

Git 심화 + Git 협업

#번째 세션

NEXT X LIKELION 허영범

Introduction

떡밥 회수의 시간이 왔다.

Git 기능 이게 끝인가요?

그래도 git의 기초 기능을 익혔습니다.

1. 그래서 이전 버전으로는 어떻게 돌아가나요?
2. 이걸로 협업이 진짜 가능한가요?
3. 안에서 여러 개의 코드 변화 시도를 해보고 싶은데 어떡 하나요?

사실상 떡밥 회수 못했..

Detail Curriculum

Git and Github

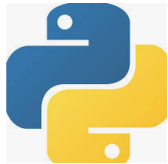
1. Git 로컬에서 기초 버전관리
2. Github Repository에 푸시하기
3. Git으로 버전 세부적으로 관리 해보기
4. Github 및 Git을 통한 협업



5월 협업 세션에서 진행할 예정입니다!

Entire Curriculum

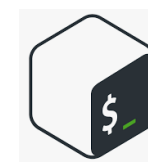
Programming Language



Framework



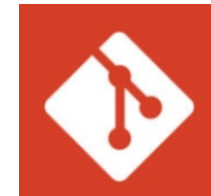
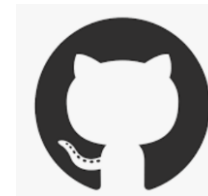
Tools



Detail Curriculum

Git and Github

1. Git 로컬에서 기초 버전관리
2. Github Repository에 푸시하기
3. Git으로 버전 세부적으로 관리 해보기
4. Github 및 Git을 통한 협업

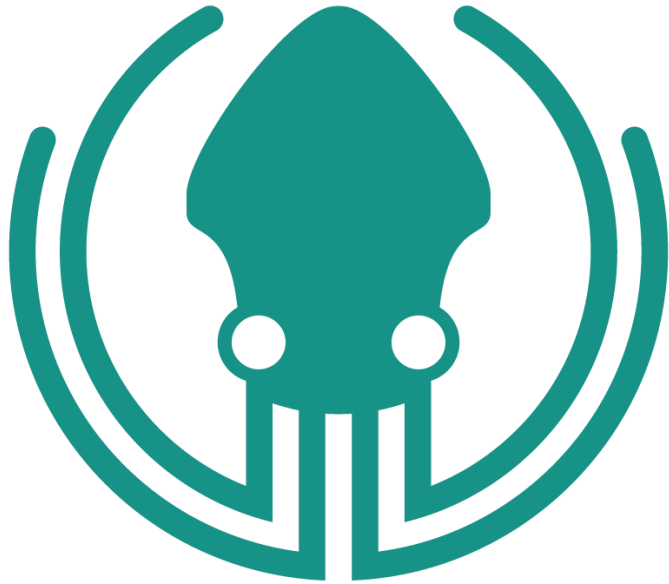


| Index

1. 간단한 복습 (Git이란? Git 명령어 등)
2. 저장소에 있는 코드 가져오기
3. git branch
4. git 이전 커밋으로 돌아가기
5. git pull request && review 남겨 보기
6. git merge & fetch & rebase
7. conflict

CLI가 익숙하지 않으면 추천하는 프로그램

오늘 강의가 어려웠다면, 이 프로그램들을 써보면서 감을 잡는 것도 좋아요!



GitKraken



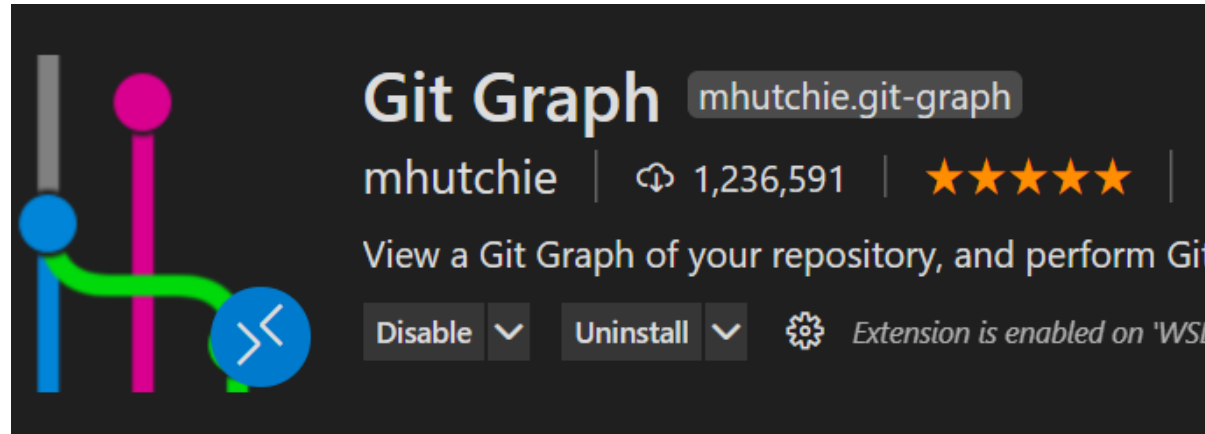
Github Desktop

Chapter 1 & 2

git 단순 복습 + 저장소 연결 및 코드 가져오기

(이해를 돕기 위한) git graph 설치

Git을 활용하고 협업하는데 있어서 필수는 아니지만, 이해하기 쉽도록 이 vscode extension 도구를 사용 할게요!



| Git?



분산형 버전 관리 시스템

- 버전 관리
- 협업 지원

| Git 명령어들



- git init
- git add <추가할 파일>
- git commit -m "<커밋 메시지>"
- git remote add <저장소 별칭> <저장소 주소>
ex) git push remote add origin "멋사저장소"
- git push <저장소 별칭> <브랜치 이름>
ex) git push origin master



Git F&Q

git init을 여러 군데 했을 때...



여러 개의 Git?

- Git으로 버전 관리를 할 때에는 하나의 git 설정 파일만 존재를 해야 합니다. (git init . 은 한 번만 하라는 의미)
- 여러 개의 git을 실수로 설정하였다면, 최상단에 있는 git을 제외하고 하위 폴더로 이동하여
"rm -rf ./git" 이라는 명령어로 git 설정 파일을 삭제해줍니다.



Git F&Q

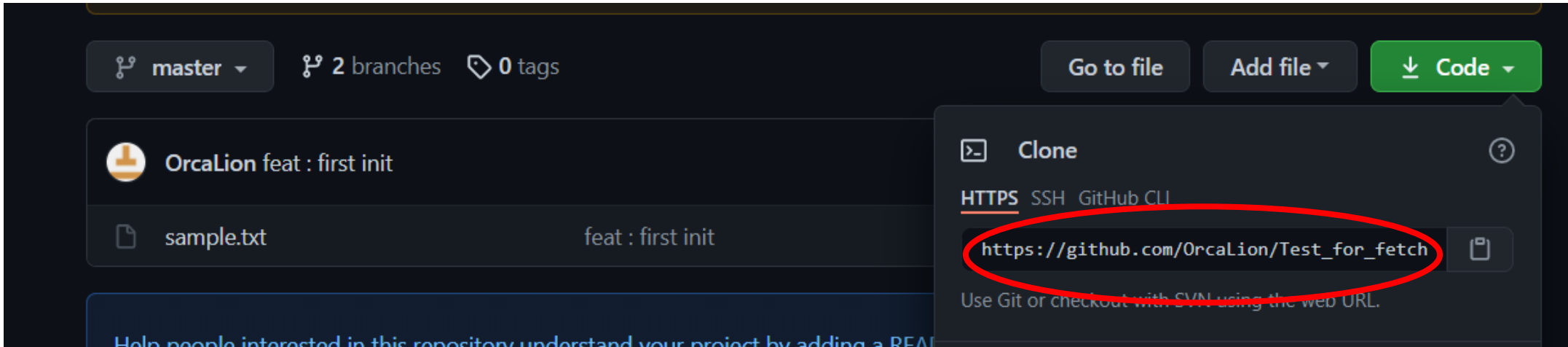
git fetch, push, pull, merge 등 명령어 전에



변경 사항이 있으면 커밋!

그 변경 사항이 스테이징만 되어있지만, 추가 하고 싶지 않으면 삭제 후 git 명령어를 실행해주세요.

기존의 저장소에 있는 코드 가져 오기



In terminal,

git clone "저장소 주소 "

```
→ Session_Git_Collaborate git clone https://github.com/OrcaLion/Test_for_fetch.git
Cloning into 'Test_for_fetch'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 1), reused 8 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 673 bytes | 168.00 KiB/s, done.
→ Session_Git_Collaborate s
zsh: command not found: s
→ Session_Git_Collaborate ls
Test_for_fetch
```

기존의 저장소에 있는 코드 가져 오기

오늘은 단순히 복사하는 것이 아닌
내 코드를 합쳐 볼 꺼라 한 단계 더!

In terminal,

git clone "저장소 주소 "

```
→ Session_Git_Collaborate git clone https://github.com/OrcaLion/Test_for_fetch.git
Cloning into 'Test_for_fetch'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 1), reused 8 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 673 bytes | 168.00 KiB/s, done.
→ Session_Git_Collaborate s
zsh: command not found: s
→ Session_Git_Collaborate ls
Test_for_fetch
```

기존의 저장소 포크해서 가져오기



범진님 레포에서 가져온 예시



본인 id/ 포크한 저장소 이름
Forked from 원주인 id / 포크한 저장소이름

In terminal,

`git clone "포크한 저장소 주소 "`

`git clone -b "포크할 브랜치 " "포크한 저장소 주소 "`

(특정 브랜치만 가져올 때)

기존의 저장소 포크해서 가져오기

OrcaLion / LikeLion_Git_Collaborate

Unwatch 1

Star

Fork 0

회장님 레포에서 가져온 예시

goldlamp95 / pyenv
forked from pyenv/pyenv

본인 id/ 포크한 저장소 이름
Forked from 원주인 id / 포크한 저장소이름

In terminal,

git clone "포크한 저장소 주소 "

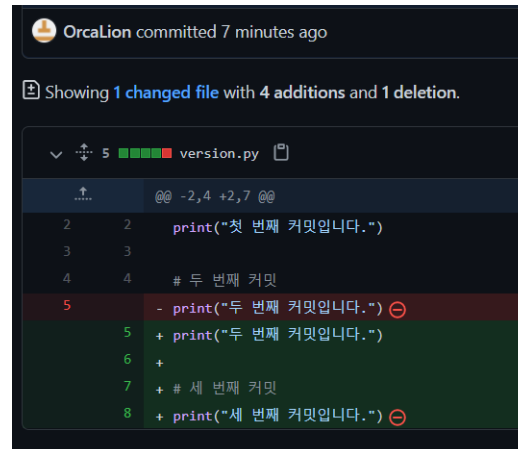
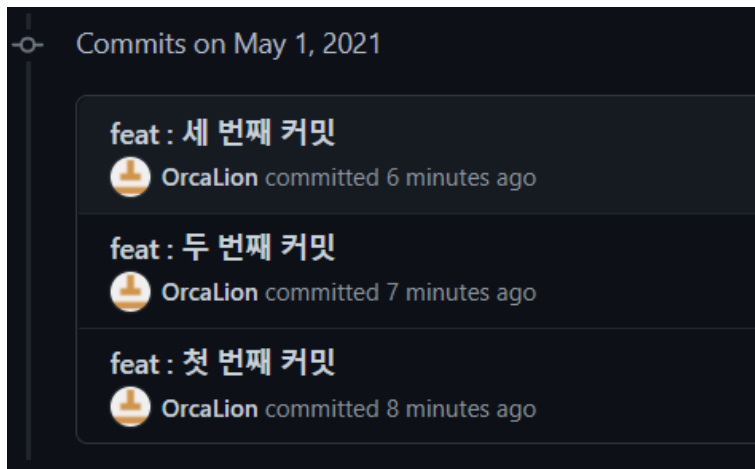
✓ git clone -b "포크할 브랜치 " "포크한 저장소 주소 "

(특정 브랜치만 가져올 때)

Ex) git clone -b "dev" https://github.com/{여러분 아이디}/LikeLion_Git_Collaborate.git

(지금 너무 브랜치가 많아요!)

Github 사이트 살펴 보기



각 커밋 별로 변경 사항
확인 가능

git head?

현재 가르키고 있는 커밋, branch

git log

```
commit d856b286fa6c3aec9ed64a15ac0c802ede8e7a09 (HEAD -> dev,  
n/OrcaLion, origin/HEAD)  
Author: OrcaLion <hybeom@likelion.org>  
Date: Sat May 1 16:29:00 2021 +0900  
  
    feat : 세 번째 커밋
```

현재 어디까지 코드가 진행되어 왔다~ 라고 파악하시면 됩니다.

그런데 Branch는 무엇이죠?

Chapter 3

git branch란?



git branch 개요

Branch란?

독립적인 작업을 위해서 다른 줄기를 파는 것

Name	Date modified	Type	Size
과제 공통 회의분	2021-05-01 오후 4:56	Text Document	0 KB
과제 공통 회의분 허영범_1번 문항	2021-05-01 오후 4:56	Text Document	0 KB
과제 공통 회의분 최주원_4번 문항	2021-05-01 오후 4:56	Text Document	0 KB
과제 공통 회의분 이소영_2번 문항	2021-05-01 오후 4:56	Text Document	0 KB
과제 공통 회의분 김동현_3번 문항	2021-05-01 오후 4:57	Text Document	0 KB
과제 공통 회의분 통합	2021-05-01 오후 4:57	Text Document	0 KB

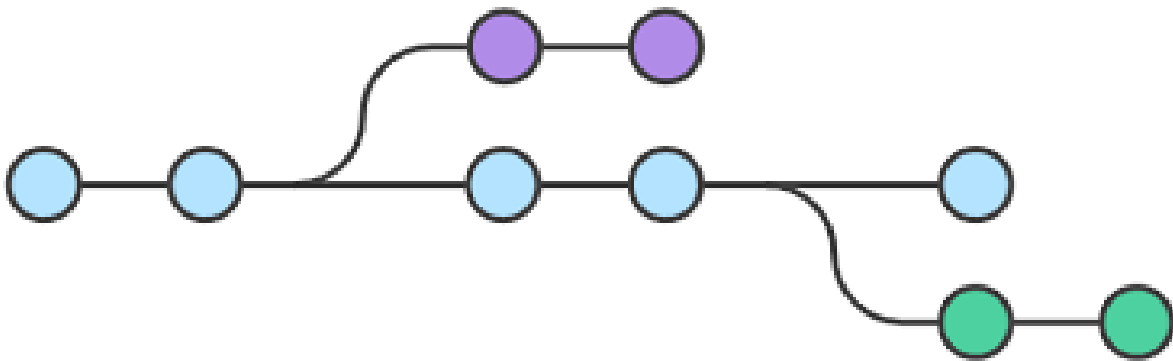
팀플 할 때, 처음 회의 때 공통 파일 만들고 -> 각자 문항 답변해오고 -> 마지막에 합치는

~~이번 세션 끝나면 팀플에 git을 사용할 수 있습니다.~~

git branch 개요

Branch란?

독립적인 작업을 위해서 다른 줄기를 파는 것



ex) Orca/purple_function

ex) main

ex) Orca/green_function

출처 : [Git Checkout](#) | [Atlassian Git Tutorial](#)

git branch 개요

Branch 생성하기

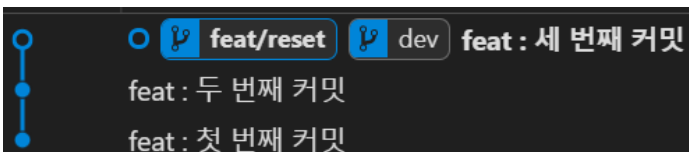
`git checkout -b "<브랜치 네임>"`

브랜치 네임 컨벤션

- 새로운 기능을 추가할 브랜치 : `feat/<추가할 기능 이름>`
`feat/calendar`

조금 있다 리셋을 공부해 볼테니까 `feat/reset`으로 통일 해보도록 하겠습니다.

```
→ LikeLion_Git_Collaborate git:(dev) git checkout -b "feat/reset"  
Switched to a new branch 'feat/reset'  
→ LikeLion_Git_Collaborate git:(feat/reset) _
```






The diagram shows a vertical line representing the commit history. At the top, a blue circle represents the current branch, `feat/reset`. Below it, a blue circle represents the `dev` branch. The text `feat : 세 번째 커밋` is next to the `dev` branch. Below the `dev` branch, the text `feat : 두 번째 커밋` is shown. At the bottom, the text `feat : 첫 번째 커밋` is shown. This indicates that the `feat/reset` branch is a new branch created from the `dev` branch, and the `dev` branch has three commits, while the `feat/reset` branch has two commits (the first and second commits of the `dev` branch).

git graph에서도
추가된 것 확인

git branch 개요







파일 추가 후 새로운 커밋 시 트리 확인!

Graph	Description
	 feat/reset feat : addBranch
	 dev feat : 세 번째 커밋
	feat : 두 번째 커밋
	feat : 첫 번째 커밋

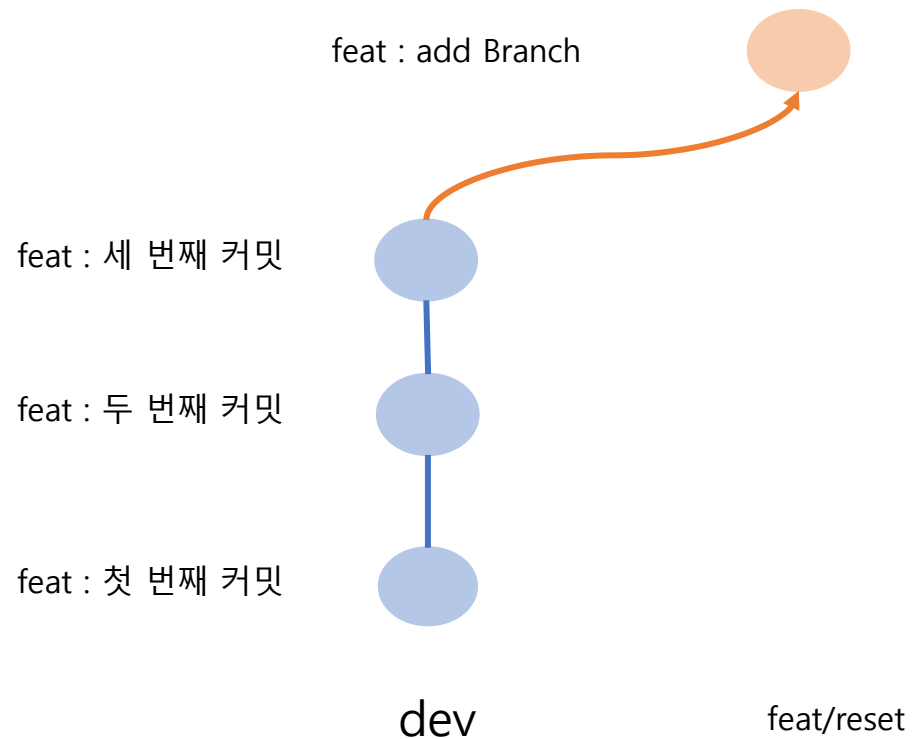
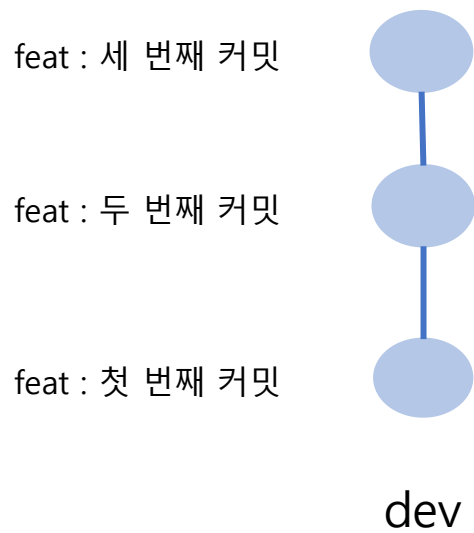
설명 : dev branch는 “feat : 세번째 커밋” 에 머물러 있는데,
새 branch인 feat/reset은 다음 커밋으로 이동해있다.

Dev는 종합본

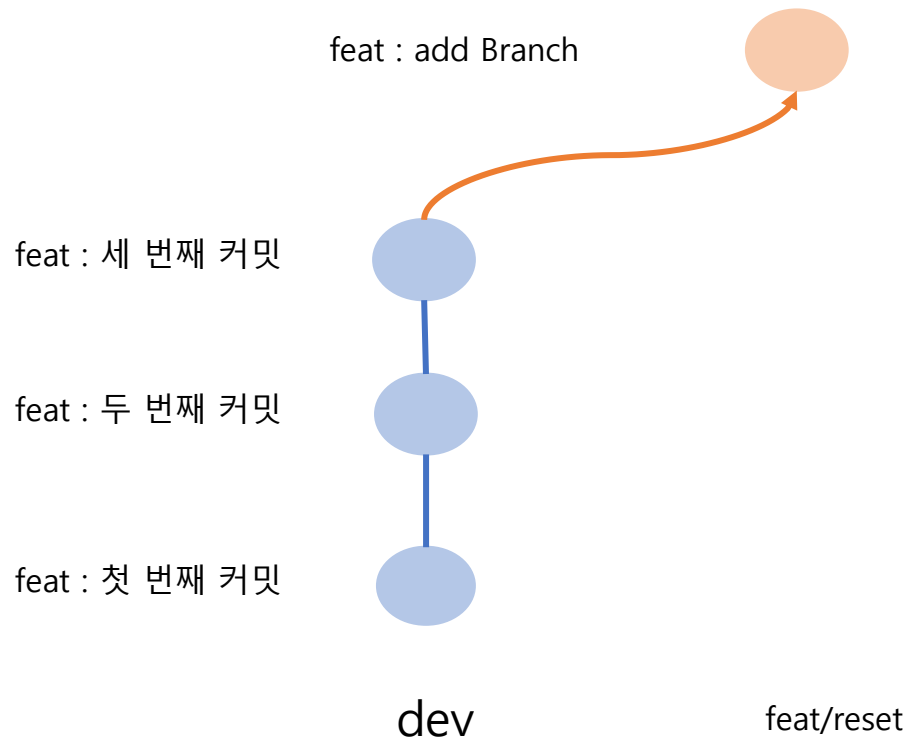
feat/reset 여러분의 각자의 파일이라 생각하시면 편합니다.

 과제 공통 회의분
 과제 공통 회의분 허영범_1번 문항
 과제 공통 회의분 최주원_4번 문항
 과제 공통 회의분 이소영_2번 문항
 과제 공통 회의분 김동현_3번 문항
 과제 공통 회의본 통합

그림으로 보는 현재상황



3분 쉬는 시간

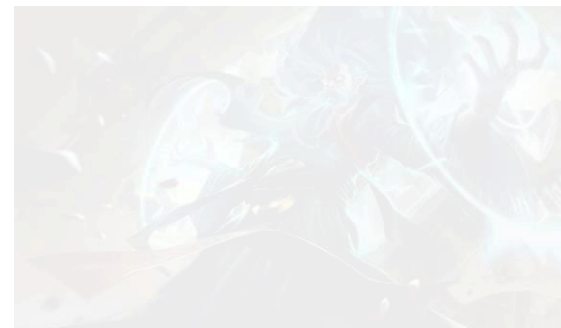


새로 판 브랜치에
2~3 개의 커밋을 더 올려 보세요!

자기 소개, 이번 학기에 듣는 강의 등
모두 좋습니다!

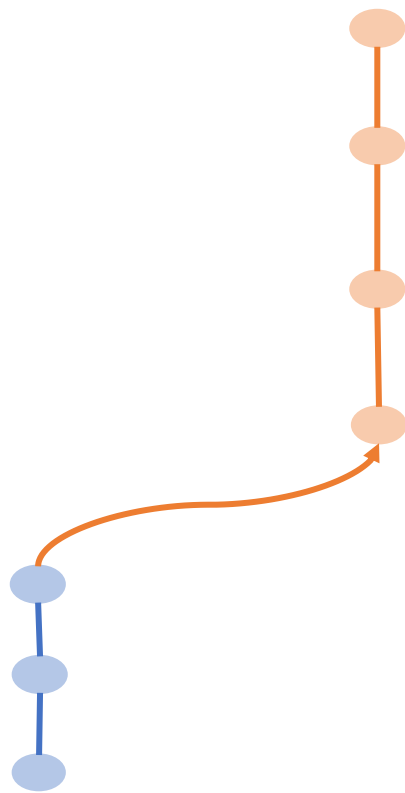
Chapter 4

commit 되돌리기



실수를 했다....

이전 커밋으로 돌리고 싶어요



이쯤 와있는데

여기 커밋 된 상태로 돌리고 싶다

커밋을 되돌리는 방법

방법이 여러 가지입니다!

reset

(타임머신)

해당 커밋으로 돌아 가는 것

Option :

Soft : 커밋하기 직전으로 돌리기

Mixed : 변경된 내용은 남아있지만, 다시 스테이지에 올려야 함

hard : 그냥 싹 다 초기화

revert

(기억 삭제)

해당 커밋을 삭제하는 것

삭제한 기록이 남는다.

커밋을 되돌리는 방법

방법이 여러 가지입니다!

 **reset**

(타임머신)

해당 커밋으로 돌아 가는 것

Option :

Soft : 커밋하기 직전으로 돌리기

Mixed : 변경된 내용은 남아있지만, 다시 스테이지에 올려야 함

hard : 그냥 싹 다 초기화

revert

(기억 삭제)

해당 커밋을 삭제하는 것

삭제한 기록이 남는다.

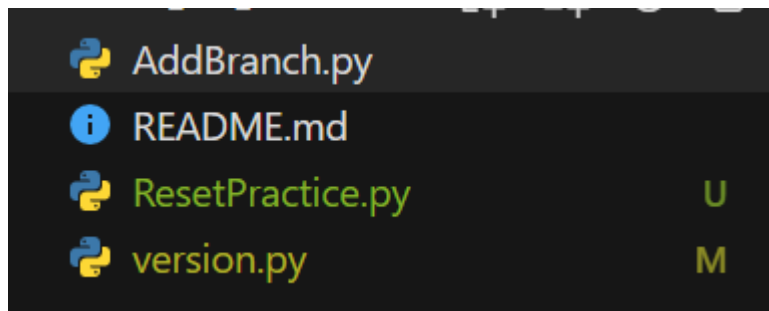
되돌리기 끝!

어때요, 참 쉽죠?



git 되돌리기 이해를 위한 용어 & 상태

이해하기 위한 첫 단추



A : Added (깃 스테이징에 새로 올라간 파일)

U : Untracked (깃 스테이징에 올라가있지 않은 파일들, 지난 커밋 이후 새로 만든 파일 들에서 주로 나타남)

M : Modified (지난 커밋 이후에 수정된 파일들)

D : Deleted (지난 커밋 이후에 삭제된 파일들)

Reset

이해하기 위한 첫 단추

단계	여러분들의 작업	reset option
1. 기존 커밋	레포 불러오기 or 커밋 직후	soft
2. 내용 변경	코드 변경 or 파일 추가	Mixed
3. 변경된 내용 스테이징	git add .	Hard
4. 커밋!	git commit -m "<커밋 메시지>"	

Reset

초기 상태

Staging

Staging Area

README.md
version.py

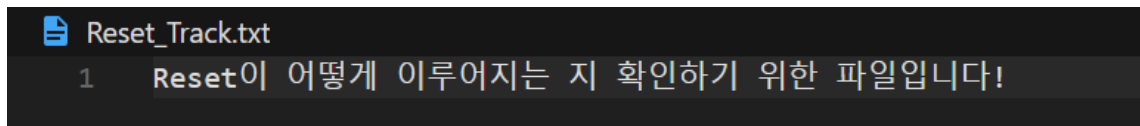
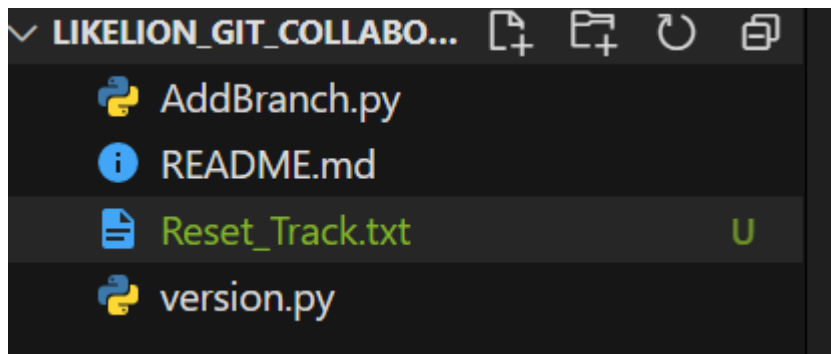
Local directory

Commit

README.md
version.py

Reset

실습을 위하여



```
→ LikeLion_Git_Collaborate git:(feat/reset) ; git add .  
→ LikeLion_Git_Collaborate git:(feat/reset) ; git commit -m "feat :study reset"  
[feat/reset 157e0bc] feat :study reset  
1 file changed, 1 insertion(+)  
create mode 100644 Reset_Track.txt
```

```
git  
commit 157e0bc16a440e2258d257fd0d60767a04cf2928 (HEAD -> feat/reset)  
Author: hoon9729 <h9729@gmail.com>
```

```
feat :study reset
```

이런 내용들로 커밋을 넣겠습니다!

Reset

git 현상태

Staging

Staging Area

README.md
version.py
Reset_Track.txt

Local directory

Commit

README.md
version.py
Reset_Track.txt

Commit -> 해당 커밋 해시값 가져오기

실습!

```
commit 99c2afa76fa4bdfc9f015ba94d1fd27e3439da2a
Author:
Date:
    feat : addBranch
```

commit 뒤의 해시값을 이정도 굵으면 됩니다!

Reset -soft

이해하기 위한 첫 단추

단계

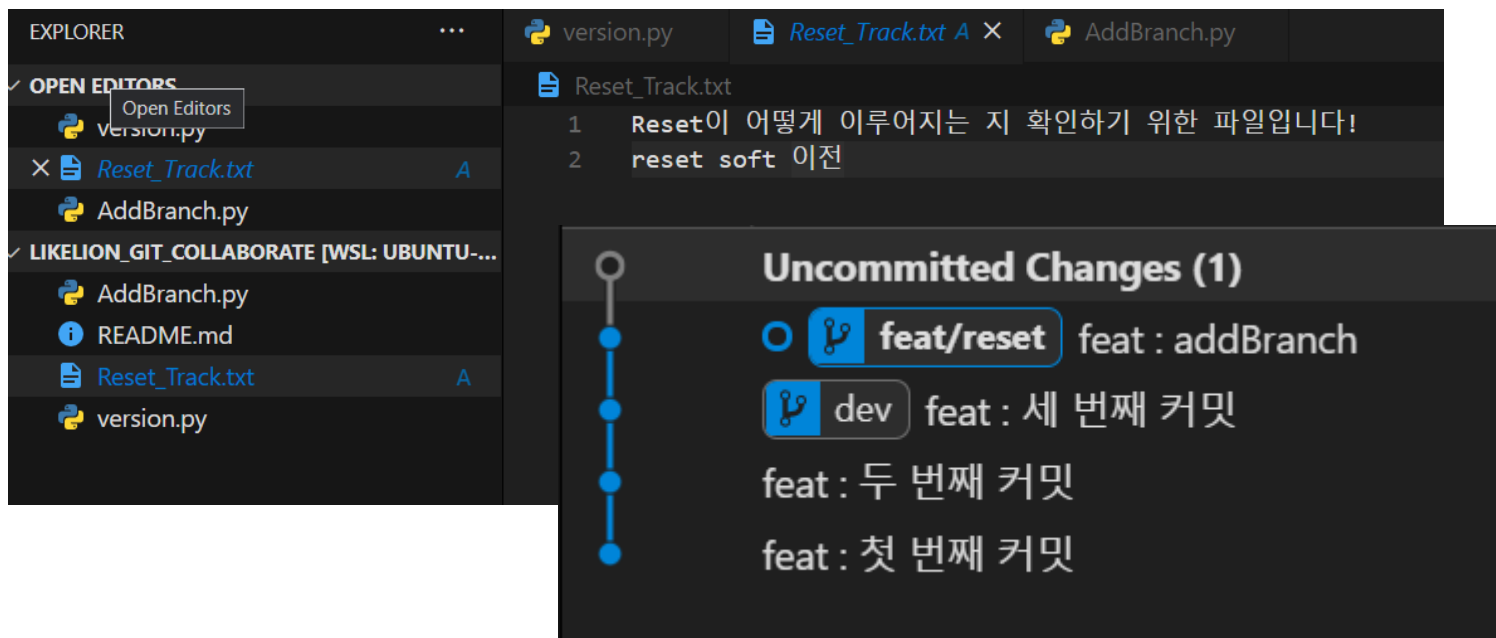
기존 커밋

내용 변경

✓ 변경된 내용 스테이징

커밋!

```
LikeLion_Git_Collaborate git:(feat/reset) git reset --soft 99c2afa76fa4
```



수정된 내용들은 남아있고, 스테이징에는 올라가 있지만, git log 자체는 모두 삭제 됩니다!

Reset

git 현상태

Staging

Commit

Staging Area

README.md
version.py
Reset_Track.txt

Local directory

README.md
version.py
Reset_Track.txt

수정된 내용들은 남아있고, 스테이징에는 올라가 있지만, git log 자체는 모두 삭제 됩니다!

Reset - mixed

이해하기 위한 첫 단추

단계

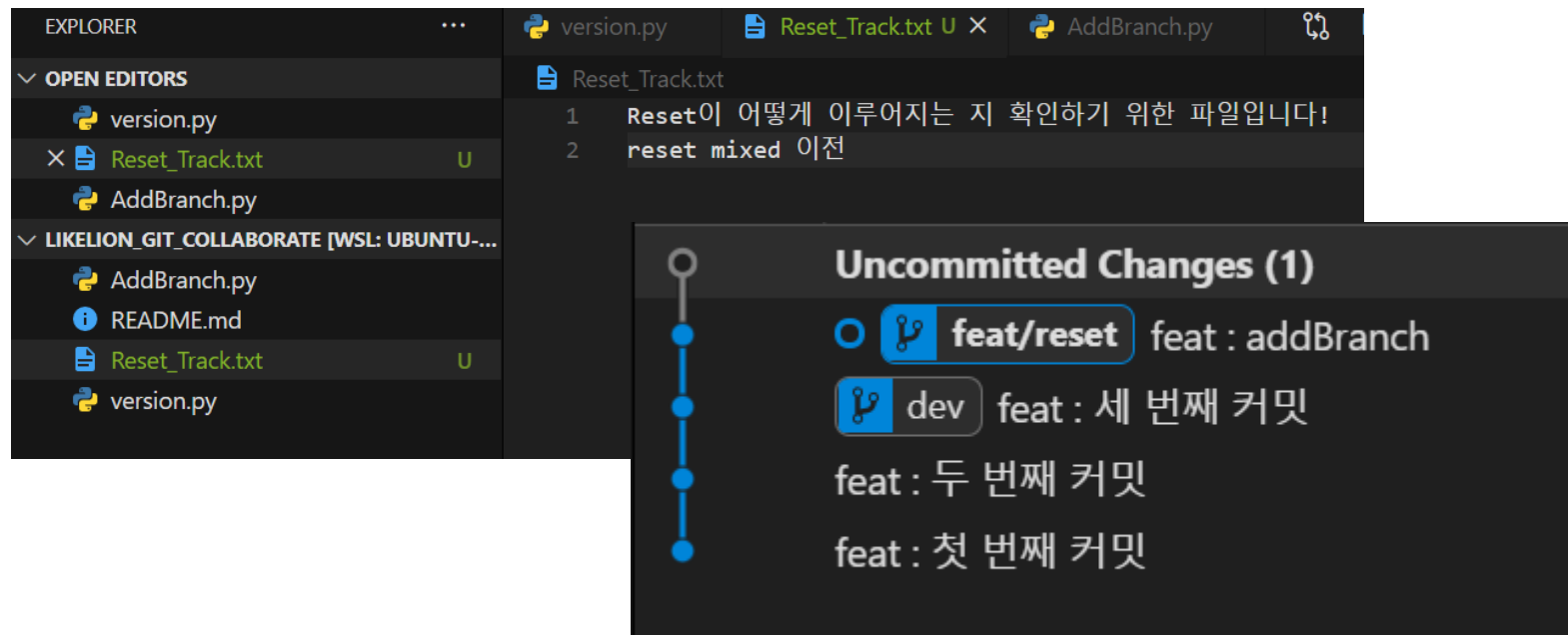
기존 커밋

✓ 내용 변경

변경된 내용 스테이징

커밋!

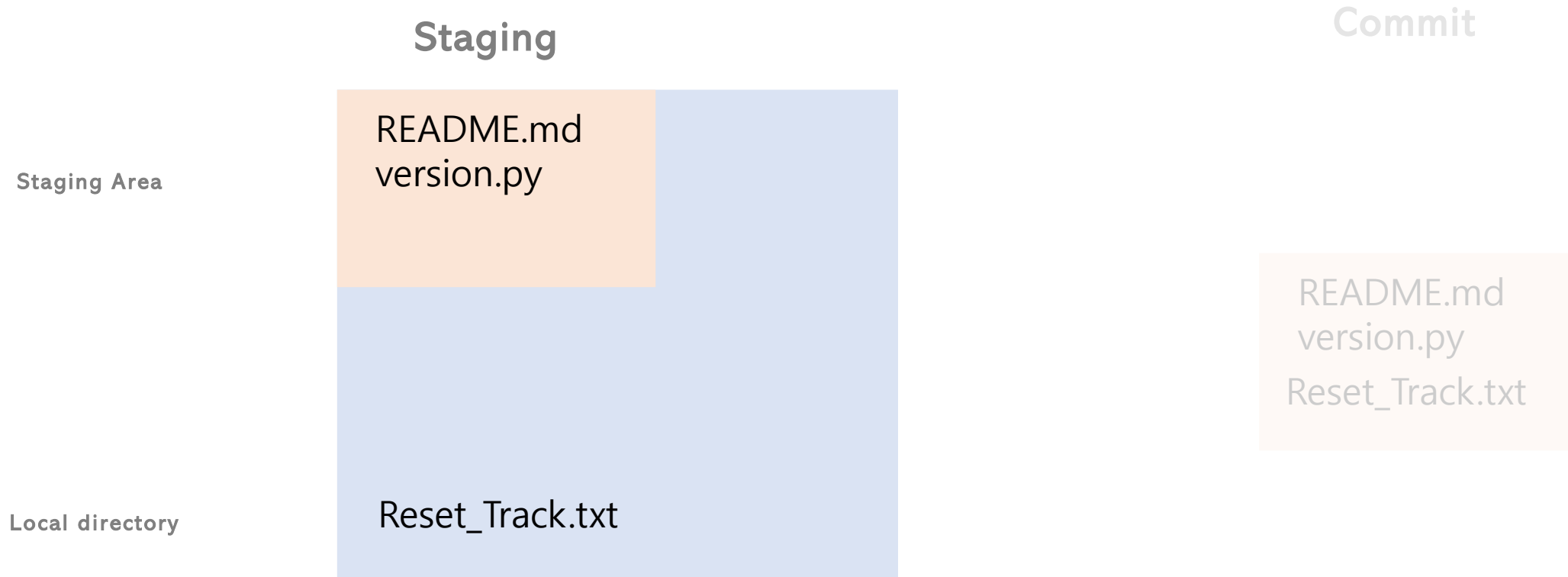
```
LikeLion_Git_Collaborate git:(feat/reset) git reset 99c2afa76fa4
```



수정된 내용들은 남아있지만, 스테이징이 되지 않았습니다!

Reset

git 현상태



수정된 내용들은 남아있지만, 스테이징이 되지 않았습니다!

Reset – hard

이해하기 위한 첫 단추

단계

✓ 기존 커밋

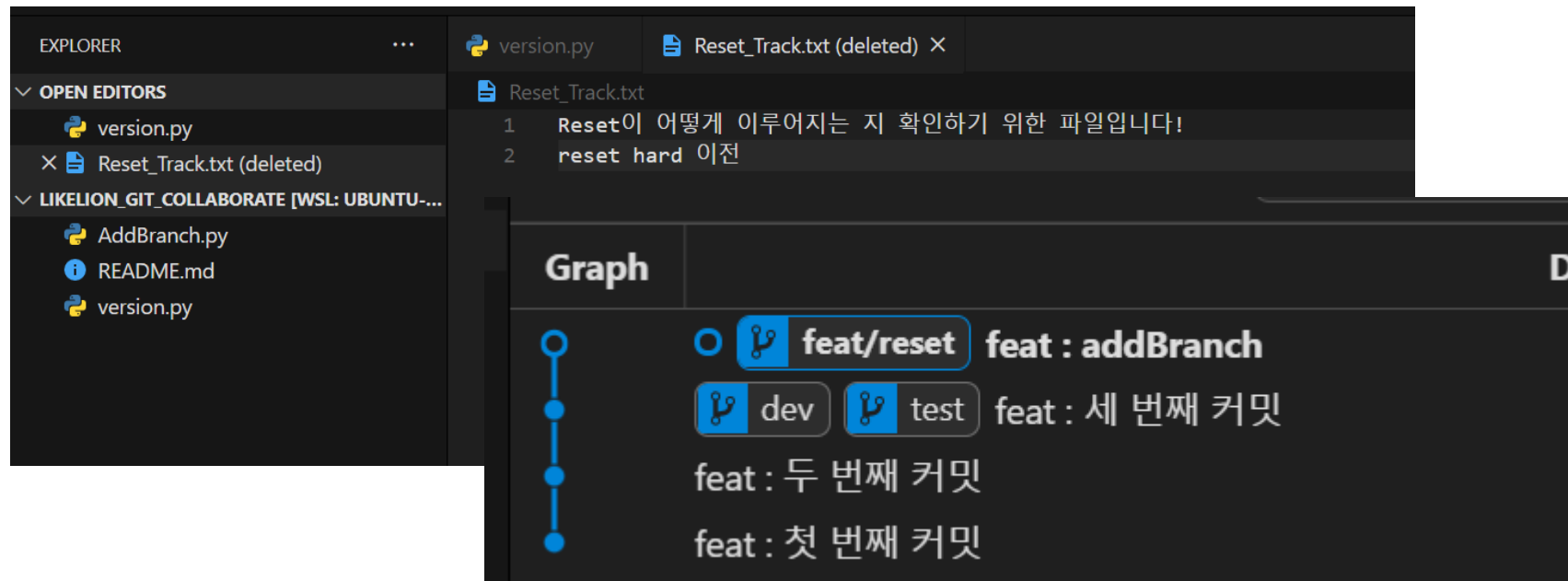
내용 변경

변경된 내용 스테이징

커밋!

수정된 내용들도 모두 삭제 됩니다! (커밋 그 자체로 돌아감)

```
→ LikeLion_Git_Collaborate git:(feat/reset) git reset --hard 99c2afa76fa4bdfc9
```



Reset

git 현상태

Staging

Commit

Staging Area

README.md
version.py

Local directory

README.md
version.py
Reset_Track.txt

수정된 내용들도 모두 삭제 됩니다! (커밋 그 자체로 돌아감)

Chapter 5

pull request & review

실습 준비

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!

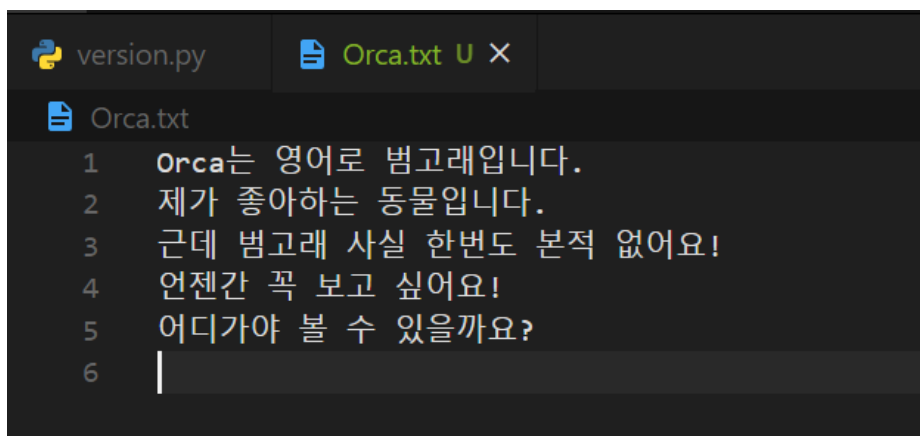
git checkout dev

git checkout -b "feat/{닉네임}/pull-request" (ex : feat/OrcaLion/pull-request)

git push origin "feat/{닉네임}/pull-request"

git checkout -b "feat/pull-request"

5줄 이상의 텍스트 파일 추가 (이름은 상관없어요!)

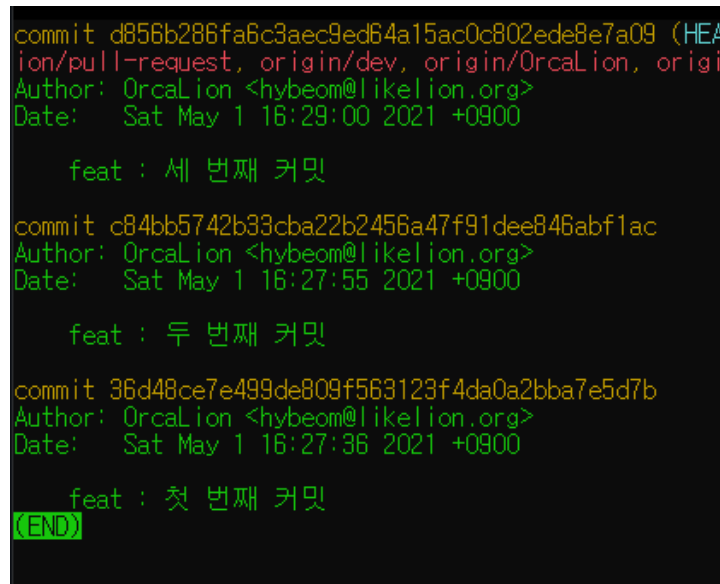


```
version.py  Orca.txt U x
Orca.txt
1  Orca는 영어로 범고래입니다.
2  제가 좋아하는 동물입니다.
3  근데 범고래 사실 한번도 본적 없어요!
4  언젠간 꼭 보고 싶어요!
5  어디가야 볼 수 있을까요?
6  |
```

git add .

git commit -m "feat : pull request"

Checkout 후에
Git log를 입력하여
Feat : 세 번째 커밋 인지 확인



```
commit d856b286fa6c3aec9ed64a15ac0c802ede8e7a09 (HEAD)
    feat : 세 번째 커밋
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:29:00 2021 +0900

commit c84bb5742b33cba22b2456a47f91dee846abf1ac
    feat : 두 번째 커밋
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:27:55 2021 +0900

commit 36d48ce7e499de809f563123f4da0a2bba7e5d7b
    feat : 첫 번째 커밋
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:27:36 2021 +0900

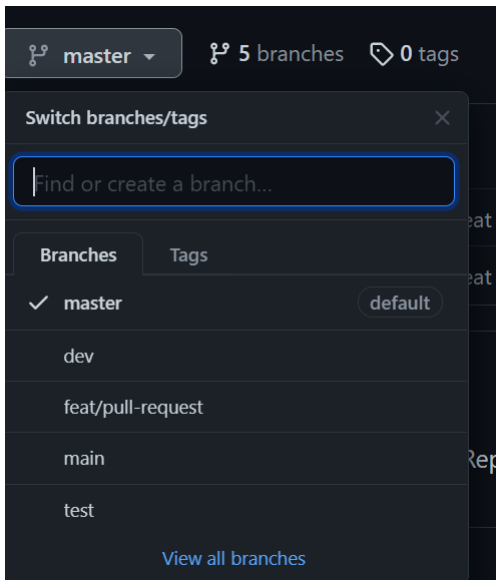
(END)
```

레포지토리 브랜치로 푸시하기

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!

git push origin feat/pull-request

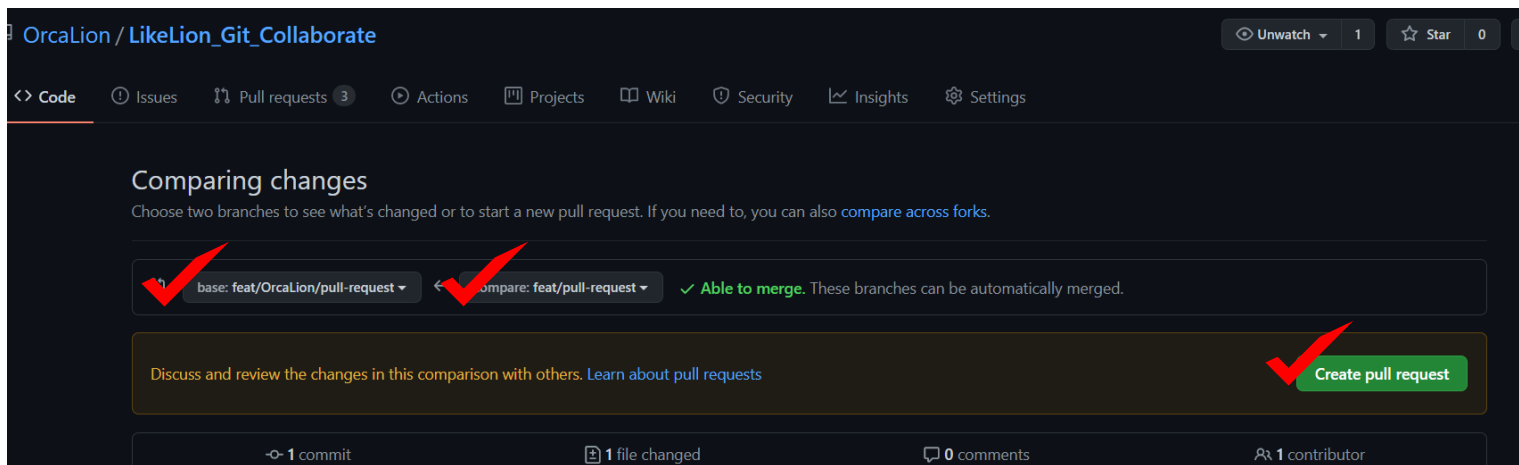
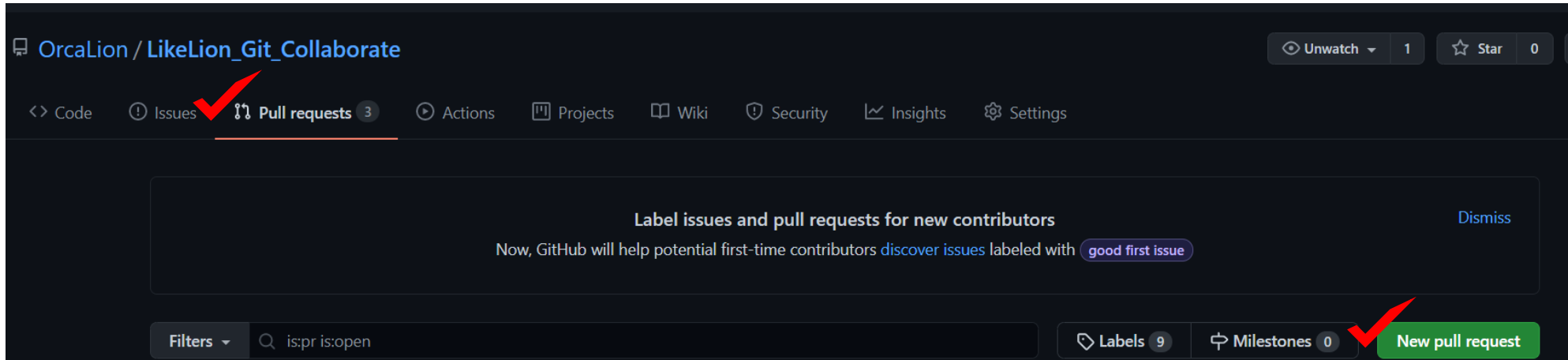
```
→ LikeLion_Git_Collaborate git:(feat/pull-request) git push origin feat/pull-request
Username for 'https://github.com': OrcaLion
Password for 'https://OrcaLion@github.com':
```



본인의 브랜치 중 하나에 올라가 있는 것 확인!

레포지토리 브랜치에서 pull-request해보기

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!




Merge라는 말은 좀 있다
또 볼 거예요.

레포지토리 브랜치에서 pull-request해보기


Pull request 작업을 위해서 다음과 같은 작업을 해주세요!

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

 base: feat/OrcaLion/pull-request < compare: feat/pull-request

✓ **Able to merge.** These branches can be automatically merged.




나의 첫 pull request (나의 레포에서)

Write

Preview

H B I ≡ <> 🔗 ≡ ≡ ☑ @ ↻ ↵


나의 레포에서 처음으로 pull request를 해보았습니다.

Attach files by dragging & dropping, selecting or pasting them. 

✓

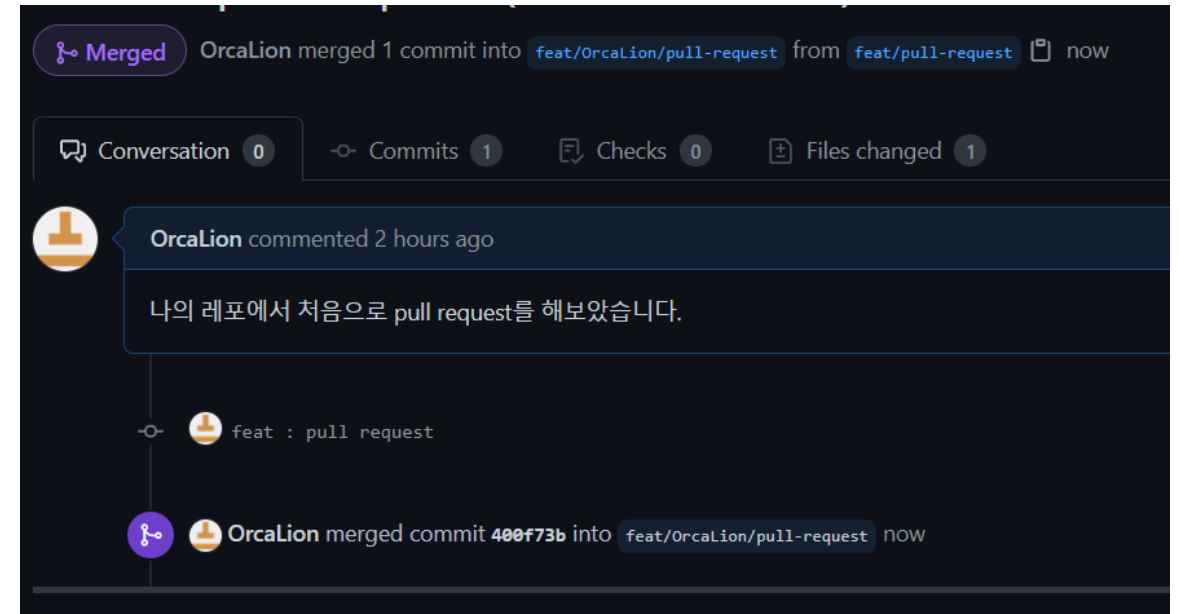
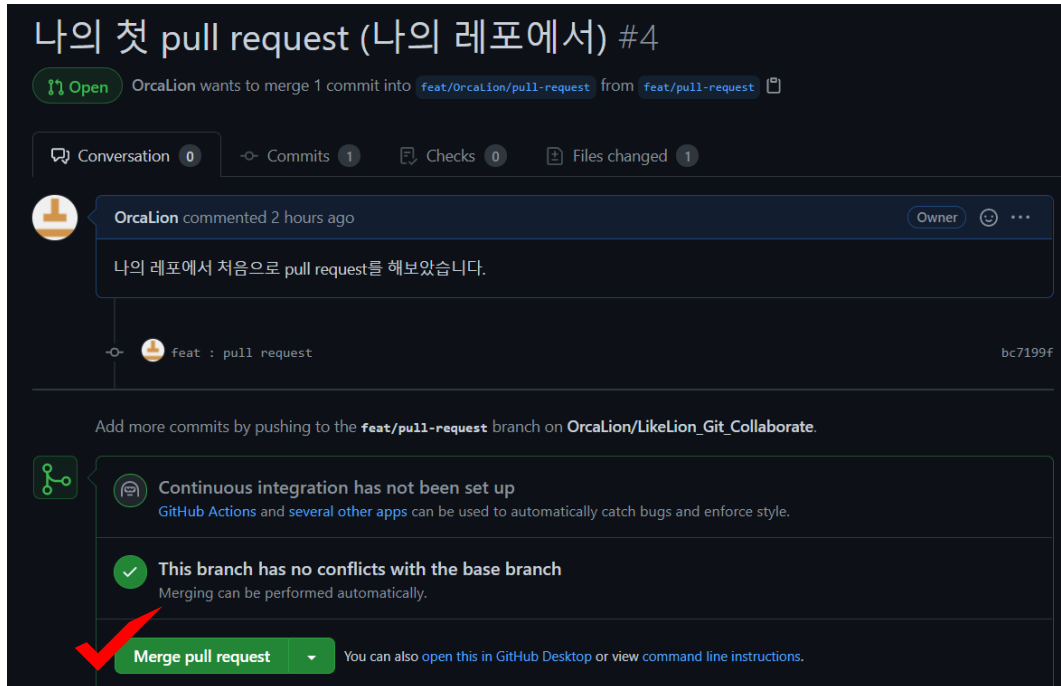
Create pull request

▼

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

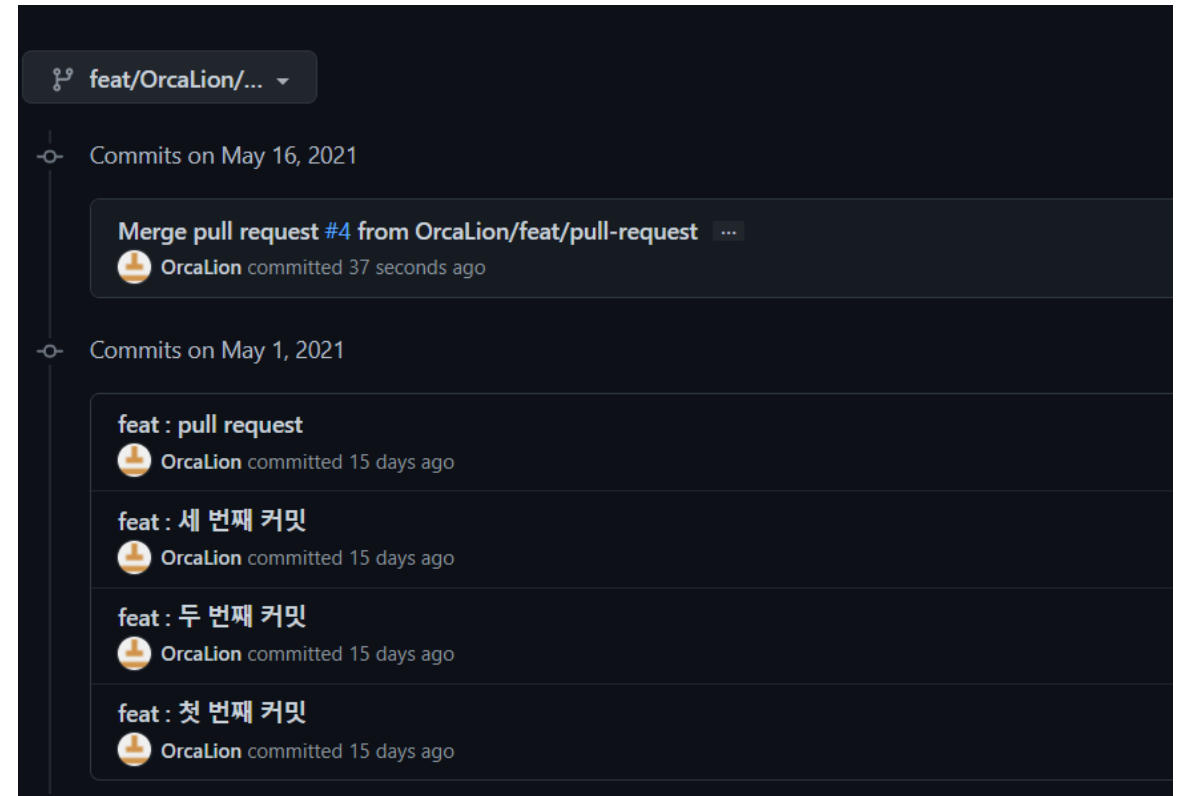
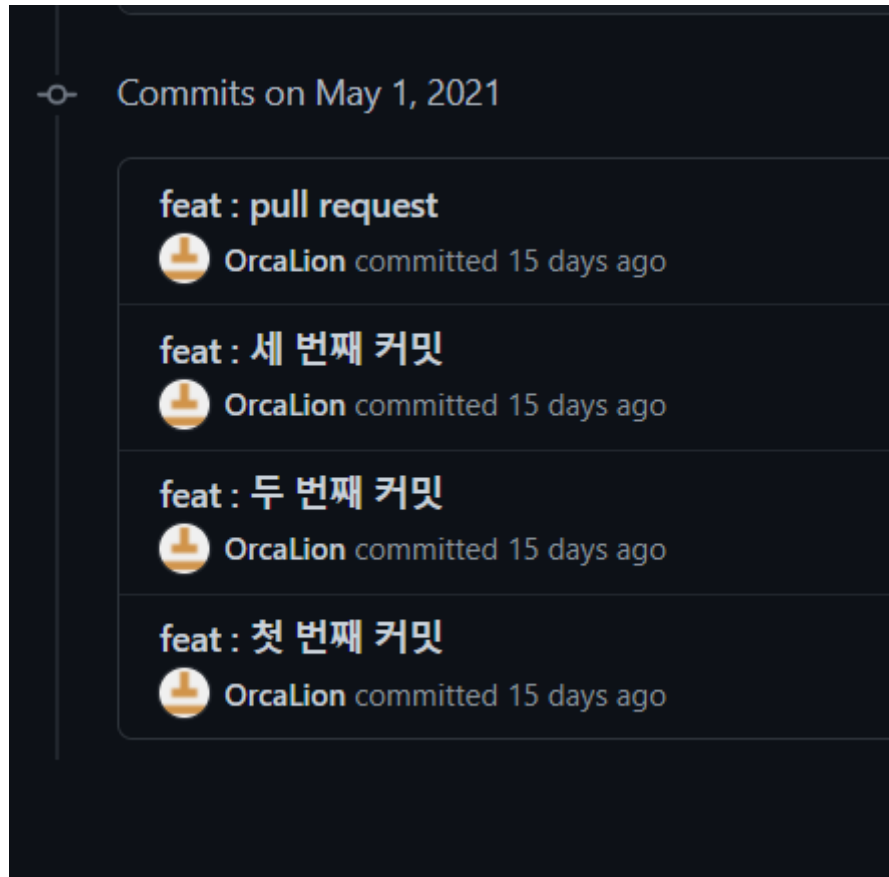
레포지토리 브랜치에서 pull-request해보기

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!

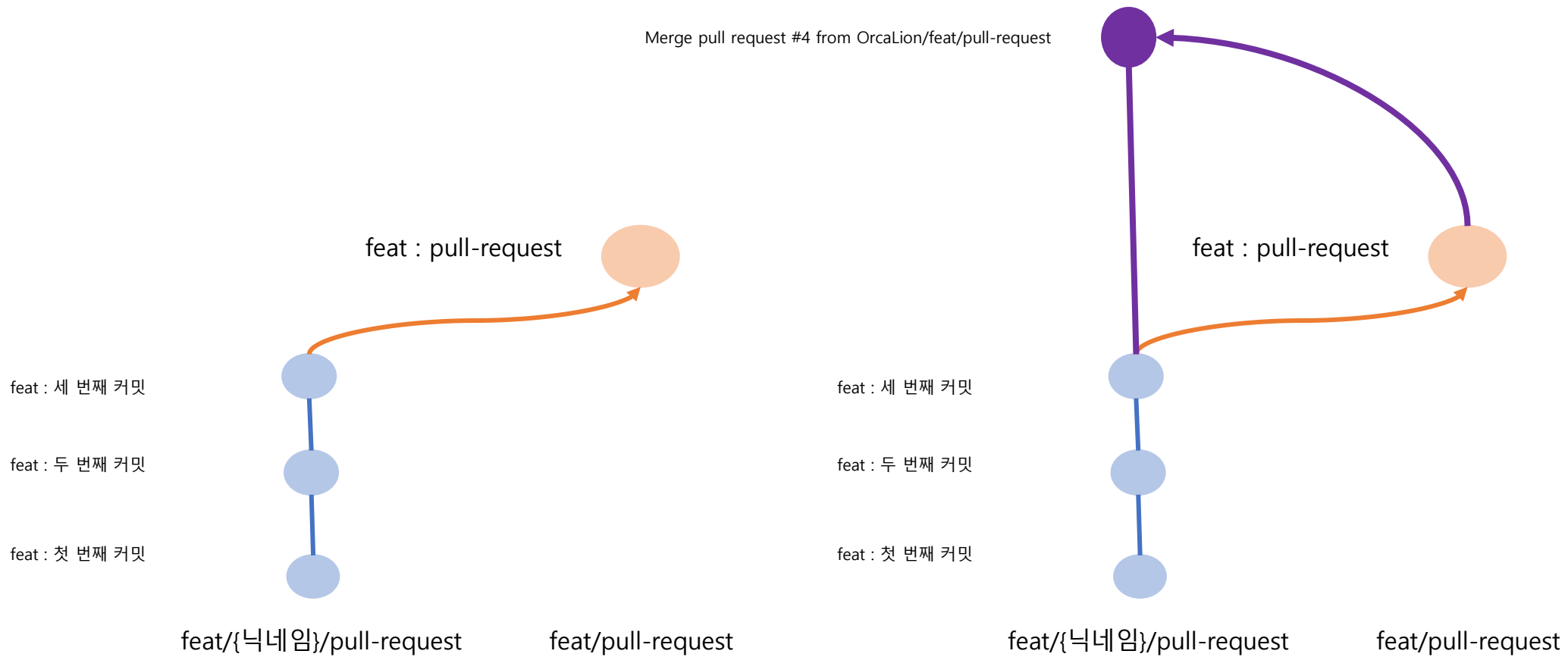


레포지토리 브랜치에서 pull-request해보기

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!

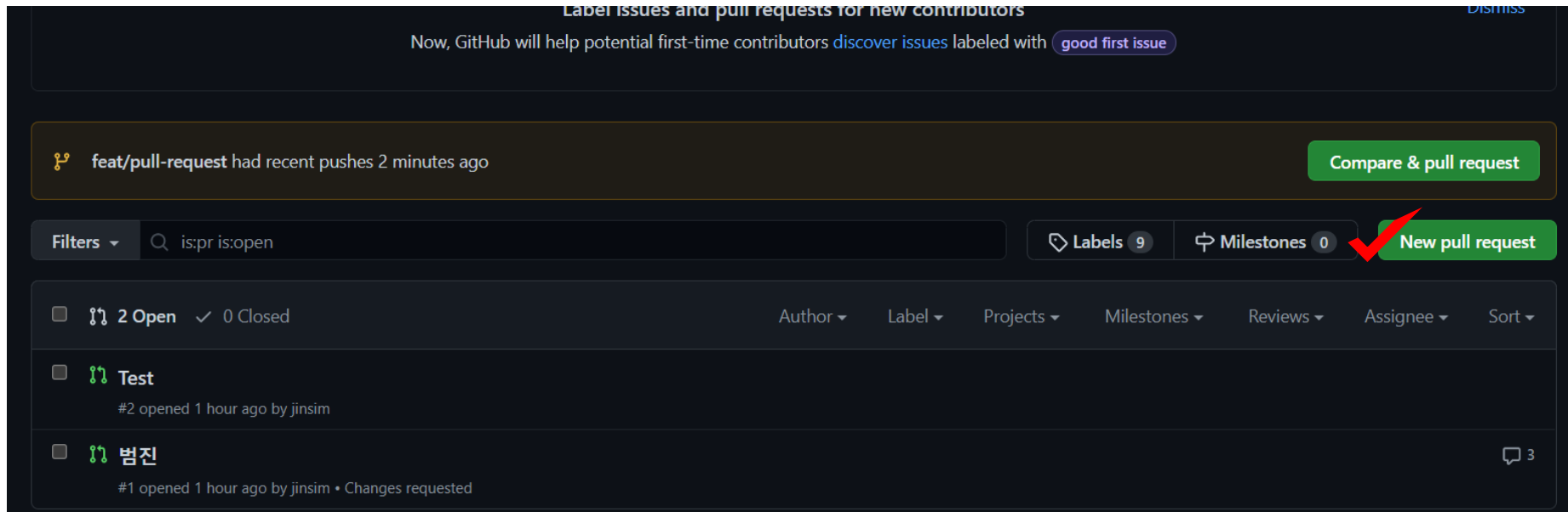


현재 상황



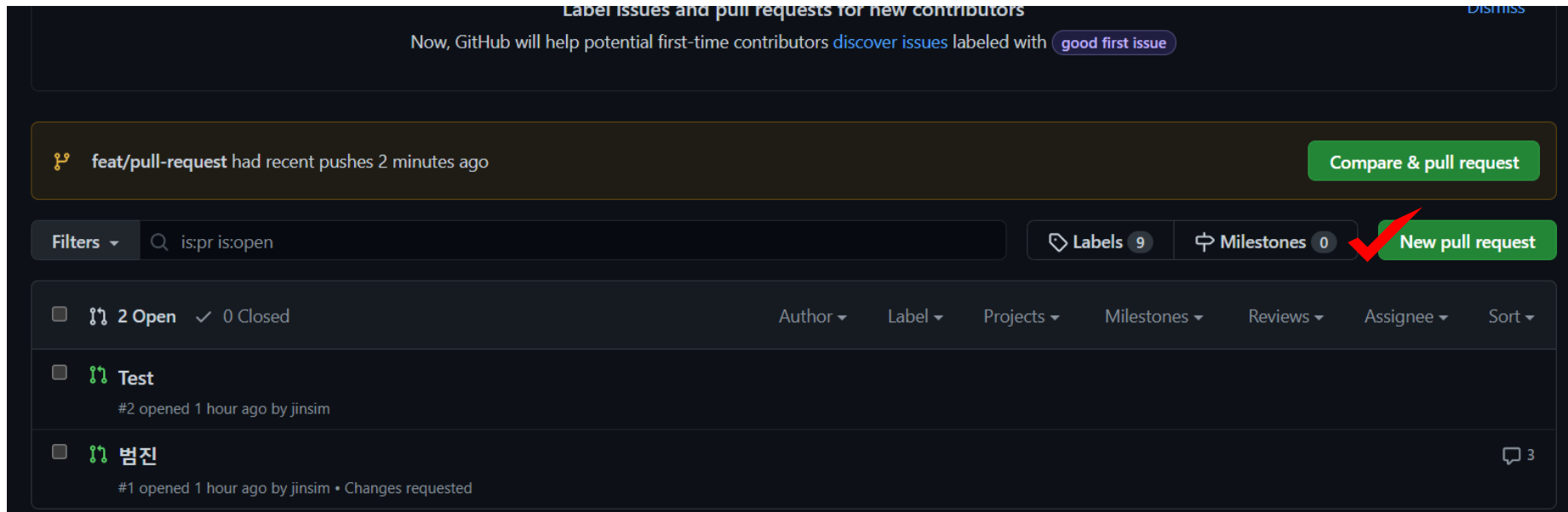
다른 레포지토리에서 Pull – request!

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!



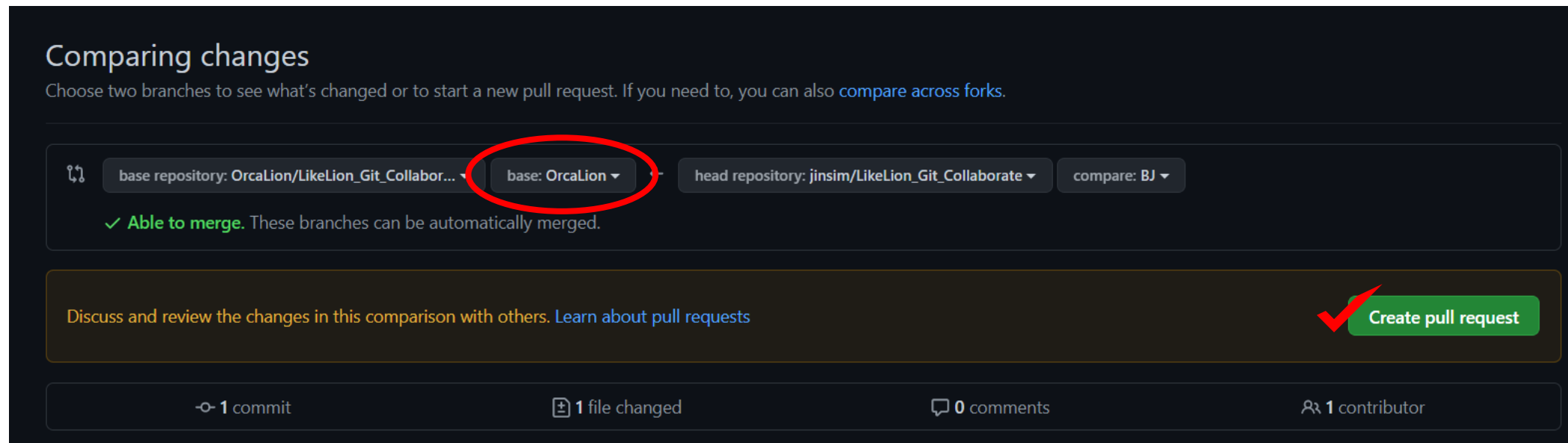
다른 레포지토리에서 Pull – request!

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!



다른 레포지토리에서 Pull – request!

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!



본인 닉네임으로 푸쉬 해주세요!

다른 레포지토리에서 Pull – request!

Pull request 작업을 위해서 다음과 같은 작업을 해주세요!

base repository: OrcaLion/LikeLion_Git_Collabor... base: OrcaLion head repository: jinsim/LikeLion_Git_Collaborate compare: BJ

✓ **Able to merge.** These branches can be automatically merged.

<본인 이름> - 나의 첫 pull request입니다.

Write Preview

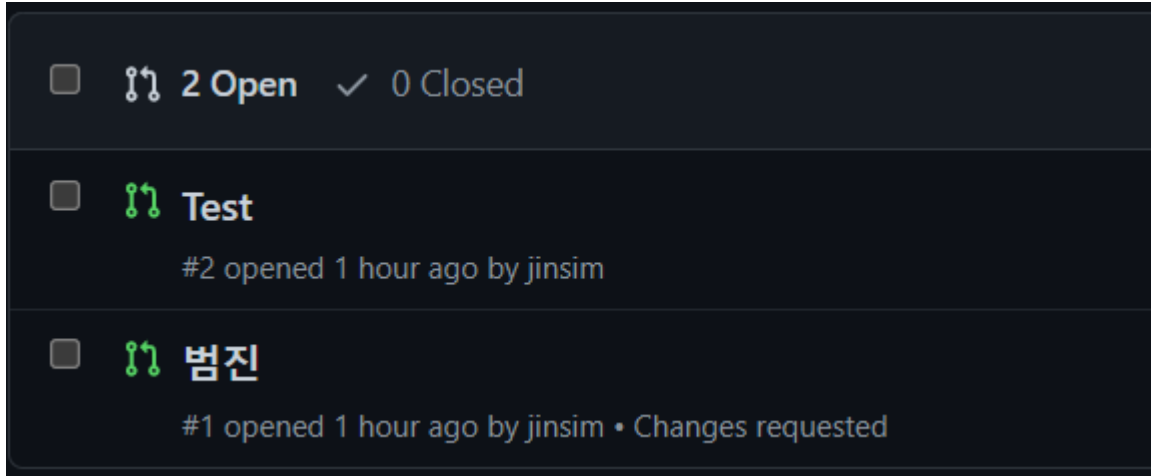
잘 부탁드립니다!

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

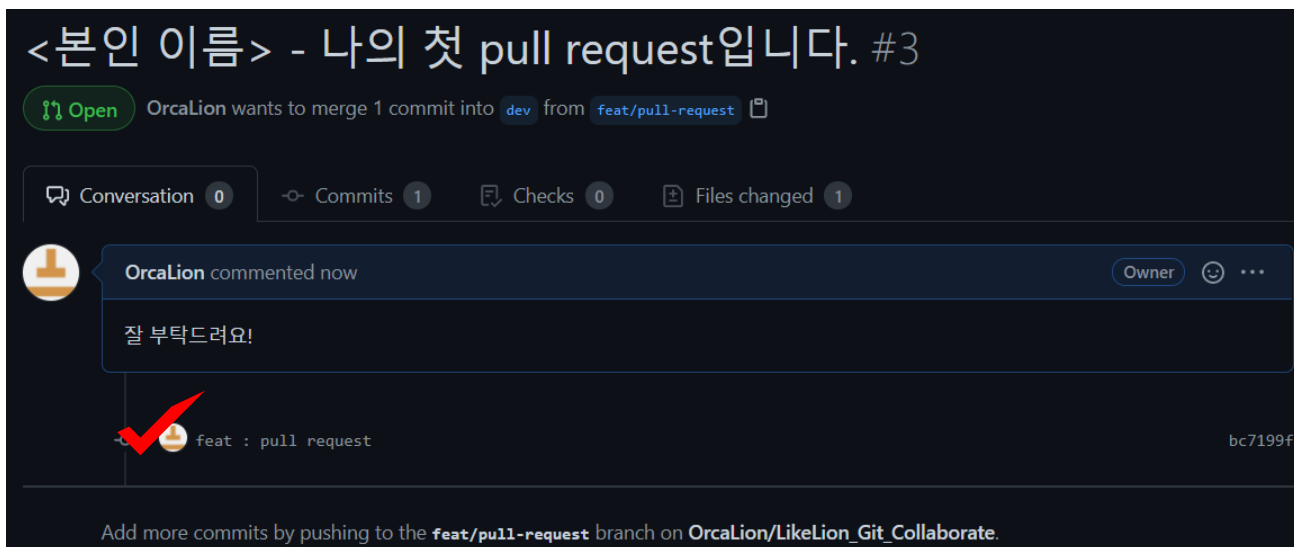
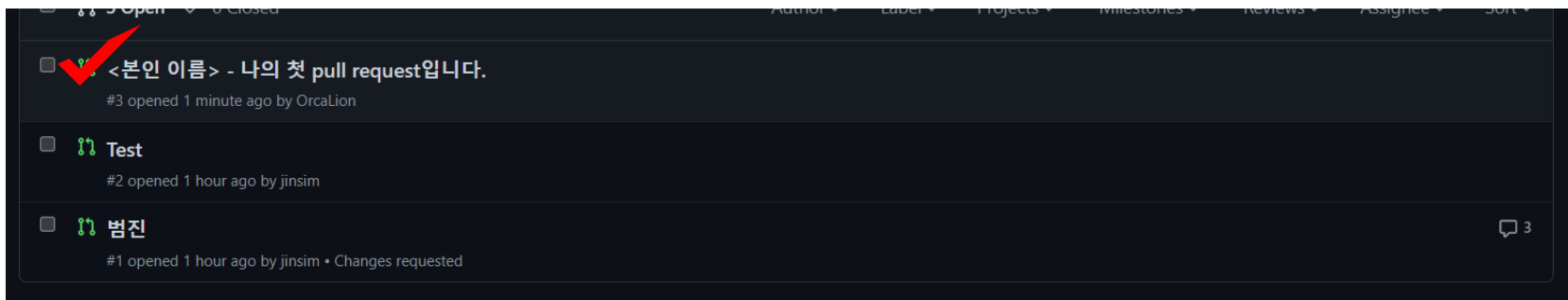
Pull – request!

생성 확인!



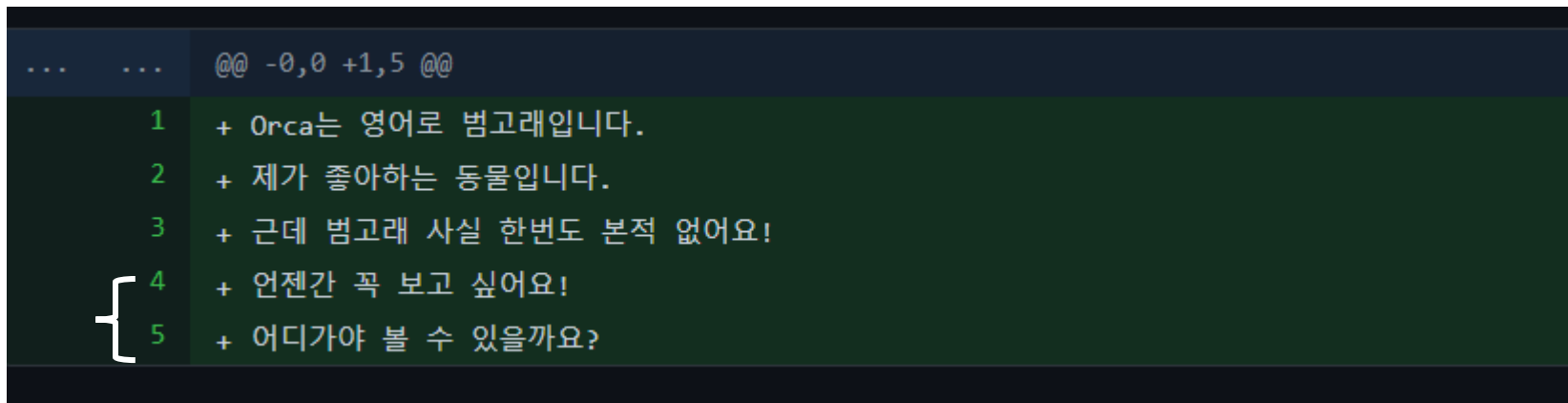
Review 다른 사람 코드에 의견 주기

리뷰 시간 룰루랄라

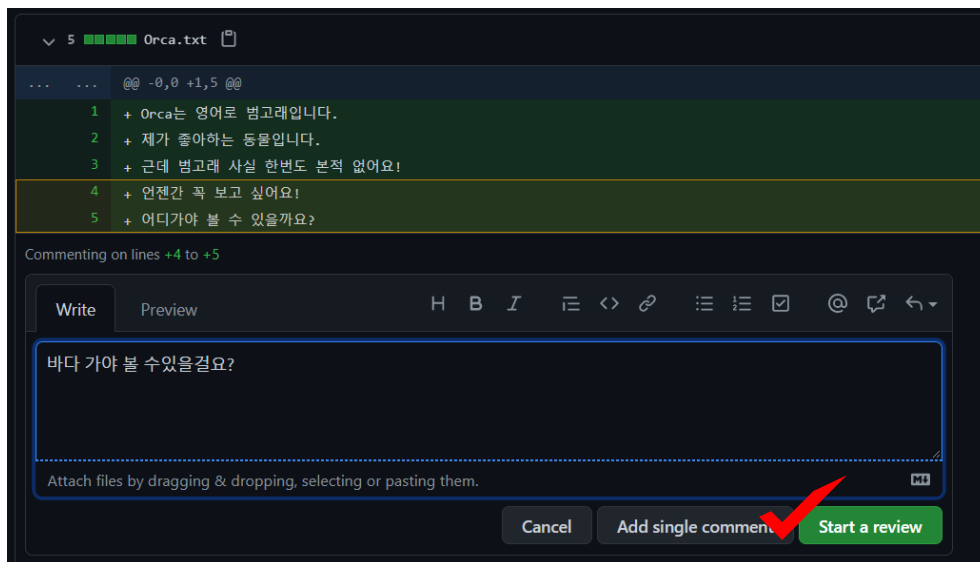


Review 다른 사람 코드에 의견 주기

리뷰 시간 롤루랄라



드래그!



이후 해당 커밋에 원하는 만큼 리뷰 남긴다

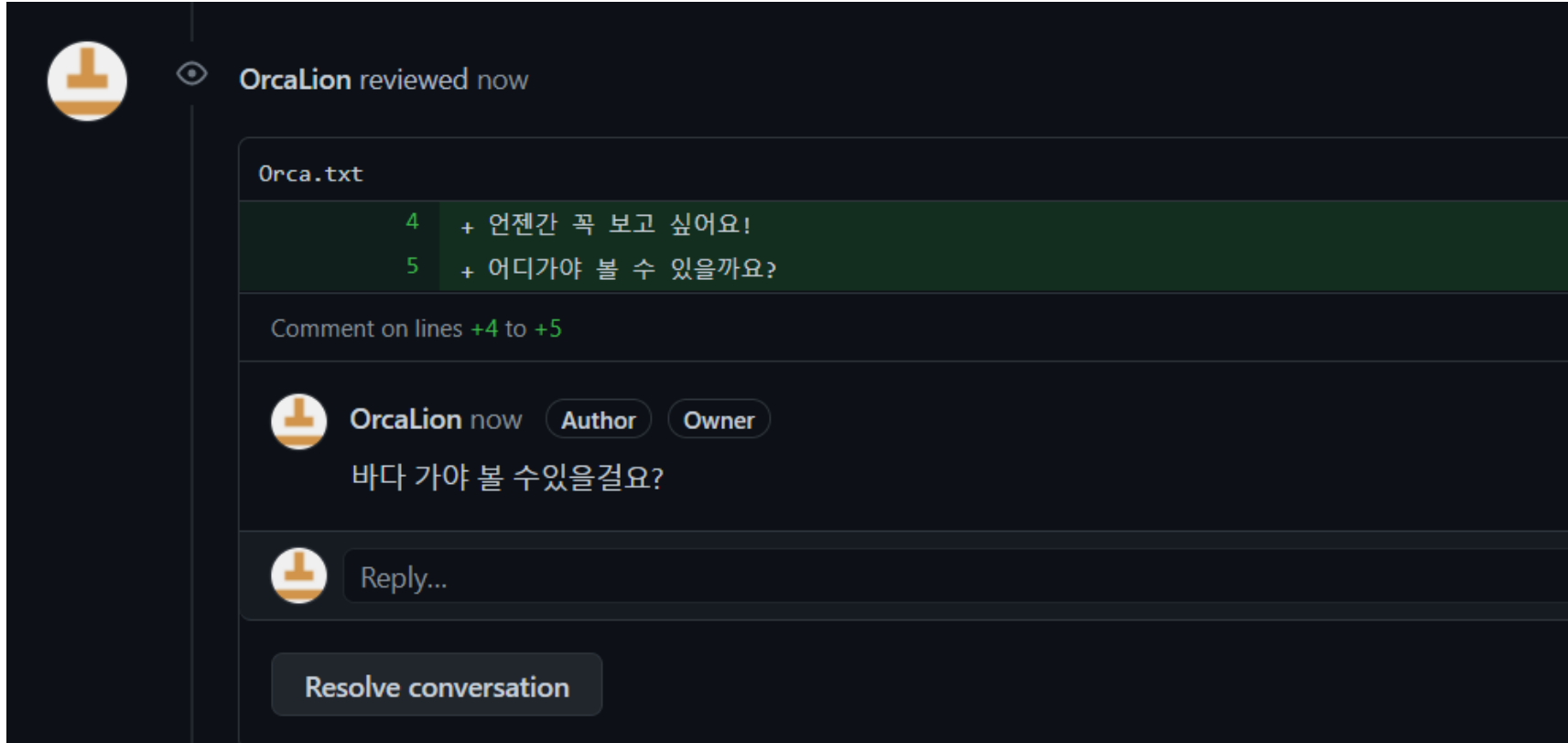
리뷰 시간 룰루랄라



Session 0

Review 다른 사람 코드에 의견 주기

Review 남은 거 확인



OrcaLion reviewed now

Orca.txt

4 + 언젠간 꼭 보고 싶어요!

5 + 어디가야 볼 수 있을까요?

Comment on lines +4 to +5

OrcaLion now Author Owner

바다 가야 볼 수있을걸요?

Reply...

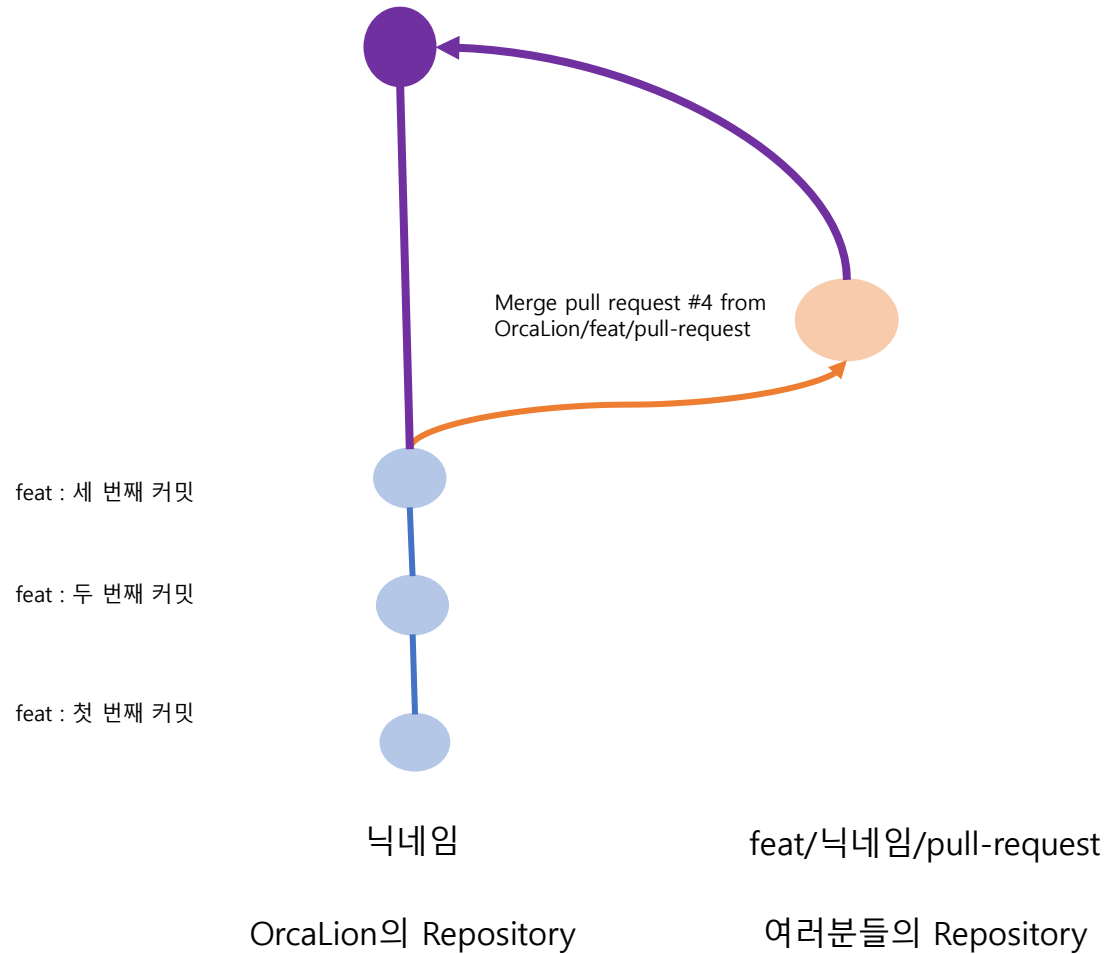
Resolve conversation

Chapter 5

fetch & merge

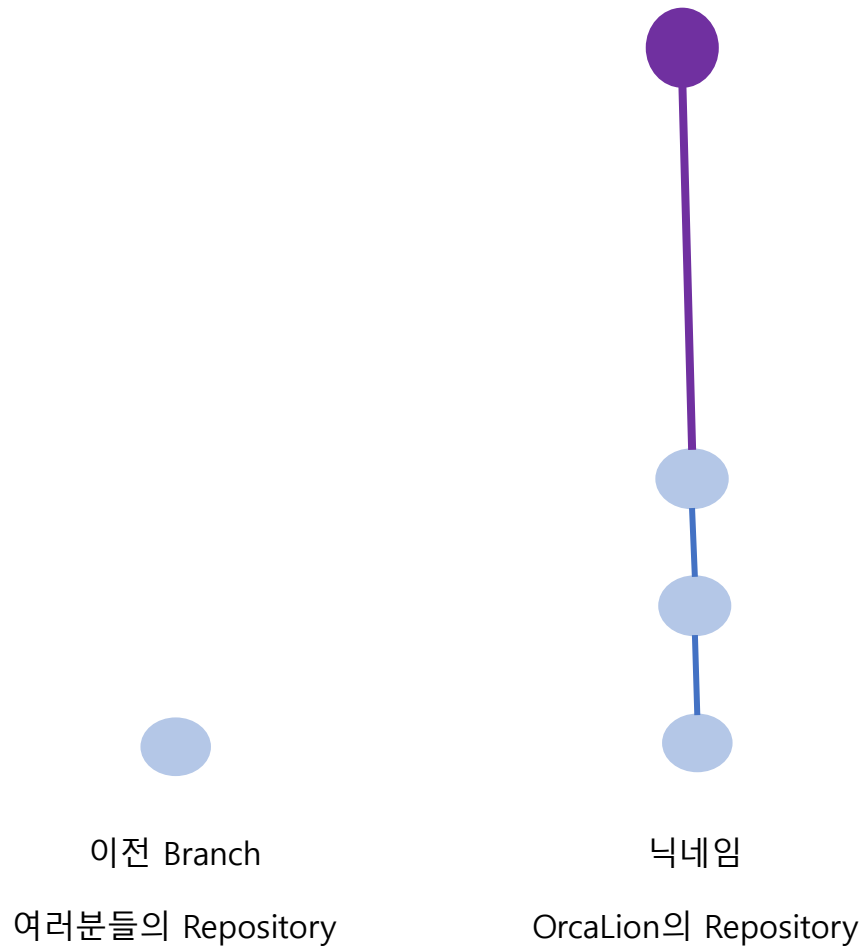
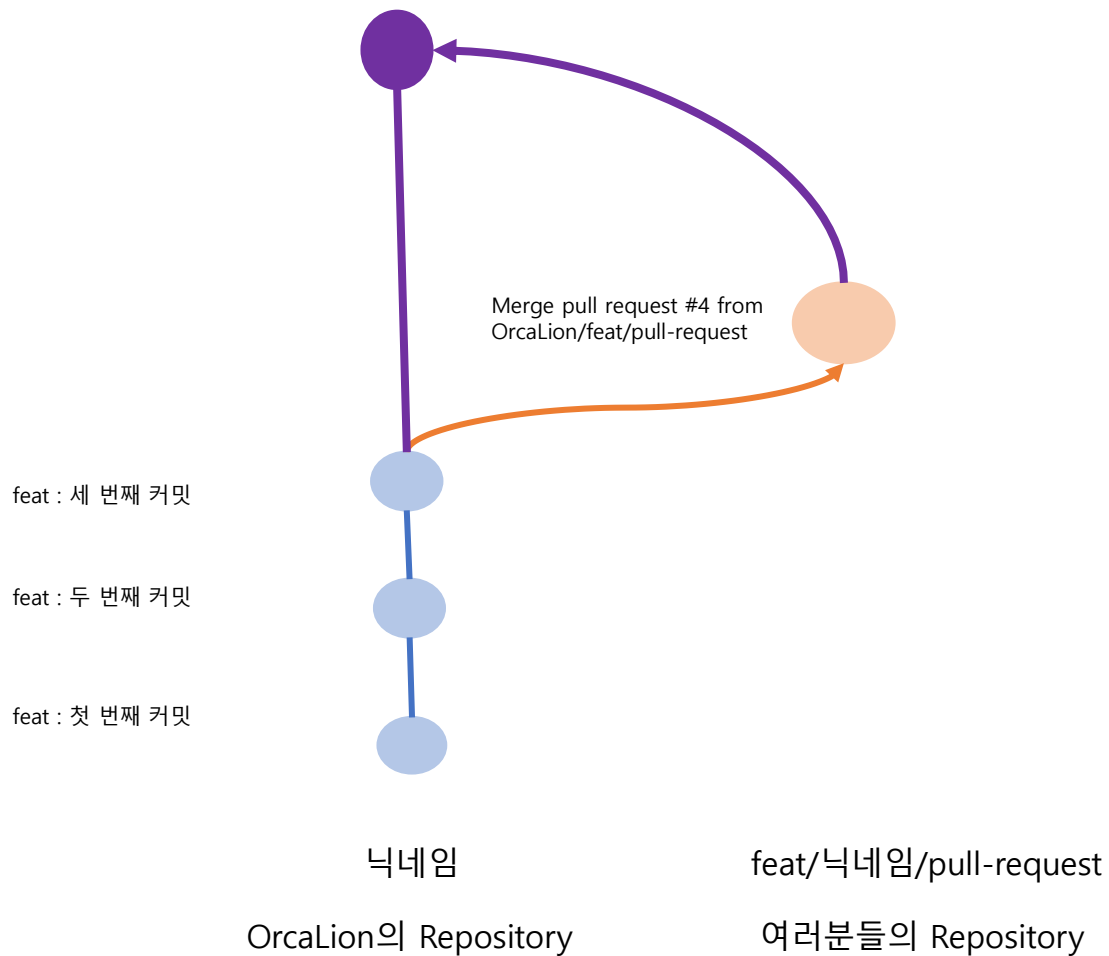


현재 상황



현재 상황

변경 사항을 기존 Repo에 병합하고 싶다면!



remote : 원격 저장소 등록하기

```
git remote add upstream "https://github.com/OrcaLion/LikeLion_Git_Collaborate.git"
```

-> 원격 저장소를 등록하는 명령어입니다. (첫날 세션 참고)

```
git remote -v
```

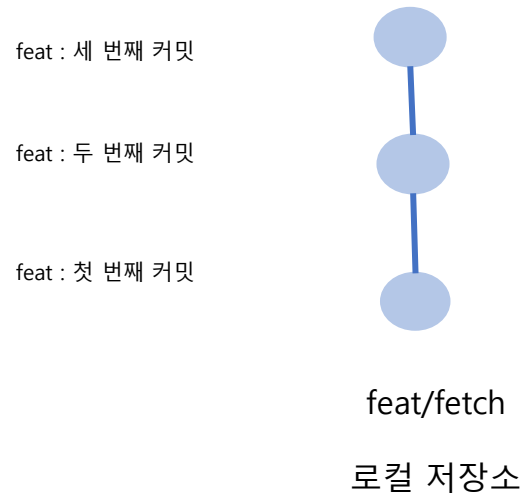
-> 등록된 저장소 확인

Fetch : 저장소 내용 확인하기

git checkout dev

git checkout -b "feat/fetch"

(새 브랜치 파기)



현재 상태

Fetch : 저장소 내용 확인하기

git checkout dev

git checkout -b "feat/fetch"

(새 브랜치 파기)

```
Switched to branch "feat/fetch"  
→ LikeLion_Git_Collaborate git:(feat/fetch) git fetch origin feat/pull-request  
From https://github.com/OrcaLion/LikeLion_Git_Collaborate  
* branch          feat/pull-request -> FETCH_HEAD  
→ LikeLion_Git_Collaborate git:(feat/fetch) _
```

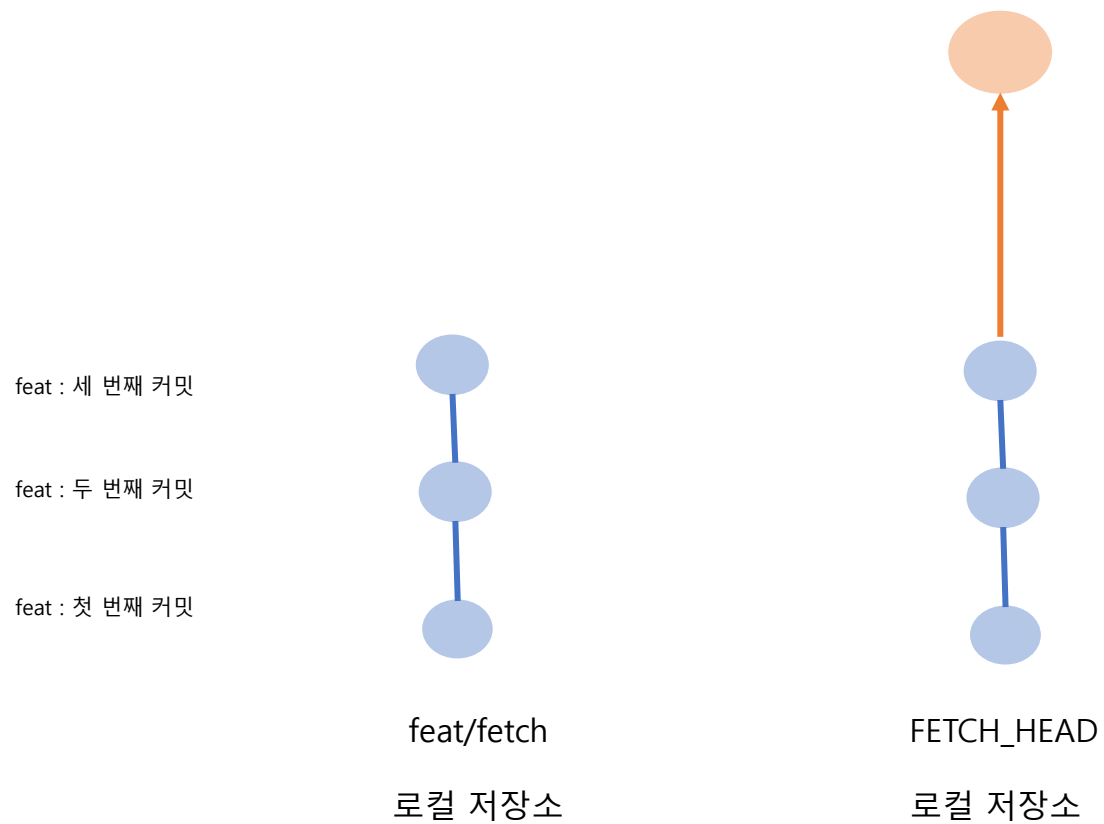
git fetch (해당 저장소) (가져오고 싶은 브랜치)

Ex) git fetch origin feat/pull-request

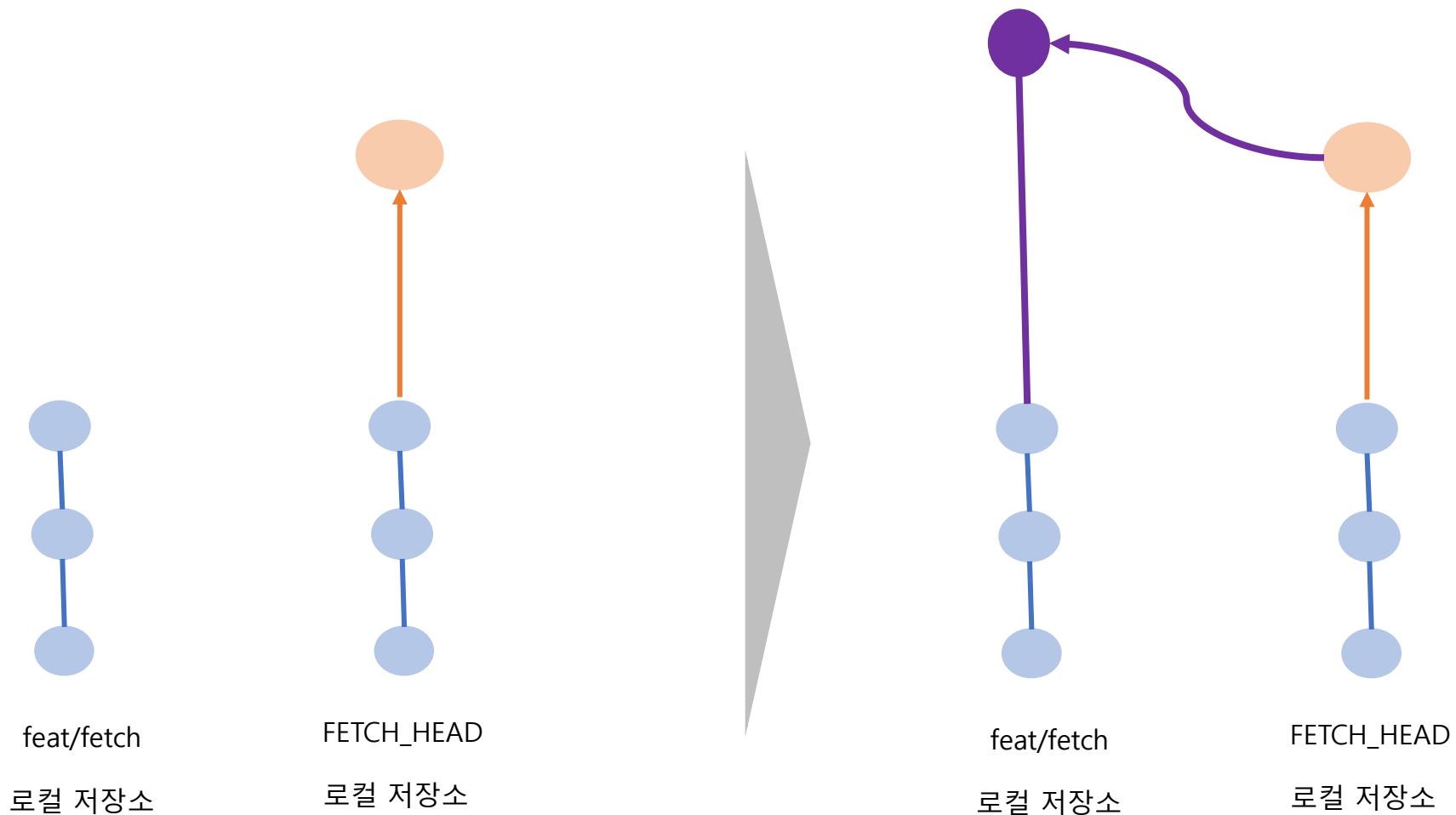
FETCH_HEAD라는 브랜치가 생성이 되면서 우리가 보고 싶은 저장소가 끌어와집니다!

확인하려면 git checkout FETCH_HEAD

Fetch : 저장소 내용 확인하기



Merge : 브랜치 병합하기



Merge : 브랜치 병합하기

git merge FETCH_HEAD

```
* branch          feat/pull-request -> FETCH_HEAD
→ LikeLion_Git_Collaborate git:(feat/fetch) git merge FETCH_HEAD
Updating d856b28..bc7199f
Fast-forward
 Orca.txt | 5 +++++
 1 file changed, 5 insertions(+)
 create mode 100644 Orca.txt
→ LikeLion_Git_Collaborate git:(feat/fetch)
```

Graph	Description
○	○ feat/fetch feat/pull-request: 1 M
	Commit: bc7199fcd1db915b12335...
	Parents: d856b286fa6c3aec9ed64a...
	Author: OrcaLion < hybeom@likelion.org >
	Committer: OrcaLion < hybeom@likelion.org >
	Date: Sat May 01 2021 19:45:46 GMT+0900 (Korean Standard Time)
	feat : pull request
●	dev feat/OrcaLion/pull-reque: 1 M
●	feat : 두 번째 커밋 1 M
●	feat : 첫 번째 커밋 1 M

Merge : 브랜치 병합하기

다른 브랜치들도 병합할 수가 있어요!

git checkout dev

원래 브랜치로 돌아가기

git checkout -b "feat/merge"

머지 배우는 branch 만들기

git merge "feat/pull-request"

다른 브랜치 머지하기

```
The most similar command is
checkout
→ LikeLion_Git_Collaborate git:(merge) git checkout dev
Switched to branch 'dev'
Your branch is up to date with 'origin/dev'.
→ LikeLion_Git_Collaborate git:(dev) git checkout -b "feat/merge"
Switched to a new branch 'feat/merge'
→ LikeLion_Git_Collaborate git:(feat/merge) git merge "feat/pull-request"
Updating d856b28..bc7199f
Fast-forward
 Orca.txt | 5 +++++
 1 file changed, 5 insertions(+)
 create mode 100644 Orca.txt
```

Vs pull

자주 쓰이는 명령어, 뭐가 다른걸까요?

```
git fetch {저장소} {브랜치}
```

```
git merge FETCH_HEAD
```



```
git pull {저장소} {브랜치}
```

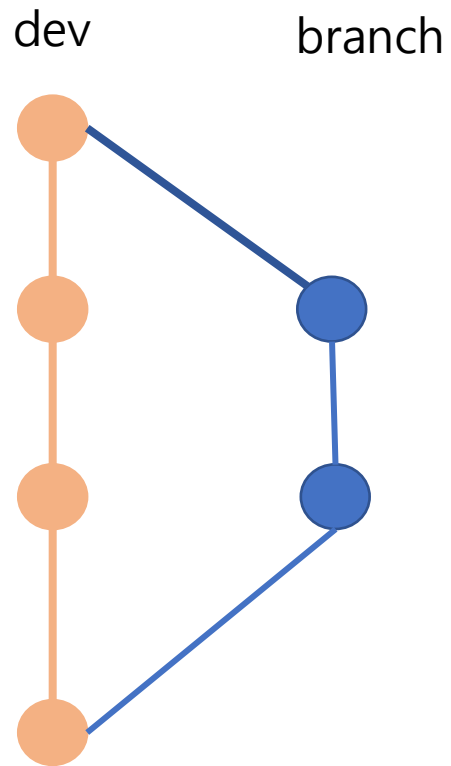
Chapter 6

rebase

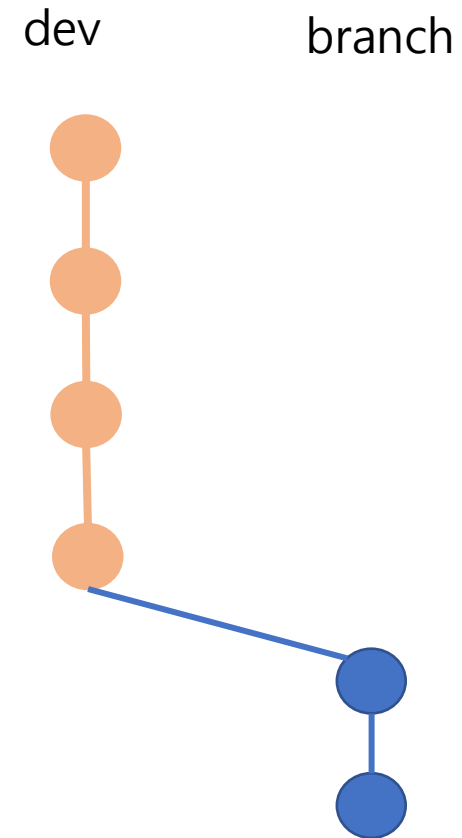
rebase

병합을 하는 두 가지 방법

Merge



Rebase



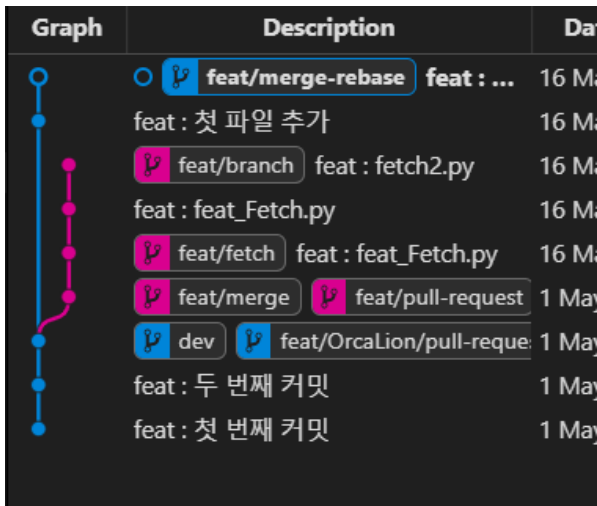
rebase

먼저, 기준이 될 브랜치 하나를 가져와보도록 하겠습니다.

```
git fetch upstream feat/merge-rebase
```

```
git checkout FETCH_HEAD
```

```
git checkout -b "feat/merge-rebase"
```



Graph	Description	Date
○ feat/merge-rebase	feat : ...	16 May
○ feat : 첫 파일 추가		16 May
feat/branch	feat : fetch2.py	16 May
feat : feat_Fetch.py		16 May
feat/fetch	feat : feat_Fetch.py	16 May
feat/merge	feat/pull-request	1 May
dev	feat/OrcaLion/pull-reque	1 May
feat : 두 번째 커밋		1 May
feat : 첫 번째 커밋		1 May

똑같은 환경으로 해보고 싶으신분은

```
git fetch upstream feat/target
```

```
git checkout FETCH_HEAD
```

```
git checkout -b "feat/target"
```

rebase

Merge의 경우에는?

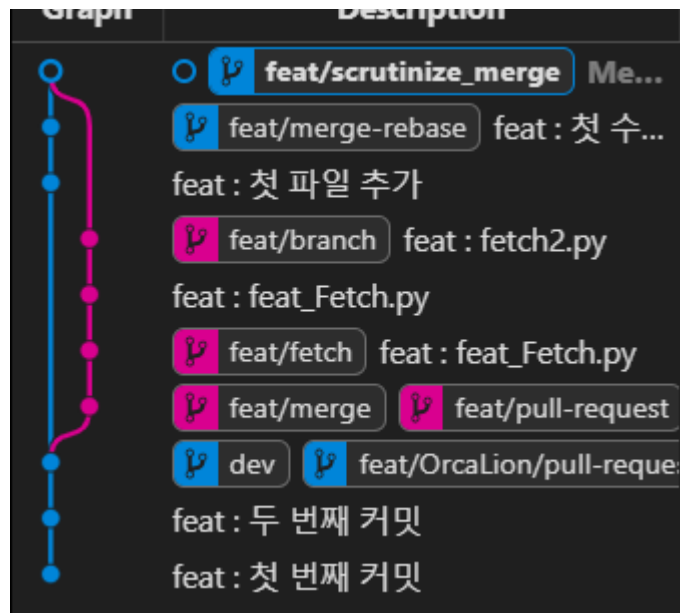
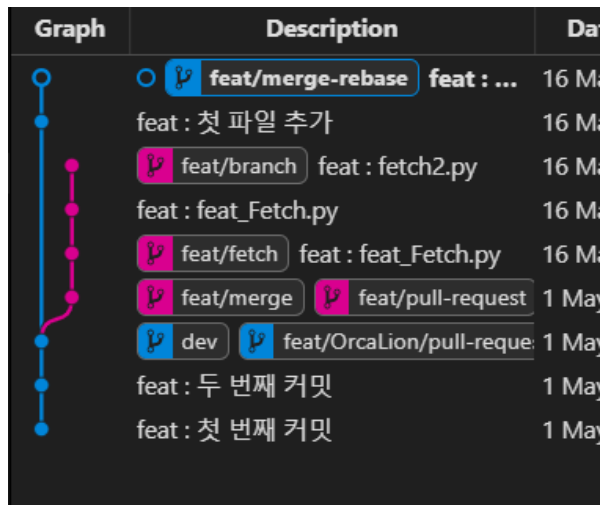
git checkout -b "feat/scrutinize_merge"

git merge feat/branch

Ctrl x 누르시면 됩니다!

git merge feat/target

ex) target 가져 오신분들



```
git
commit: af84d4c9b91991858b5c70759d25ede1056809e4 (HEAD -> feat/scrutinize_merge)
Merge: dd3a825 e7695a2
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 18:15:42 2021 +0900

    Merge branch 'feat/branch' into feat/scrutinize_merge

commit: dd3a825089e33ef6ec9cac827c4d4b9341302f37 (origin/feat/merge-rebase, feat/merge-rebase)
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 18:01:56 2021 +0900

    feat : 첫 수정 추가

commit: 2cc621b3df095012a01320099bf4679c491bd650
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 18:01:39 2021 +0900

    feat : 첫 파일 추가

commit: e7695a2cd2b3b1c613c5a1c9566e19f27ff6643f (feat/branch)
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 17:44:03 2021 +0900

    feat : fetch2.py

commit: 8315a55290ee977d2f079d731c018aef3624c6cd
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 17:43:26 2021 +0900

    feat : feat_Fetch.py

commit: 17afc99d3d454fd0217d1ecb9ad5473a629110c3 (feat/fetch)
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 17:41:37 2021 +0900

    feat : feat_Fetch.py

commit: bc7199fcd1db915b12335e623bff450e6f7d0d4f (origin/feat/pull-request, feat/pull-request, feat/merge)
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 19:45:46 2021 +0900

    feat : pull request

commit: d856b286fa6c3aec9ed64a15ac0c802ede8e7a09 (origin/test, origin/master, origin/main, origin/feat/OrcaLion/pull-request, origin/dev, origin/OrcaLion, origin/HEAD, merge, feat/OrcaLion/pull-request, dev)
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:29:00 2021 +0900

    feat : 세 번째 커밋

commit: c84bb5742b33cba22b2456a47f91dee84abf1ac
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:27:55 2021 +0900

    feat : 두 번째 커밋

commit: 36d48ce7e499de809f563123f4da0a2bba7e5d7b
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:27:36 2021 +0900

    feat : 첫 번째 커밋
```

rebase

rebase의 경우에는?

git checkout "feat/merge-rebase"


git branch -D "feat/scrutinize_merge"


git checkout -b "feat/scrutinize_rebase" git rebase feat/target

git rebase feat/branch

Ctrl x 누르시면 됩니다!

ex) target 가져 오신분들

Graph	Description	Date
	feat/merge-rebase feat : ...	16 May 2021
	feat : 첫 파일 추가	16 May 2021
	feat/branch feat : fetch2.py	16 May 2021
	feat : feat_Fetch.py	16 May 2021
	feat/fetch feat : feat_Fetch.py	16 May 2021
	feat/merge feat/pull-request	1 May 2021
	dev feat/OrcaLion/pull-request	1 May 2021
	feat : 두 번째 커밋	1 May 2021
	feat : 첫 번째 커밋	1 May 2021

	feat/scrutinize_rebase feat...
	feat : 첫 파일 추가
	feat/merge-rebase feat : 첫 수...
	feat : 첫 파일 추가
	feat/branch feat/target f...
	feat : feat_Fetch.py
	feat/fetch feat : feat_Fetch.py
	feat/merge feat/pull-request
	dev feat/OrcaLion/pull-reque...
	feat : 두 번째 커밋
	feat : 첫 번째 커밋

```
commit 64de27b455ab514e556e22813a1884923dd3c3ec (HEAD -> feat/scrutinize_rebase)
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 18:01:56 2021 +0900

    feat : 첫 수정 추가

commit 3a63eb7182010fc532a0016f8abba3c1be35a8f0
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 18:01:39 2021 +0900

    feat : 첫 파일 추가

commit e7695a2cd2b3b1c613c5a1c9566e19f27ff6643f (origin/feat/target, origin/feat/branch, feat/target, feat/branch)
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 17:44:03 2021 +0900

    feat : fetch2.py

commit 8315a55290ee977c2f079d731c018aef3624c6cd
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 17:43:26 2021 +0900

    feat : feat_Fetch.py

commit 17afc99d3d454fd0217d1ecb9ad5473a629110c3 (feat/fetch)
Author: hybeom0720 <hybeom@gmail.com>
Date: Sun May 16 17:41:37 2021 +0900

    feat : feat_Fetch.py

commit bc7199fcd1db915b12395e623bff450e6f7d0d4f (origin/feat/pull-request, feat/pull-request, feat/merge)
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 19:45:46 2021 +0900

    feat : pull request

commit d856b286fa6c3aec9ed64a15ac0c802ede8e7a00 (origin/test, origin/master, origin/main, origin/feat/OrcaLion/pull-request, origin/dev, origin/OrcaLion, origin/HEAD, merge, feat/OrcaLion/pull-request, dev)
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:29:00 2021 +0900

    feat : 세 번째 커밋

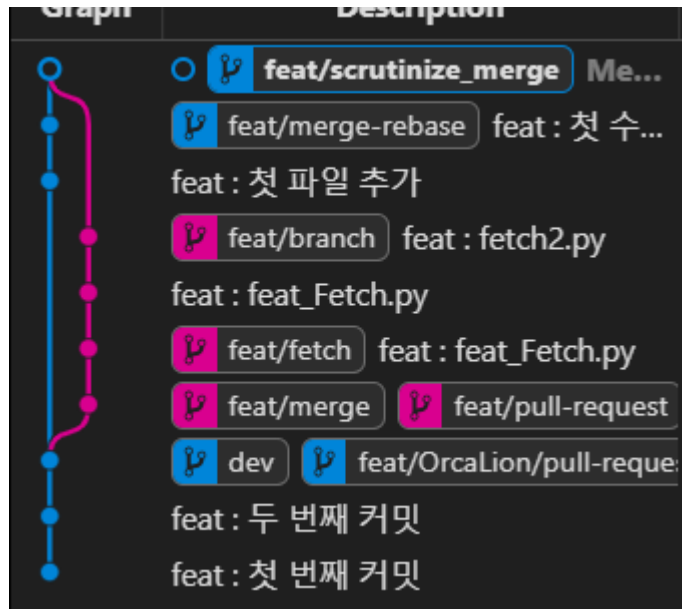
commit c84bb5742b33cba22b2456a47f91dee846abf1ac
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:27:55 2021 +0900

    feat : 두 번째 커밋

commit 36d48ce7e499de809f563123f4da0a2bba7e5d7b
Author: OrcaLion <hybeom@likelion.org>
Date: Sat May 1 16:27:36 2021 +0900

    feat : 첫 번째 커밋
```

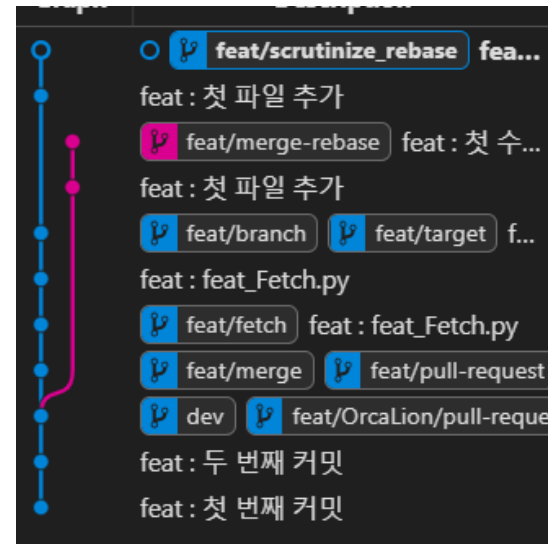
Rebase 와 Merge 비교



Merge의 경우

Merge의 경우 커밋 메시지가 하나 더 많다.

Rebase의 경우에는 병합의 성격보단 기본이 되는 base (master나 dev) 를 재설정 해주는 측면이 높다!



Rebase의 경우

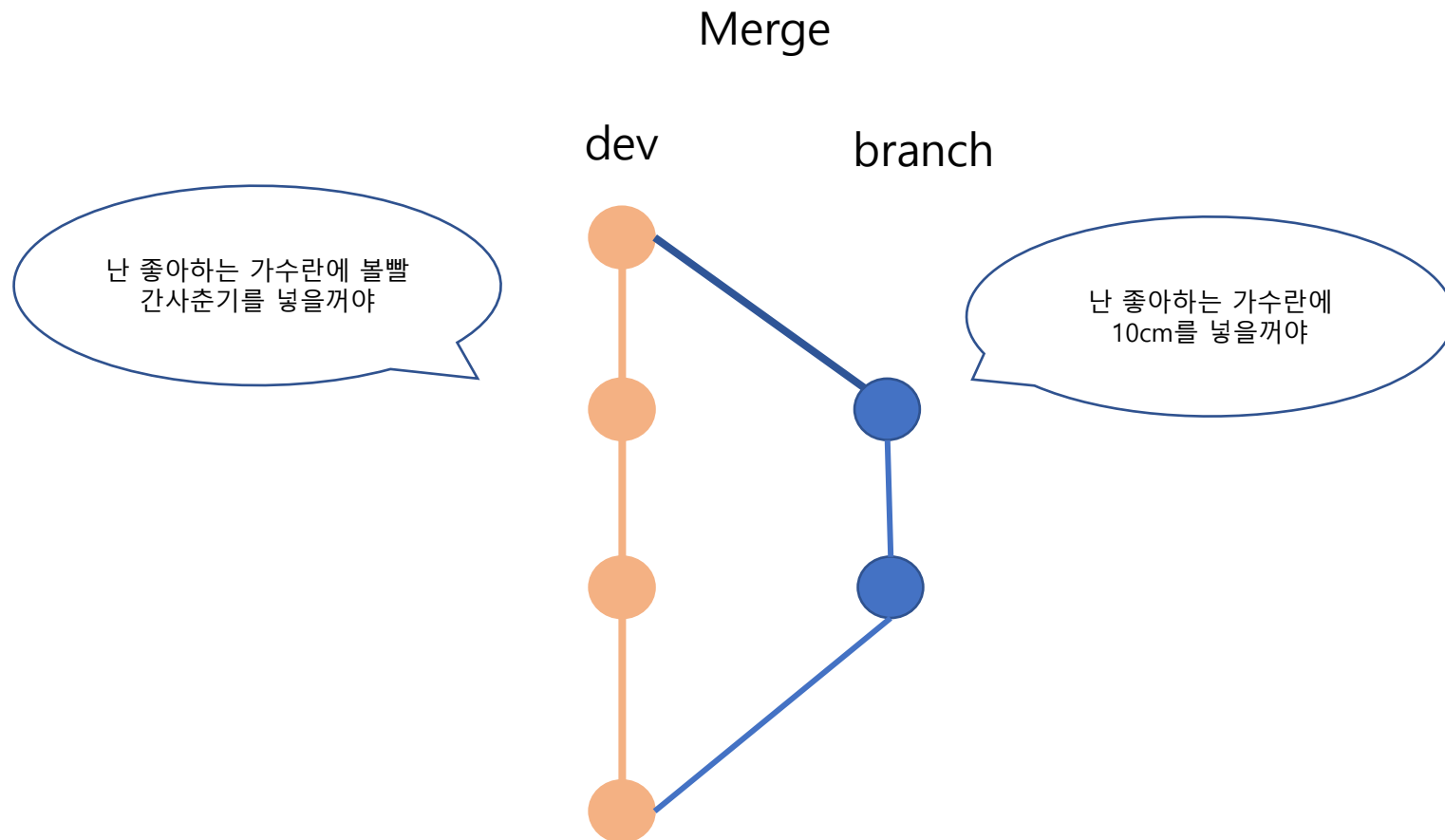
일단은 merge 쓰는거 추천 드립니다...

Chapter 7

Conflict

병합 도중에 충돌나는 경우가 있어요!

많은 커밋들 사이에 차이



병합 도중에 충돌나는 경우가 있어요!

여러분들의 커밋으로 돌아가고 하나 더 가져옵시다.

일단 기본적인 충돌 설정을 해줍시다!

```
git checkout dev
```

```
git fetch upstream "feat/conflict1"
```

```
git checkout FETCH_HEAD
```

```
git checkout -b "feat/conflict1"
```

```
git fetch upstream "feat/conflict2"
```

```
git checkout FETCH_HEAD
```

```
git checkout -b "feat/conflict2"
```

병합 도중에 충돌나는 경우가 있어요!

여러분들의 커밋으로 돌아가고 하나 더 가져옵시다.

```
git checkout -b "feat/conflict"
```

```
git merge feat/conflict1
```

```
# 첫 번째 커밋
print("첫 번째 커밋입니다.")

# 두 번째 커밋
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
while TRUE:
    print("hihihi")

# 세 번째 커밋
a = "충돌충돌충돌"
print(a + 123)

=====
if true :
    print("conflcit1")

# 세 번째 커밋
if true :
    print("conflcit1")
>>>>>> feat/conflict1 (Incoming Change)
```


병합 도중에 충돌나는 경우가 있어요!

Compare change -> 어떤 점이 다른가?

Version 1	Version 2
1 # 첫 번째 커밋	1 # 첫 번째 커밋
2 print("첫 번째 커밋입니다.")	2 print("첫 번째 커밋입니다.")
3	3
4 # 두 번째 커밋	4 # 두 번째 커밋
5- while True:	5+ if true :
6- print("hihihi")	6+ print("conflcit1")
7-	
8	7
9 # 세 번째 커밋	8 # 세 번째 커밋
10- a = "충돌충돌충돌"	9+ if true :
11- print(a + 123)	10+ print("conflcit1")
12	11

병합 도중에 충돌나는 경우가 있어요!

주어진 옵션들 (자동)

```
# 첫 번째 커밋
print("첫 번째 커밋입니다.")

# 두 번째 커밋
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
while TRUE:
    print("hihihi")

# 세 번째 커밋
a = "충돌충돌충돌"
print(a + 123)

=====
if true :
    print("conflcit1")

# 세 번째 커밋
if true :
    print("conflcit1")
>>>>>> feat/conflict1 (Incoming Change)
```

(feat/conflict2) git merge feat/conflict1

현재 브랜치

다가온 브랜치

Accept Current Change :

현재 브랜치(feat/conflict2) 기준 병합

Accept Incoming Change :

다가온 브랜치(feat/conflict1) 기준 병합

Accept Both Changes : 둘 다 사용하기

병합 도중에 충돌나는 경우가 있어요!

주어진 옵션들 (수동)

```
version.py
# 첫 번째 커밋
print("첫 번째 커밋입니다.")

# 두 번째 커밋
while True:
    print("hihihi")

# 세 번째 커밋
if true :
    print("conflcit1")
```

수동으로 필요한 부분 지우고 사용하셔도 됩니다!

병합 도중에 충돌나는 경우가 있어요!

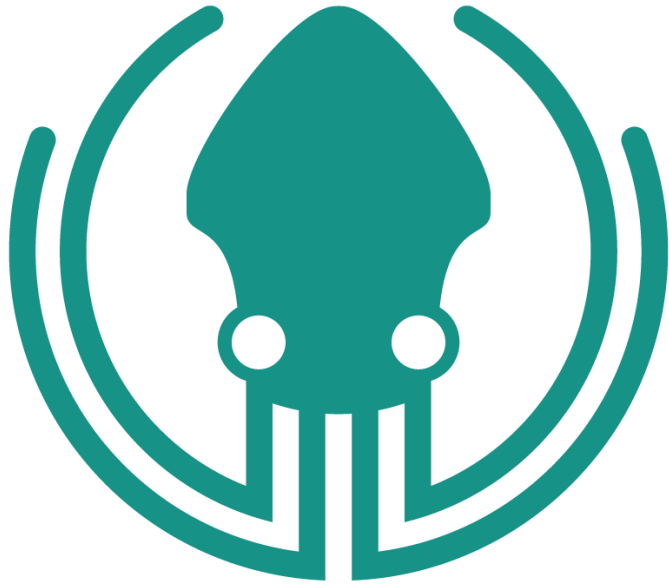
주어진 옵션들 (수동)

충돌을 해결한 상태는, 아직 커밋되지 않은 상태예요!
한번 더 커밋을 해줍니다.

```
→ LikeLion_Git_Collaborate git:(conflict) ; git add  
→ LikeLion_Git_Collaborate git:(conflict) ; git commit -m "fix : 충돌 해결"
```

CLI가 익숙하지 않으면 추천하는 프로그램

오늘 강의가 어려웠다면, 이 프로그램들을 써보면서 감을 잡는 것도 좋아요!



GitKraken



Github Desktop

과제

Session 0 과제

팀별 과제

1. 한 명이 해커톤을 위한 레포지토리 만든 다음에 팀원들 초대하기
2. Fork 이후에 pull-request 방식으로 각자의 이메일 주소 및 github 계정있는 txt 파일 제출하기.
3. 4명 다 merge 된 이후에 제출!

Merge 된 이후에 해당 레포 주소 한명이 대표하여 제출!

기한 (5월 24일까지)