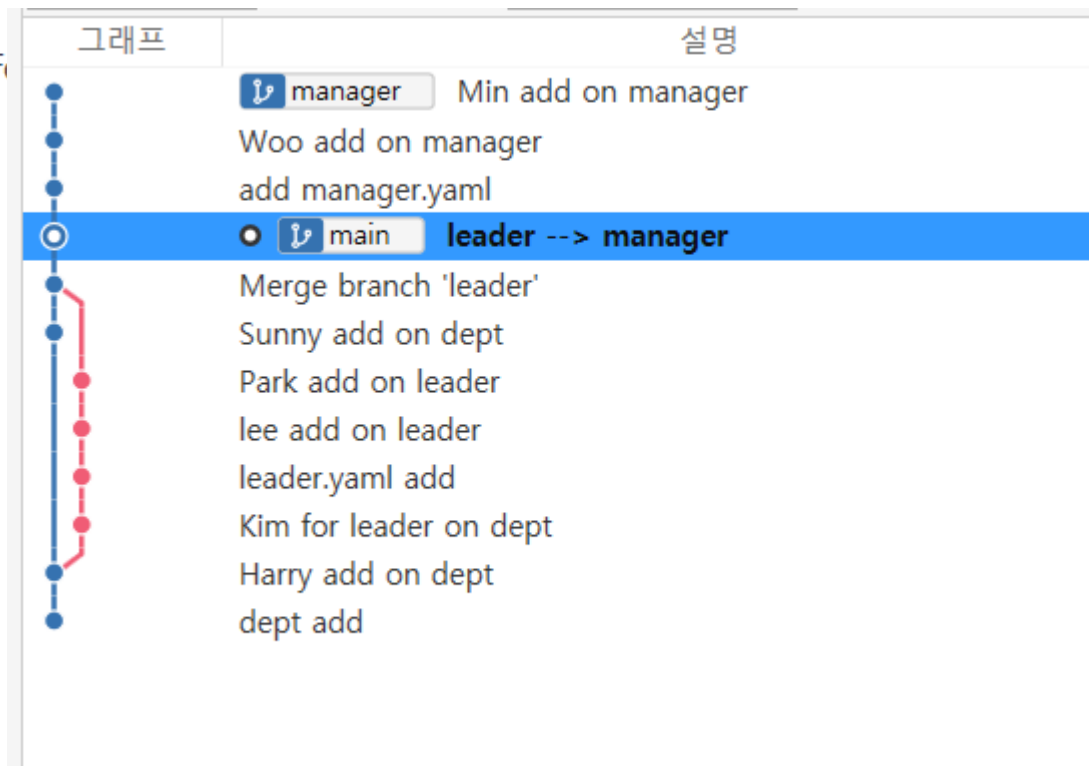
그래프	설명	커밋
	main Tom add on dept	9b87284
	manager Min add on dept	ae59ae6
	Woo add on dept	275ed28
	manager.yaml add	996b4a1
	choi for manager add on detp	5df9eee
	Sunny add on dept	b8a1ae2
	leader Park add on leader	9de9149
	Lee add on leader	772ae77
	leader.yaml	e476635
	Harry add on dept	fc457ff
	add dept	8987174

실습)branch 병합하기- rebase

manager 브랜치로 이동 → git rebase main

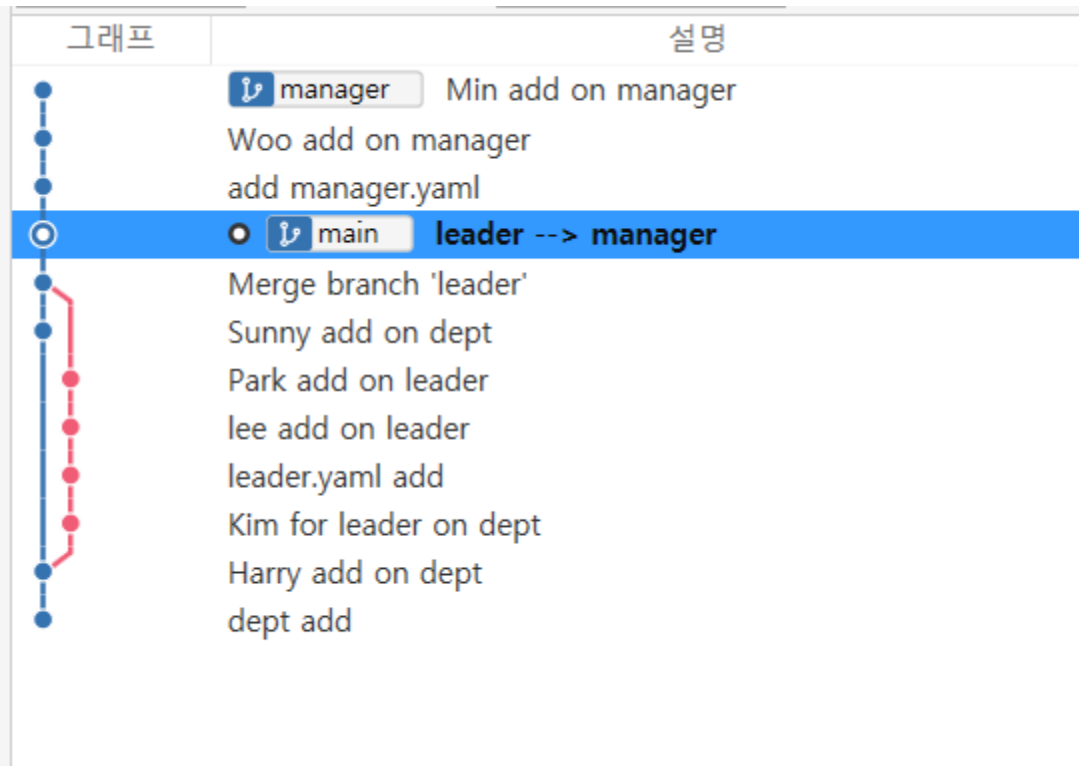
```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_a (main)
• $ git switch manager
  Switched to branch 'manager'

admin@DESKTOP-15RVNPD MINGW64 /d/git_it_a (manager)
• $ git rebase main
dropping e1c74acd9564c3035aa58b865aee1fa29a5d08cf Choi f
Successfully rebased and updated refs/heads/manager.
```



실습)branch 병합하기- rebase

소스트리 확인!

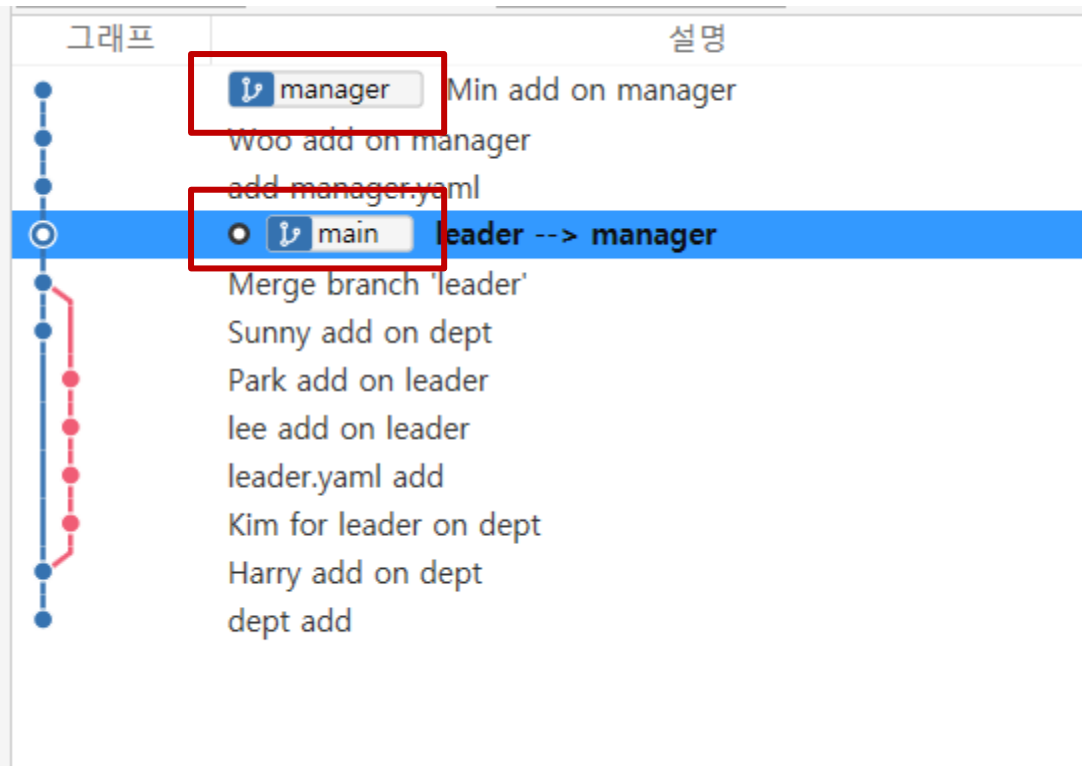


main브랜치가 뒤쳐져 있음!!!!

manager.yaml 파일이 어디있을까?!!

실습)branch 병합하기- rebase

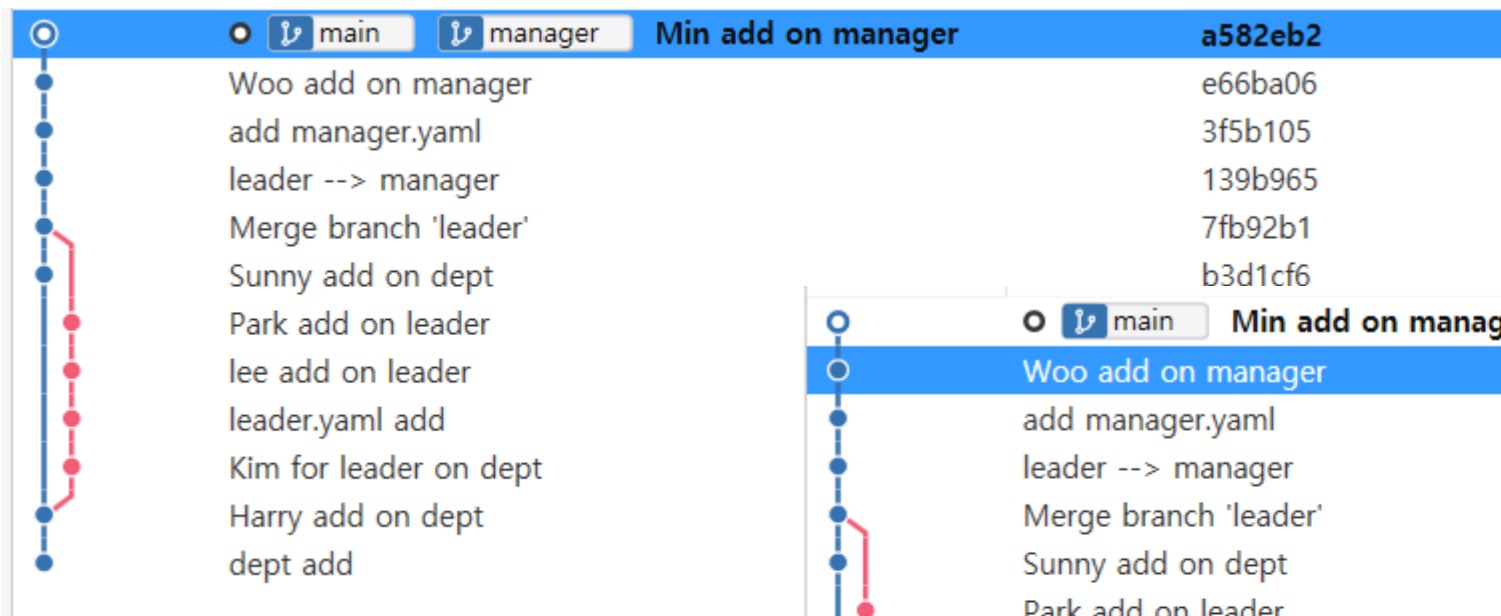
main브랜치 이동! main브랜치를 manager시점으로 이동하여 merge



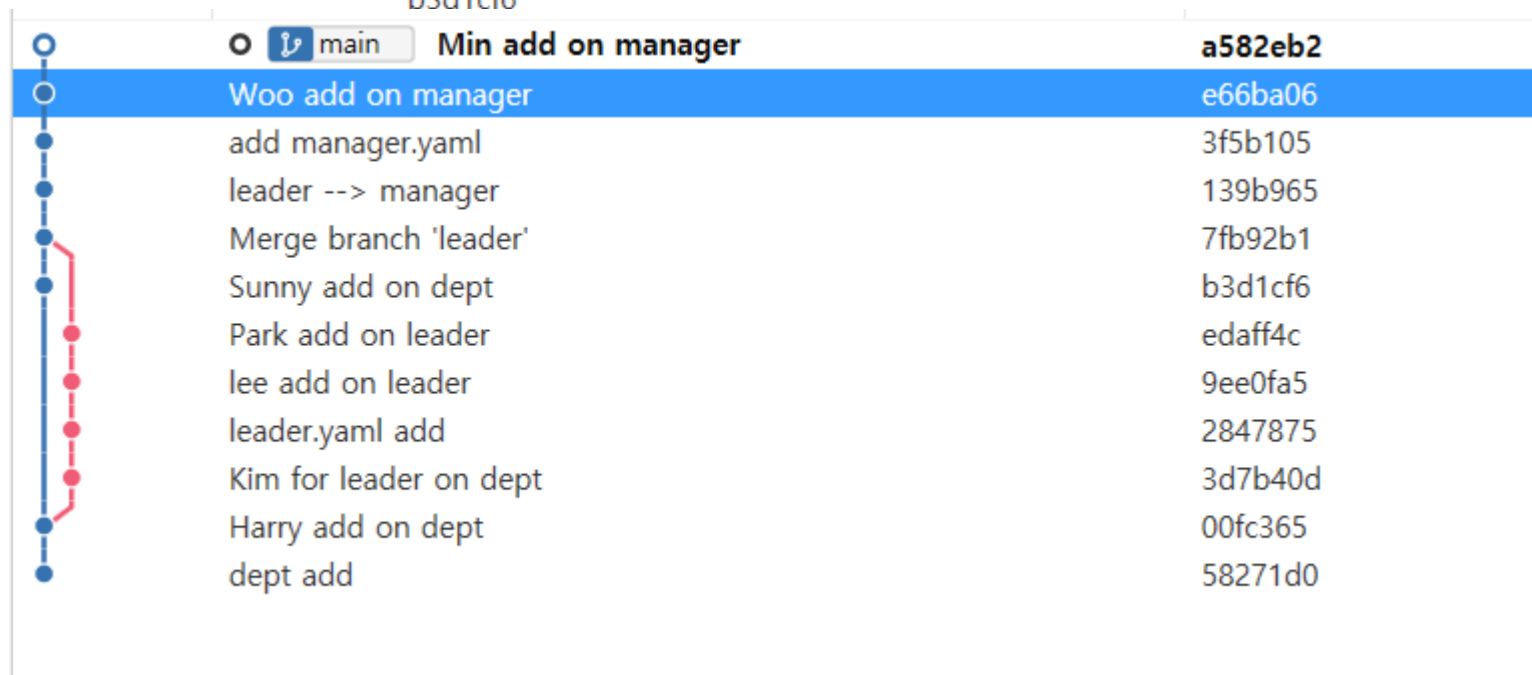
git merge manager

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_a (main)
$ git merge manager
Updating 139b965..a582eb2
Fast-forward
 manager.yaml | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 manager.yaml
```

실습)branch 병합하기- rebase



manager 브랜치 삭제!



Branch 병합 중, 충돌! - merge

1. main Branch : confict-1.txt add & commit

dept: ITsoftware
member:
- Lee



dept: ITsoftware
member:
- Lee
- Park

2. dev Branch 생성

3. dev Branch에서 Woo 추가 후, add & commit

4. dev Branch에서 Kim 추가 후, add & commit

5. main브랜치 이동 후, 파일 확인

6. dev 브랜치 병합 : git merge dev

7. 확인! → dev 브랜치 삭제

Branch 병합 중, 충돌! - merge

1. test브랜치 생성 → test브랜치 이동
2. test branch : Min 추가 후 add & commit
3. main branch : Son 추가 후 add & commit
4. main brabch : test브랜치와 merge 시도!

```
Administrator@DESKTOP-T7Q80LC MINGW64 /e/python/git_repo (main)
$ git merge test
Auto-merging conflict-1.txt
CONFLICT (content): Merge conflict in conflict-1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
- Son
=====
- Min
>>>>>> test (Incoming Change)
|
```


Branch 병합 중, 충돌! - merge

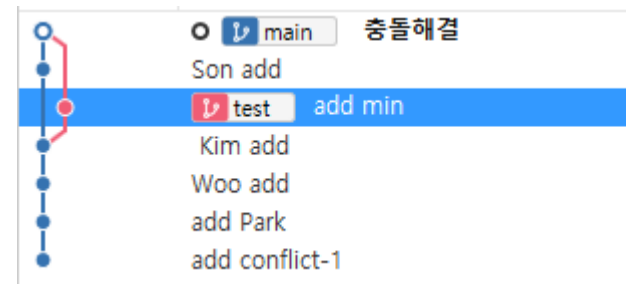
- merge 취소 : git merge --abort

```
Administrator@DESKTOP-T7Q80LC MINGW64 /e/python/git_repo (main|MERGING)
$ git merge --abort
```

- merge 해결 : 원하는 방향으로 수정 후, add and commit

```
Administrator@DESKTOP-T7Q80LC MINGW64 /e/python/git_repo (main|MERGING)
$ git add .

Administrator@DESKTOP-T7Q80LC MINGW64 /e/python/git_repo (main|MERGING)
$ git commit -m "충돌해결"
[main de07507] 충돌해결
```



Branch 병합 중, 충돌! - rebase

1. test2 브랜치 생성
2. test2 브랜치 : member에 Choi 추가 후 add and commit
3. main 브랜치 이동 후, membe에 Ha 추가 후 add and commit
4. test2 브랜치 이동 : git rebase main → 충돌 발생!

```
Administrator@DESKTOP-T7Q80LC MINGW64 /e/python/git_repo (main)
$ git switch test2
Switched to branch 'test2'

Administrator@DESKTOP-T7Q80LC MINGW64 /e/python/git_repo (test2)
$ git rebase main
Auto-merging conflict-1.txt
CONFLICT (content): Merge conflict in conflict-1.txt
error: could not apply 3039755... Choi add
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
hint: Disable this message with "git config advice.mergeConflict false"
Could not apply 3039755... Choi add
```

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
- Ha
=====
- Choi
>>>>>>> 3039755 (Choi add) (Incoming Change)
```

Branch 병합 중, 충돌! - rebase

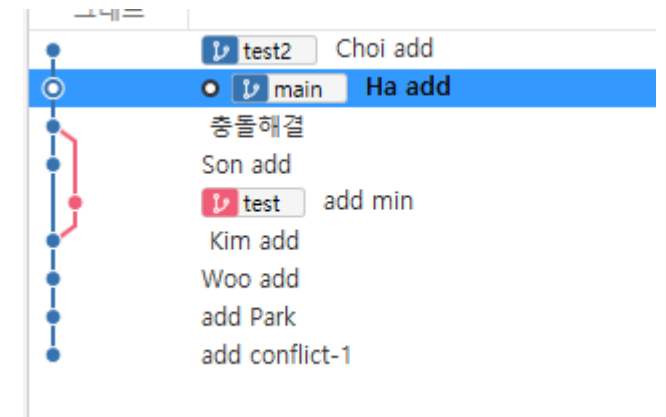
5. 충돌 해결 후 add

```
conflict-1.txt
1  dept: ITsoftware
2  member:
3    - Lee
4    - Park
5    - Woo
6    - kim
7    - Son
8    - Min
9    - Ha
10   - Choi
```

```
Administrator@DESKTOP-T7Q80LC MINGW64
$ git add conflict-1.txt
```

6. 계속 rebase진행: git rebase --continue

```
Administrator@DESKTOP-T7Q80LC MINGW64 /e/python/git_repo (test2|REBASE 1/1)
$ git rebase --continue
[detached HEAD 2fef406] Choi add
1 file changed, 3 insertions(+), 1 deletion(-)
Successfully rebased and updated refs/heads/test2.
```



Branch 병합 중, 충돌! - rebase

- rebase취소 : `git rebase --abort`