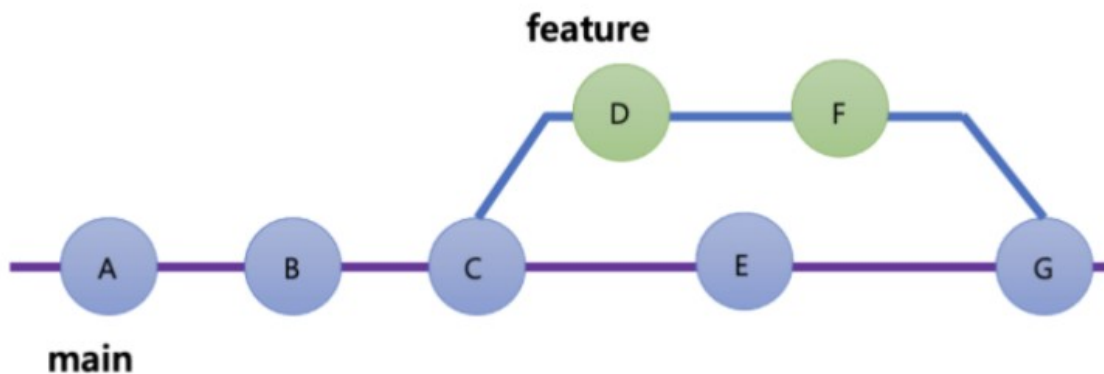


# 실습05주차\_Git(2)\_(최종)

## Git Branch

1. branch 생성 : `git branch [브랜치 이름]`
2. branch 이동 : `git checkout [브랜치 이름]`
3. branch 생성과 이동 : `git checkout -b [브랜치 이름]`
4. branch 삭제 : `git branch -d [브랜치 이름]`
5. branch 조회 : `git branch`

## Git merge



### ▼ 방법

1. main branch에서 commit A,B,C 각각 진행

```
1 # osssprac
2 fetch & merge
3 A
4 B
5 C
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL**

```
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git add README.md

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git commit -m 'A'
[main 6a49c41] A
1 file changed, 4 deletions(-)

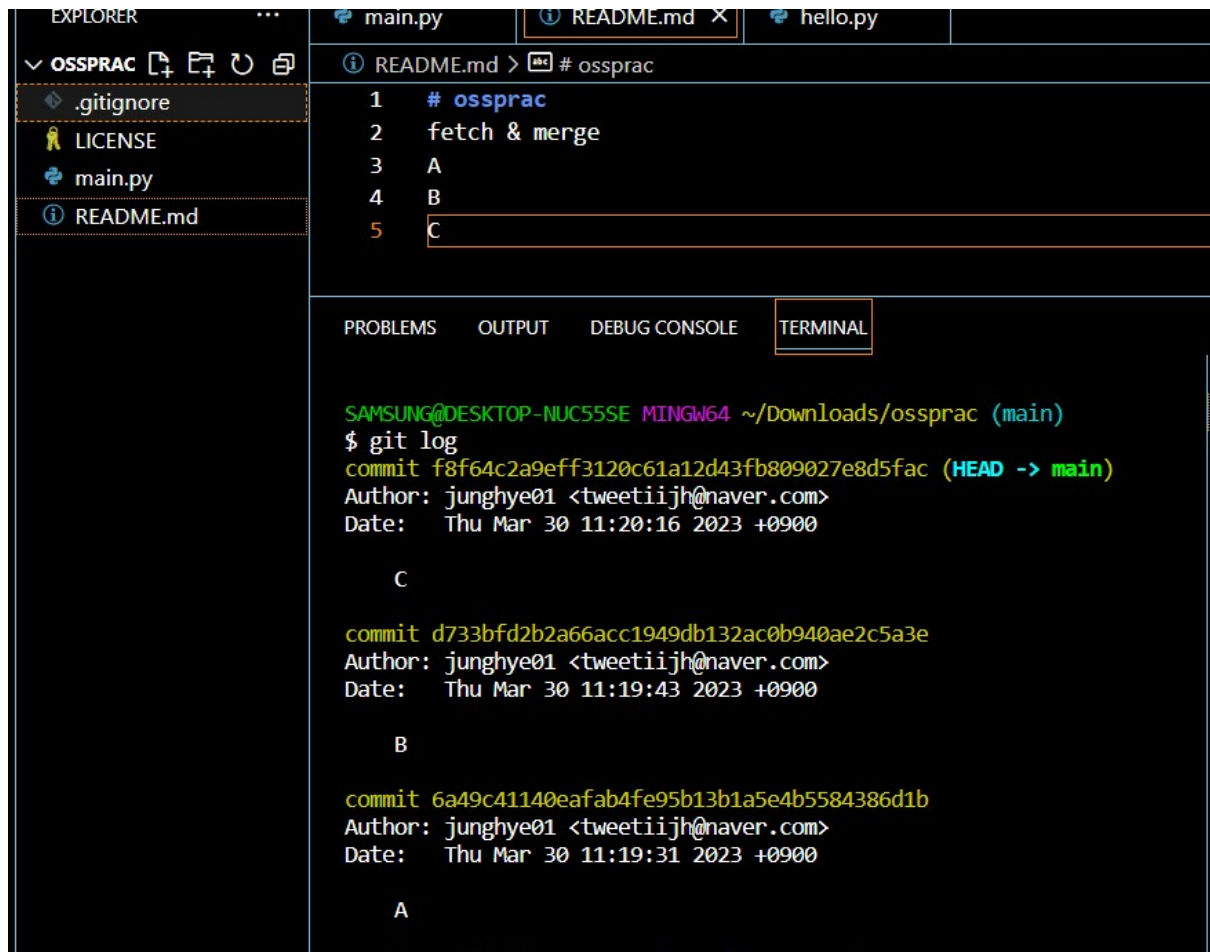
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git add README.md

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git commit -m 'B'
[main d733bfd] B
1 file changed, 1 insertion(+)

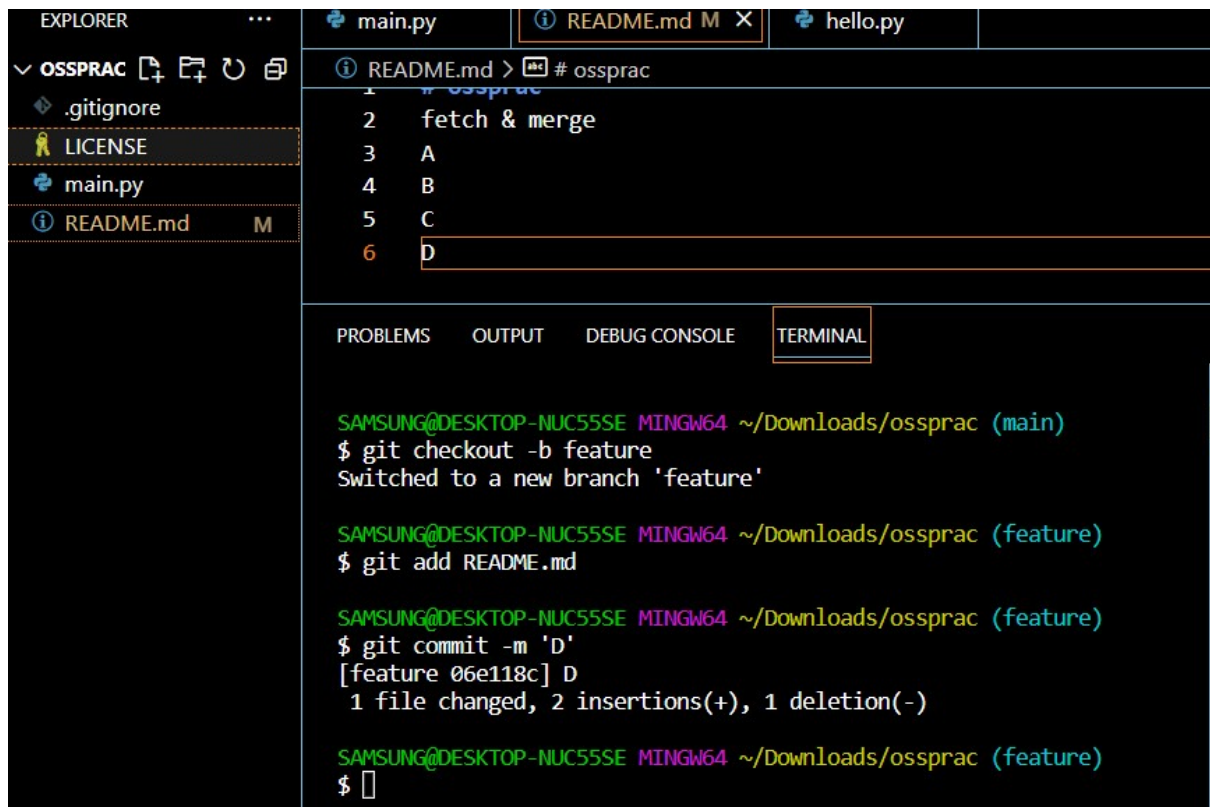
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git add README.md

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git commit -m 'C'
[main f8f64c2] C
1 file changed, 2 insertions(+), 1 deletion(-)
```

`git log` 을 통해 커밋 기록 확인



2. `git checkout -b feature`
3. feature branch에 commit 'D' 추가



#### 4. `git checkout main`

main branch에는 commit 'D'가 없음

```
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git checkout -b feature
Switched to a new branch 'feature'

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature)
$ git add README.md

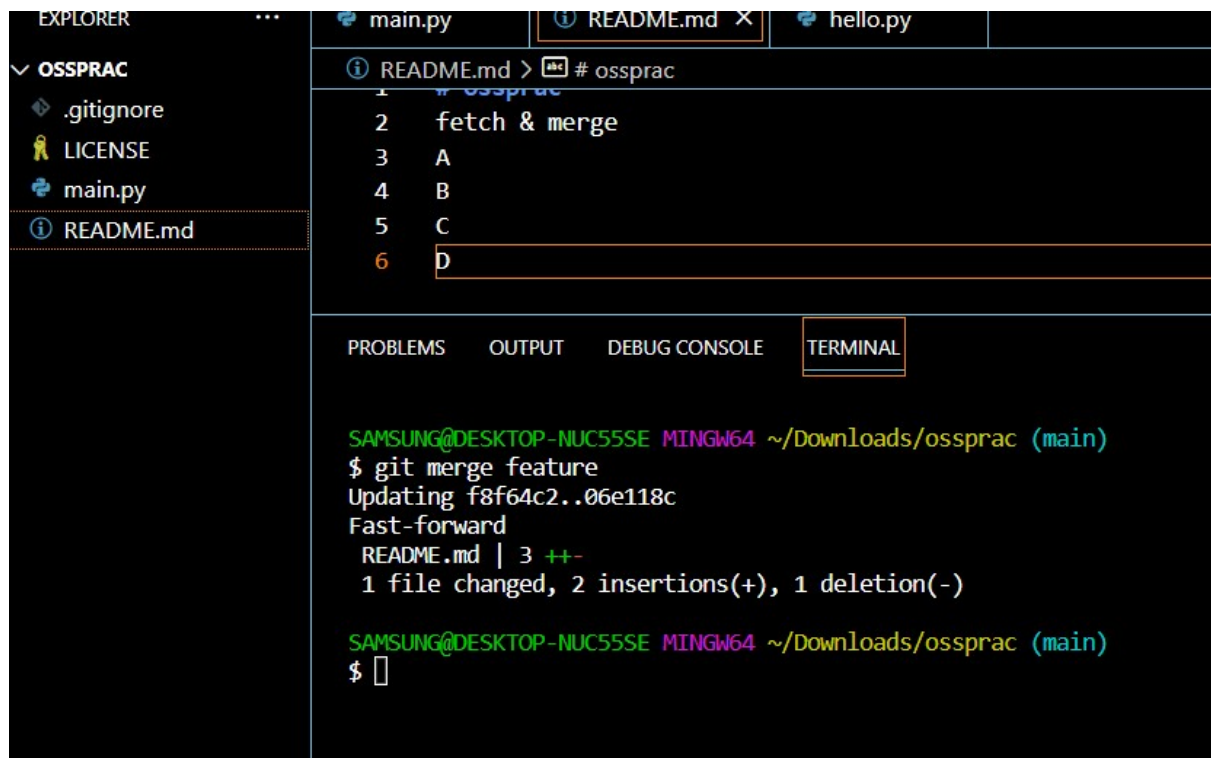
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature)
$ git commit -m 'D'
[feature 06e118c] D
1 file changed, 2 insertions(+), 1 deletion(-)

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 7 commits.
(use "git push" to publish your local commits)

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$
```

5. `git merge feature`

main branch에 commit 'D'가 생김



`git log` 을 통해 커밋 내역 확인

```
ossprac
├── .gitignore
├── LICENSE
├── main.py
└── README.md

1 # ossprac
2 fetch & merge
3 A
4 B
5 C
6 D

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git merge feature
Updating f8f64c2..06e118c
Fast-forward
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
commit 06e118c2f64b59720f0d527a0b94013340b74553 (HEAD -> main, feature)
Author: junghye01 <tweetiijh@naver.com>
Date: Thu Mar 30 11:22:07 2023 +0900

    D

commit f8f64c2a9eff3120c61a12d43fb809027e8d5fac
Author: junghye01 <tweetiijh@naver.com>
Date: Thu Mar 30 11:20:16 2023 +0900

    C

commit d733bfd2b2a66acc1949db132ac0b940ae2c5a3e
Author: junghye01 <tweetiijh@naver.com>
Date: Thu Mar 30 11:19:43 2023 +0900

    B

commit 6a49c41140eafab4fe95b13b1a5e4b5584386d1b
Author: junghye01 <tweetiijh@naver.com>
Date: Thu Mar 30 11:19:31 2023 +0900

    A
```

merge로 브랜치가 병합되고, main 브랜치의 HEAD에 새로운 커밋 이력(merge로그)가 생성됨

6. `git branch -d feature`

```
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git branch
feature
junghye
* main
```

```

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git branch -d feature
Deleted branch feature (was 9ba7308).

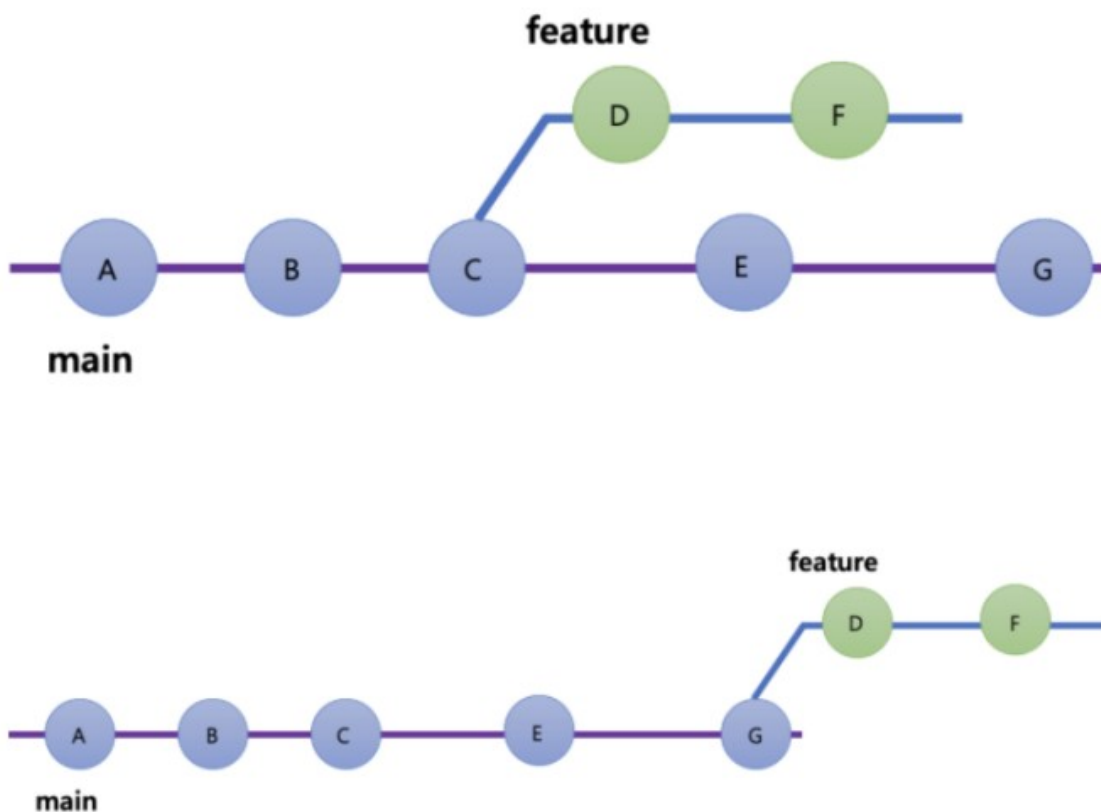
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git branch
  junghye
* main

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ 

```

## Git rebase(수정)

branch의 base를 바꾸는







## rebase와 merge의 차이

**rebase** : 브랜치의 base를 재설정하는 것 ,여기서 base는 **branch의 base**를 의미! branch는 base지점으로부터 코드를 수정함

rebase를 사용하면 merge log를 남기지 않아 git history를 더 깔끔하게 남길 수 있음

따라서 rebase를 통해 커밋이력을 재정렬한 후에 merge를 하는 방식을 추천함

- 1) 위 그림에서 C 지점을 base로 가진 feature branch가 D,F 커밋 진행
- 2) main branch에서 E,G 커밋 진행
- 3) git rebase main을 통해 main의 마지막 커밋 지점인 G로 feature branch의 base를 바꿈

**merge** : branch를 통합하고 merge 로그를 남김. 프로젝트에 참여하는 사람이 많아져서 branch 개수가 늘어나면 커밋 기록이 엉키게 됨

## ▼ 방법

1. `git checkout -b feature2` 후 `hello.py` 생성 후 commit

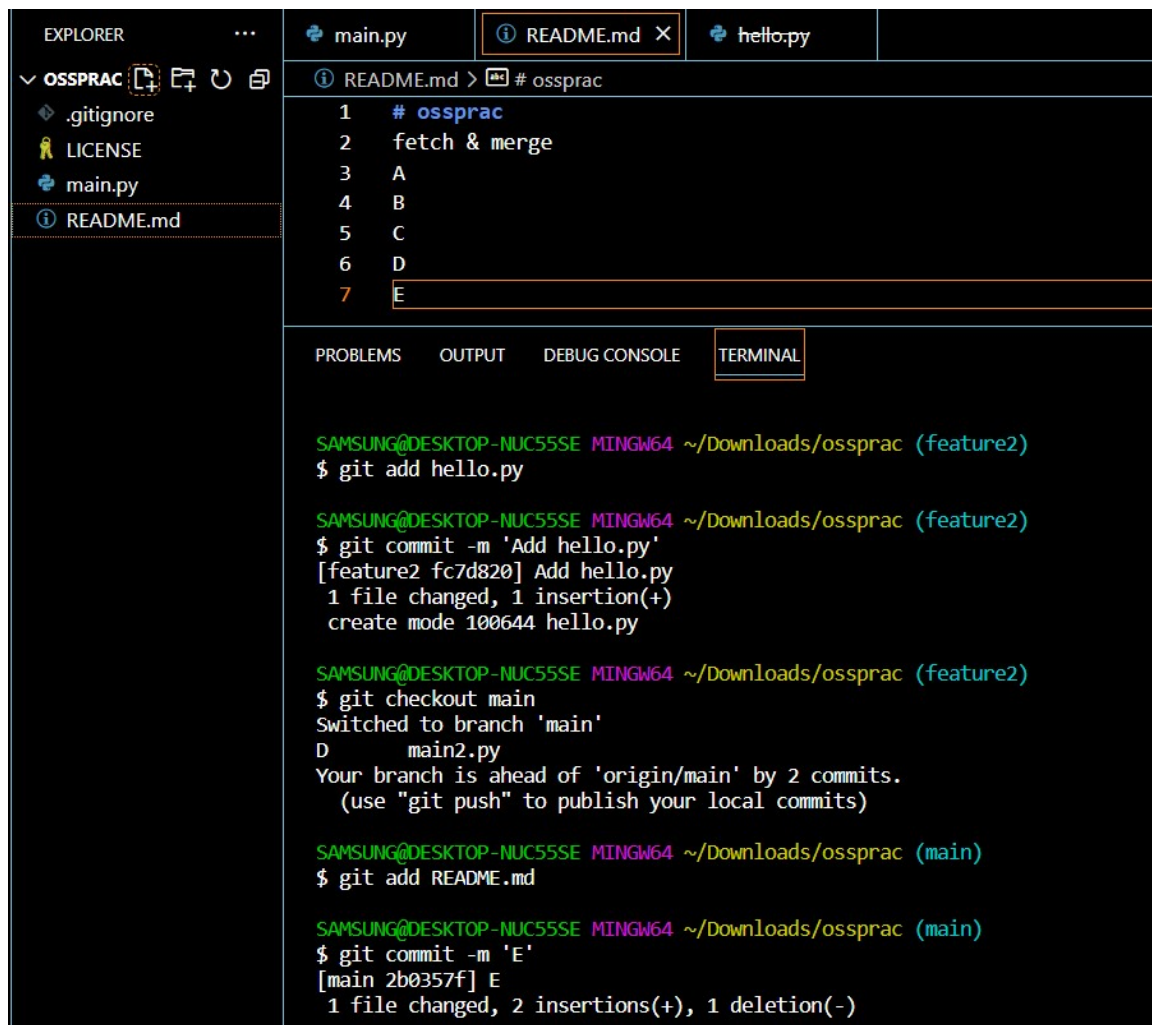
The screenshot shows a VS Code interface with a file explorer on the left and a terminal window at the bottom. The file explorer shows a project named 'OSSPRAC' with files like .gitignore, hello.py, LICENSE, main.py, and README.md. The terminal window shows the following commands and output:

```
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature2)
$ git add hello.py

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature2)
$ git commit -m 'Add hello.py'
[feature2 fc7d820] Add hello.py
1 file changed, 1 insertion(+)
create mode 100644 hello.py

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature2)
$
```

## 2. `git checkout main` 후 commit E



```
EXPLORER
  OSSPRAC
    .gitignore
    LICENSE
    main.py
    README.md

main.py | README.md | hello.py
README.md > # ossprac
1 # ossprac
2 fetch & merge
3 A
4 B
5 C
6 D
7 E

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature2)
$ git add hello.py

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature2)
$ git commit -m 'Add hello.py'
[feature2 fc7d820] Add hello.py
1 file changed, 1 insertion(+)
create mode 100644 hello.py

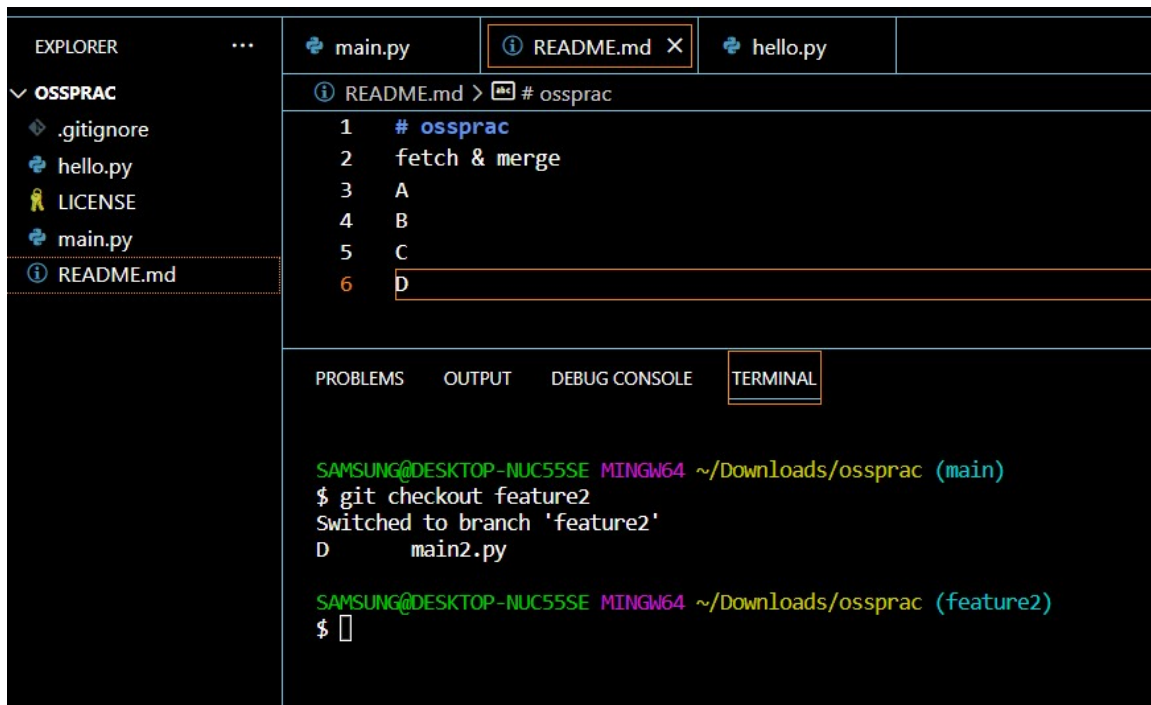
SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (feature2)
$ git checkout main
Switched to branch 'main'
D      main2.py
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git add README.md

SAMSUNG@DESKTOP-NUC55SE MINGW64 ~/Downloads/ossprac (main)
$ git commit -m 'E'
[main 2b0357f] E
1 file changed, 2 insertions(+), 1 deletion(-)
```

## 3. `git checkout feature2`

commit E가 존재하지 X



#### 4. `git rebase main`

main branch를 기준으로 rebase 시킨다는 의미, main branch의 마지막 커밋 지점인 E를 feature2 branch의 base로 재설정!

이 때 `git log` 를 하면 feature2 브랜치의 커밋 이력은 뜨지 않음, 즉 **rebase만으로 커밋 이력이 합쳐지지 않음**. 두 브랜치의 커밋 이력 및 코드를 합치고 싶으면 `git merge feature2` 까지 해야 함

#### 5. `git checkout main`

`git merge feature2`

하고 다시 `git log` 를 보면 feature2 커밋 이력들이 main branch의 커밋이력들과 함께 나타남

## Github으로 협업하기

## 협업의 큰 틀

1. 프로젝트 repo를 fork하여 fork된 저장소에서 코드를 수정한다.
2. 각자 local 환경에서 개발한 내용은 origin 에서 push한다.

origin : fork 를 통해 복사해서 my github에 생성된 repository

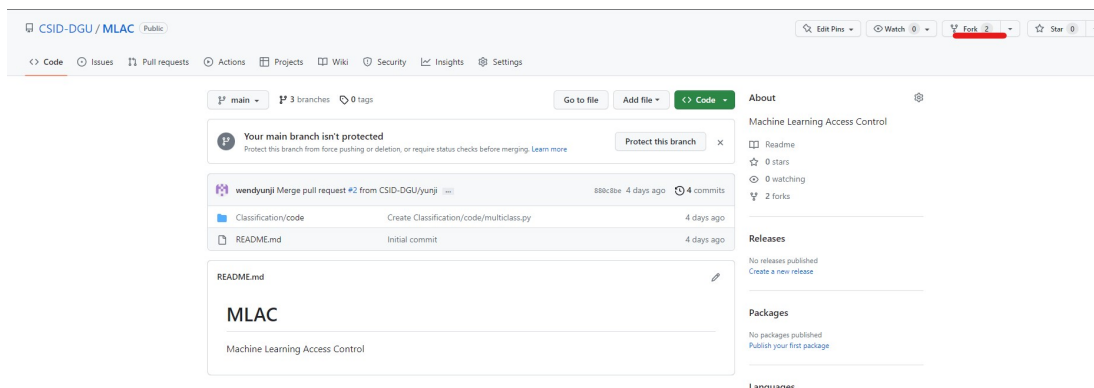
3. 해당 내용을 upstream에 pull request한다.

upstream : 가장 근본이 되는 repository

4. 코드 리뷰 후 최종 수정된 코드를 upstream에 merge 한다.
5. 변경된 upstream의 내용을 pull로 local에 반영한다.

### ▼ 방법

1. 저장소 fork



2. 코드 수정

- main 저장소가 아닌 내 계정으로 fork된 저장소를 clone받음

```
git clone https://github.com/{내계정}/{repository이름}
```

이때, 위에서 설명했던 것처럼 **master** branch에서 바로 작업하는 것이 아니라 **새로운 branch** 를 생성함

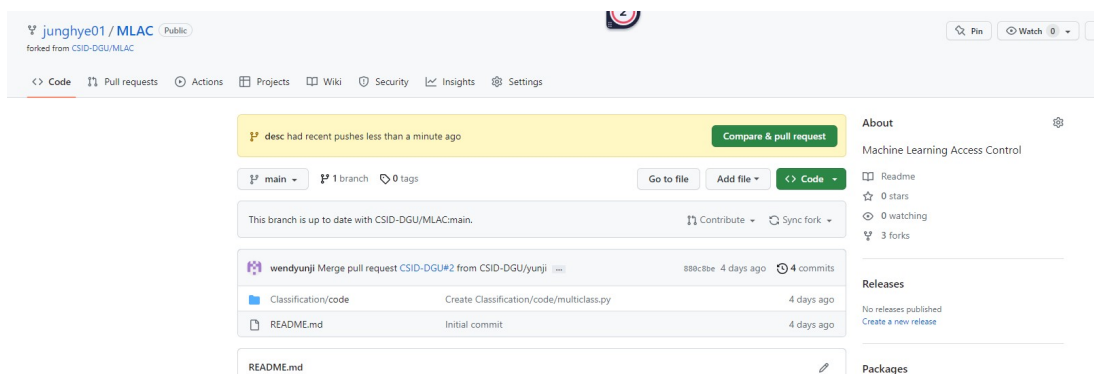
- README description을 추가하기 위해 branch이름을 **desc** 로 설정

```
irteam@ec4ec7f8b0039:~/junghye-dcloud-dir/MLAC$ git branch
* main
irteam@ec4ec7f8b0039:~/junghye-dcloud-dir/MLAC$ git checkout -b desc
Switched to a new branch 'desc'
irteam@ec4ec7f8b0039:~/junghye-dcloud-dir/MLAC$
```

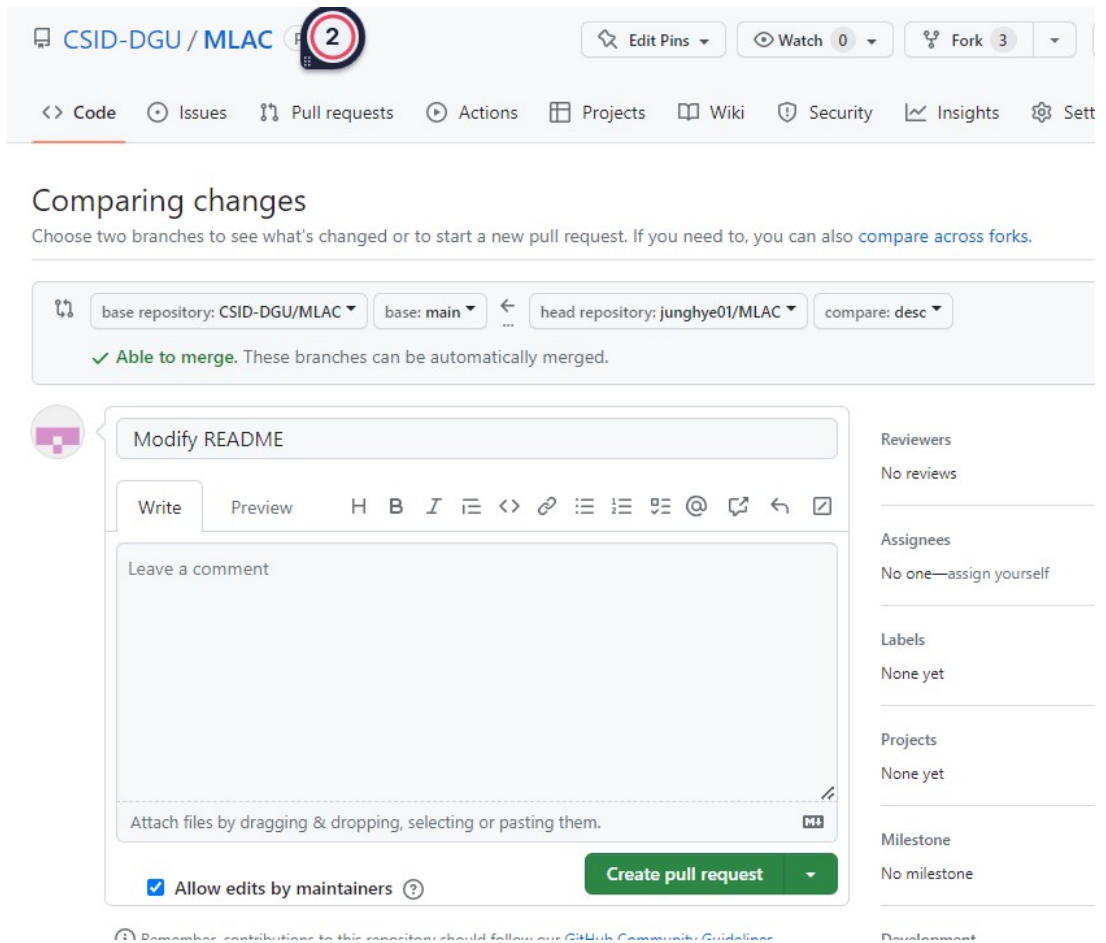
- `add`, `commit`, `push` 를 함

### 3. pull request(PR)

- repository에 가면 compare & pull request 버튼이 존재함



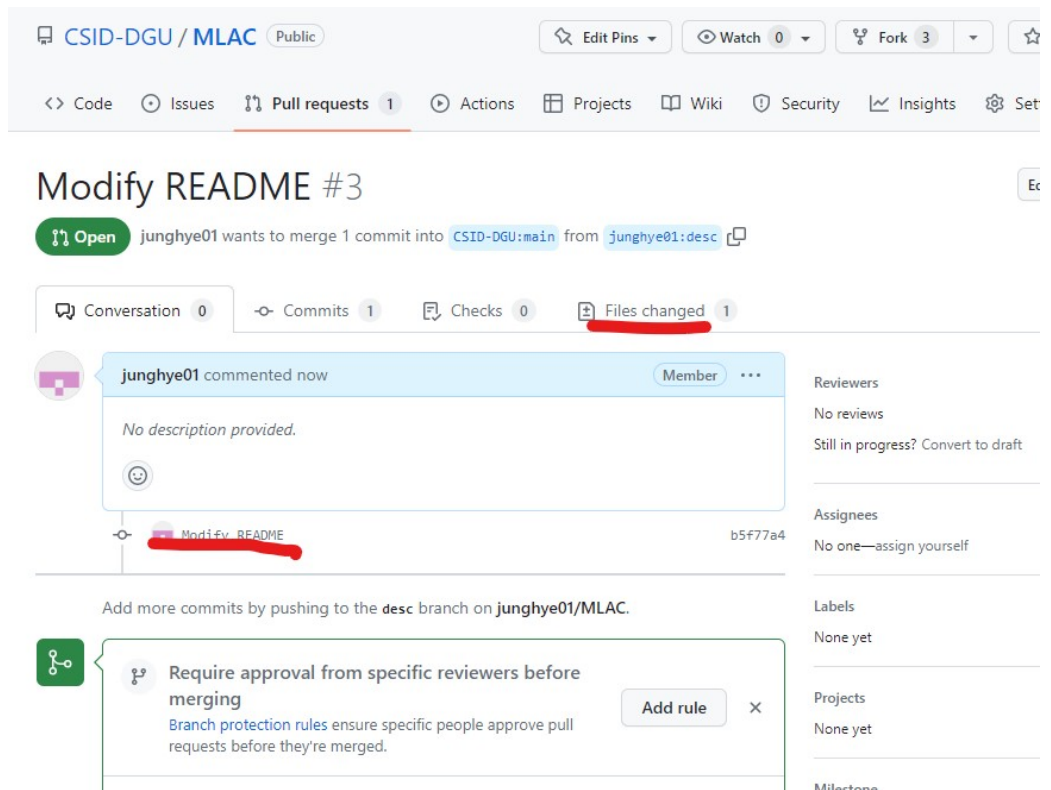
- 제목, 내용을 작성하고 create pull request를 클릭함



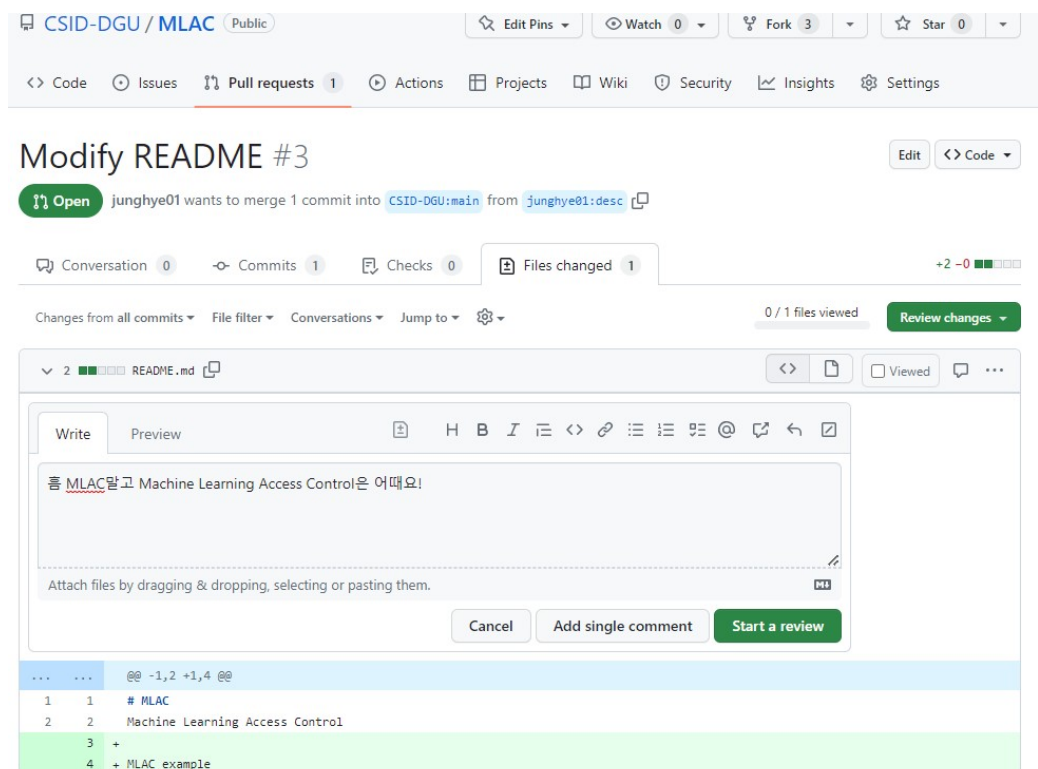
#### 4. code review

##### ▼ reviewer: 방법

- reviewer는 commit 내용을 클릭하거나 'Files changed'를 클릭함

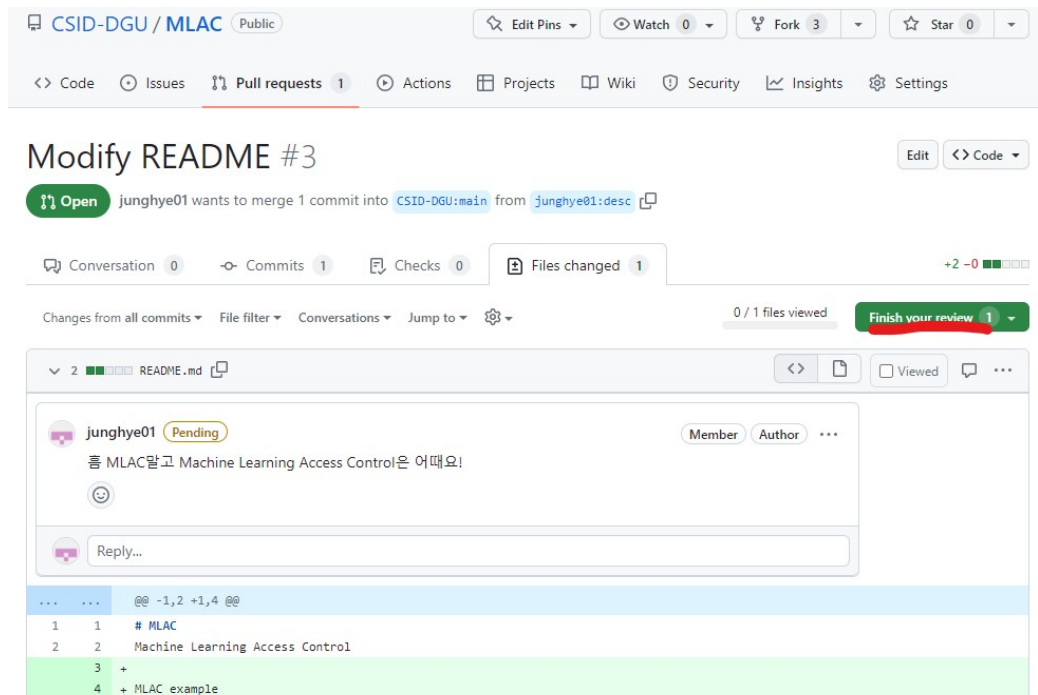


- 코드에 대한 review를 적는다





- review가 끝나면 해당 버튼을 누른다



- review가 등록됨



**Open** junghye01 wants to merge 1 commit into CSID-DGU:main from junghye01:desc

Conversation 1 | Commits 1 | Checks 0 | Files changed 1

junghye01 commented 4 minutes ago Member ...

No description provided.

Modify README b5f77a4

junghye01 commented now View reviewed changes

README.md

junghye01 now Member Author ...

흠 MLAC말고 Machine Learning Access Control은 어때요!

Reply...

Resolve conversation

Review: No review, Still in progress, Assignee: No one assigned, Labels: None, Project: None, Milestones: No milestones, Development: Success, Issues: None

## ▼ reviewee 방법

1. 리뷰 내용을 바탕으로 코드를 수정한 후 add,commit,push를 진행함

```
EXPLORER  ...  README.md X
MLAC [...  > Classification
  README.md

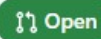
1 # MLAC
2 Machine Learning Access Control
3
4 Machine Learning Access Control example

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Delta compression using up to 64 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 337 bytes | 337.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'desc' on GitHub by visiting:
remote:   https://github.com/junghye01/MLAC/pull/new/desc
remote:
To https://github.com/junghye01/MLAC.git
* [new branch]      desc -> desc
● irteam@c4ec7f8b0039:~/junghye-dcloud-dir/MLAC$ git pull origin desc
From https://github.com/junghye01/MLAC
* branch            desc      -> FETCH_HEAD
Already up to date.
● irteam@c4ec7f8b0039:~/junghye-dcloud-dir/MLAC$ git add README.md
● irteam@c4ec7f8b0039:~/junghye-dcloud-dir/MLAC$ git commit -m 'Modify README'
[desc 9fdc9a8] Modify README
 1 file changed, 1 insertion(+), 1 deletion(-)
● irteam@c4ec7f8b0039:~/junghye-dcloud-dir/MLAC$ git push origin desc
Counting objects: 3, done.
Delta compression using up to 64 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/junghye01/MLAC.git
b55711c..9f5180c -> desc
```

2. push를 하면 기존 PR에 반영이 됨

## Modify README #3



junghye01 wants to merge 2 commits into CS10-DGU:main from junghye01:desc

Conversation 1

Commits 2

Checks 0

Files changed 1



junghye01 commented 8 minutes ago

Member ...

No description provided.



Modify README

b5f77a4



junghye01 commented 4 minutes ago

[View reviewed changes](#)

README.md



junghye01 4 minutes ago

Member Author ...

름 MLAC라고 Machine Learning Access Control은 어때요!



Reply...


Resolve conversation

Modify README

9fdc9a8

## Modify README #3

Edit <> Code

 Open junghye01 wants to merge 2 commits into CSID-DGU:main from junghye01:desc

Conversation 1

Commits 2

Checks 0

Files changed 1

+2 -0

Changes from 1 commit File filter Conversations Jump to

Review changes

### Modify README

< Prev Next >

junghye01 committed 1 minute ago

commit 9fdc9a8f172a38fa2c5d2e2ae13b4a8efb8b2739

2 README.md

<> File Comment ...

junghye01 4 minutes ago

Member Author ...

홈 MLAC라고 Machine Learning Access Control은 어때요!



Reply...

Resolve conversation

@@ -1,4 +1,4 @@

```
1 1 # MLAC
2 2 Machine Learning Access Control
3 3
4 - MLAC example
4 + Machine Learning Access Control example
```

## 5. Merge

팀원들이 review를 마친 코드는 merge하여 upstream에 반영되게 함

Open

Modify README #3  
 junghye01 wants to merge 2 commits into `CSID-DGU:main` from `junghye01:desc`

---

Modify README
 b5f77

junghye01 commented 6 minutes ago
 [View reviewed change](#)

README.md

junghye01 6 minutes ago
 

Member Author ...

홍 MLAC 말고 Machine Learning Access Control은 어때요!

Reply...

Resolve conversation

Modify README
 9fdc9

---

Add more commits by pushing to the `desc` branch on `junghye01/MLAC`.

Require approval from specific reviewers before merging  
 Branch protection rules ensure specific people approve pull requests before they're merged.
 

Add rule X

This branch has no conflicts with the base branch  
 Merging can be performed automatically.

Merge pull request

 You can also [open this in GitHub Desktop](#) or view [command line instructions](#).