

Git과 GitHub 고급 (3)

Git 협업 (3): 브랜치 병합과 GitFlow, 그리고 GitHub Issue

주차 별 활동 계획

주차	날짜	주제
1주차	9/25 (수)	Git 기초 (1): Git의 설치, Git과 Git의 Concept 소개
2주차	10/2 (수)	Git 기초 (2): 브랜치 분기와 원격 저장소 & 자주 사용되는 Git 명령어
3주차	10/16 (수)	Git 협업 (3): 브랜치 병합과 GitFlow, 그리고 GitHub Issue
4주차	10/30 (수)	Git 협업 (4): GitHub Pull Request & Code Review
5주차	11/6 (수)	Git 심화 (5): 병합 전략 및 병합 충돌 해결 전략 (Git Rebase 명령 알아보기)
6주차	11/13 (수)	Git 심화 (6): Git의 자료구조와 실수로 삭제한 커밋 복구하기
7주차	11/20 (수)	Git 응용 (7): GitHub Actions를 활용한 자동화 워크플로 생성
8주차	11/27 (수)	Git 응용 (8): Git Submodule & Git Hook (로컬에서의 자동화)

시작하기 전에...

- 2인 1조를 이루어 팀을 구성해주세요.
 - 가능하다면 주로 사용하는 프로그래밍 언어(C/Java/Python)가 같은 사람과 팀을 결성해주세요.
 - 남은 사람 1명은 멘토와 팀이 될 것입니다.
- 팀원에게 번호를 부여할 것입니다.
 - 서로 상의해서 누가 [1번 팀원], 누가 [2번 팀원]이 될 지 결정해주세요.
 - (번호에 따라 실습에서의 역할이 아주 조금 달라짐)

[실습] Repository 생성하기

- [팀원 1]은 자신의 GitHub에 Repository를 하나 생성해주세요.
 - 레포지토리 이름은 자유입니다.
 - (작명이 어렵다면 그냥, GGA-Week3 이라고 해주세요. GGA: 'G'it and 'G'ithub 'A'dvanced)
 - 반드시 공개 레포지토리로 생성해주세요.
 - [Add a README file] 에 체크해주세요.
 - [Add .gitignore] 로는 팀에서 사용할 프로그래밍 언어를 선택해주세요.

New repository

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * Repository name *

hepheir / GGA-Week3

✓ GGA-Week3 is available.

Great repository names are short and memorable. Need inspiration? How about [effective-octo-giggle](#) ?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

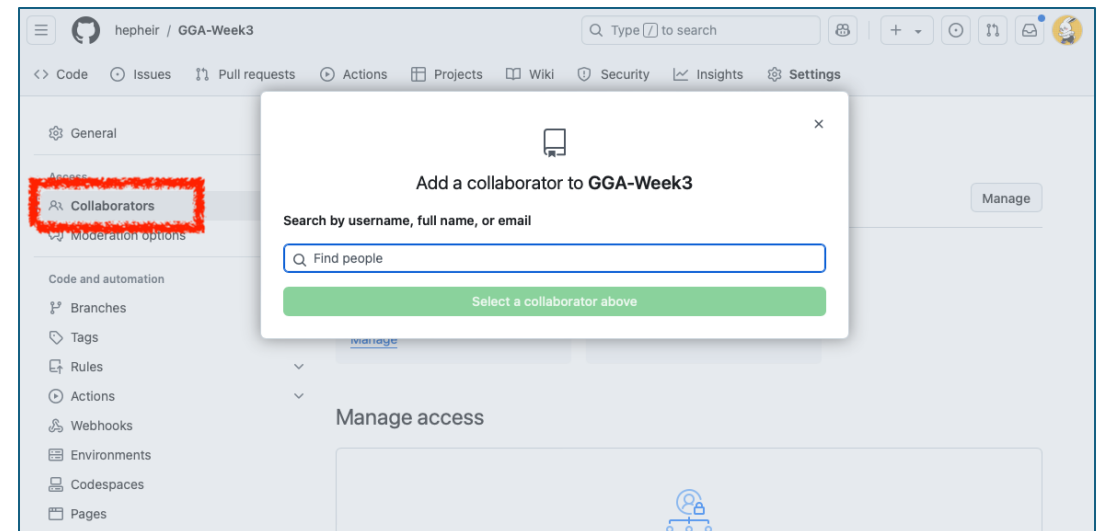
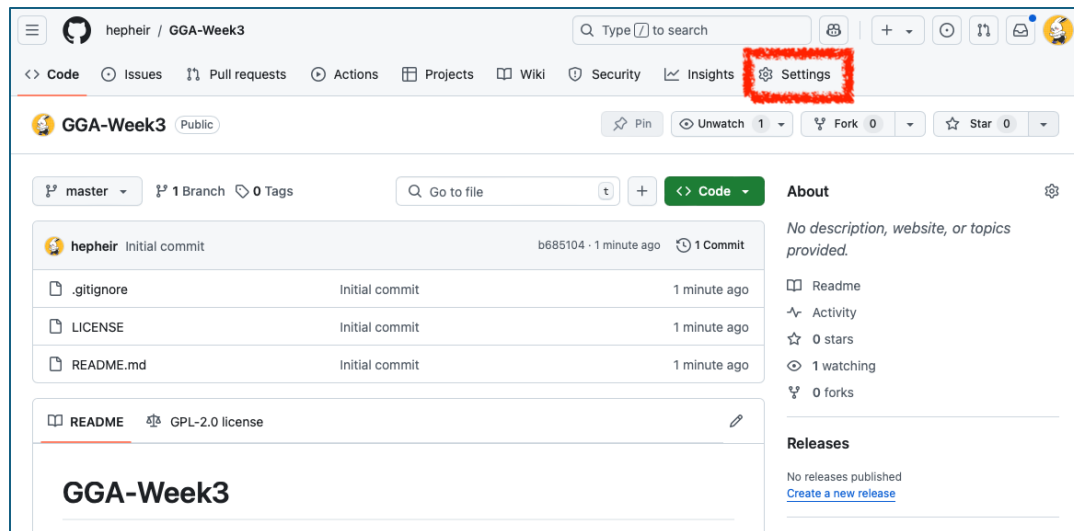
License: GNU General Public License v2.0

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set `master` as the default branch. Change the default name in your [settings](#).

[실습] Repository 에 팀원 추가하기

- [팀원 1]은 방금 생성한 레포지토리에 [팀원 2]를 추가해주세요.
 - [Settings] -> [Collaborators] -> [Add people]
 - > [팀원 2의 이메일/깃헙 사용자명으로 초대] -> [팀원 2는 이메일 확인 후 초대 수락]
 - > 완료 후 팀원 둘 다 해당 레포지토리를 Clone (반드시 clone만하고, fork는 하지 말 것)



시작하기 전에...

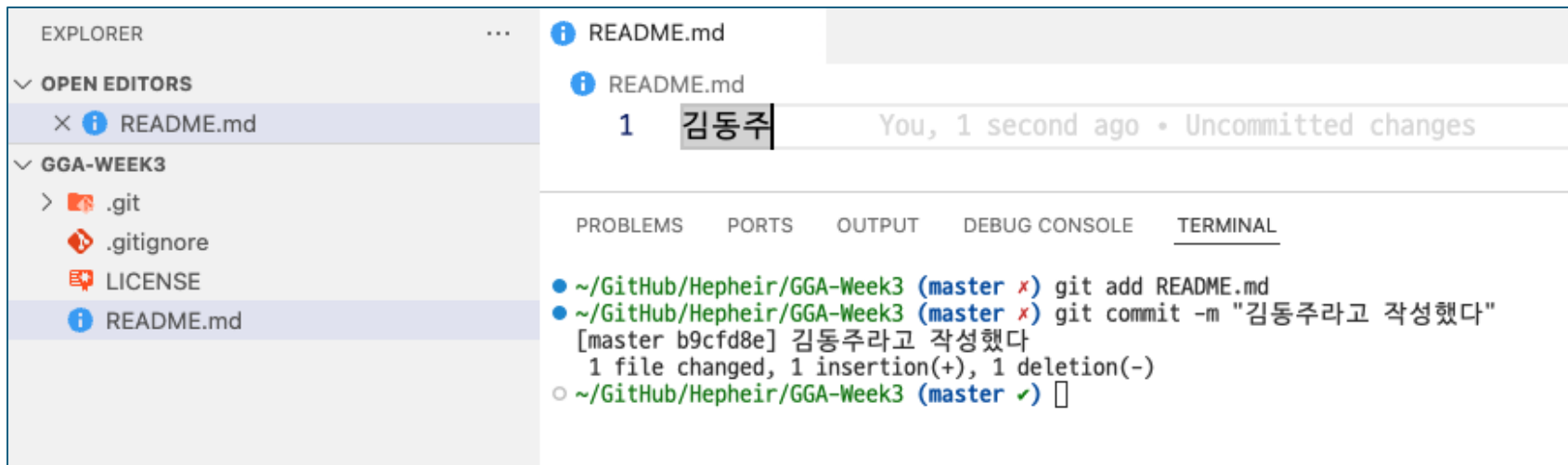
- 간단한 명령어 몇 가지만 공부해봅시다.
 - ``git add <FILE> ...``
 - Modified 혹은 Untracked 상태인 <FILE>을 Stage 한다.
 - ``git commit -m <message>``
 - <message>를 내용으로 하는 커밋을 생성.
 - ``git commit -am <message>``
 - Tracked 상태인 파일을 모두 Stage 한 뒤, <message>를 내용으로 하는 커밋을 생성. (git add + git commit)
 - ``git reset --hard HEAD~1``
 - HEAD 를 1칸 이전 커밋으로 옮김. (이전 커밋으로 되돌리기)
 - ``--hard`` 는 옵션으로, 이전 커밋으로 되돌리는 과정에서, Modified 상태인 것들은 Unmodified 상태로 되돌림 (단, Untracked는 되돌려지지 않고 유지됨.)

시작하기 전에...

- 간단한 명령어 몇 가지만 공부해봅시다.
 - ``git log --oneline``
 - 지금까지 작성한 커밋 이력을 볼 수 있다.
 - ``git checkout <branch_name>``
 - `<branch_name>`에 해당하는 브랜치로 이동.
 - ``git branch <branch_name>``
 - `<branch_name>`을 이름으로 갖는 새로운 브랜치를 생성.
 - ``git push [--set-upstream <remote_name>] <branch_name>``
 - 현재 있는 브랜치를 push하면 기본으로 `<remote_name>` 원격 저장소에 push되도록 지정하고, `<remote_name>`의 `<branch_name>` 브랜치로 push 한다.
 - 대괄호 부분은 생략이 가능하다. (최초 1회는 생략이 안될 수도 있다.)

[실습] README.md 수정하기

1. IDE, 혹은 에디터(VSCode)에서 클론한 저장소 폴더를 열고
2. README.md 파일의 모든 내용을 지운 후
3. 자신의 이름을 적어주세요.
4. Commit을 하나 생성한 후, 대기 해주세요. (아직 push는 하면 안됩니다!)

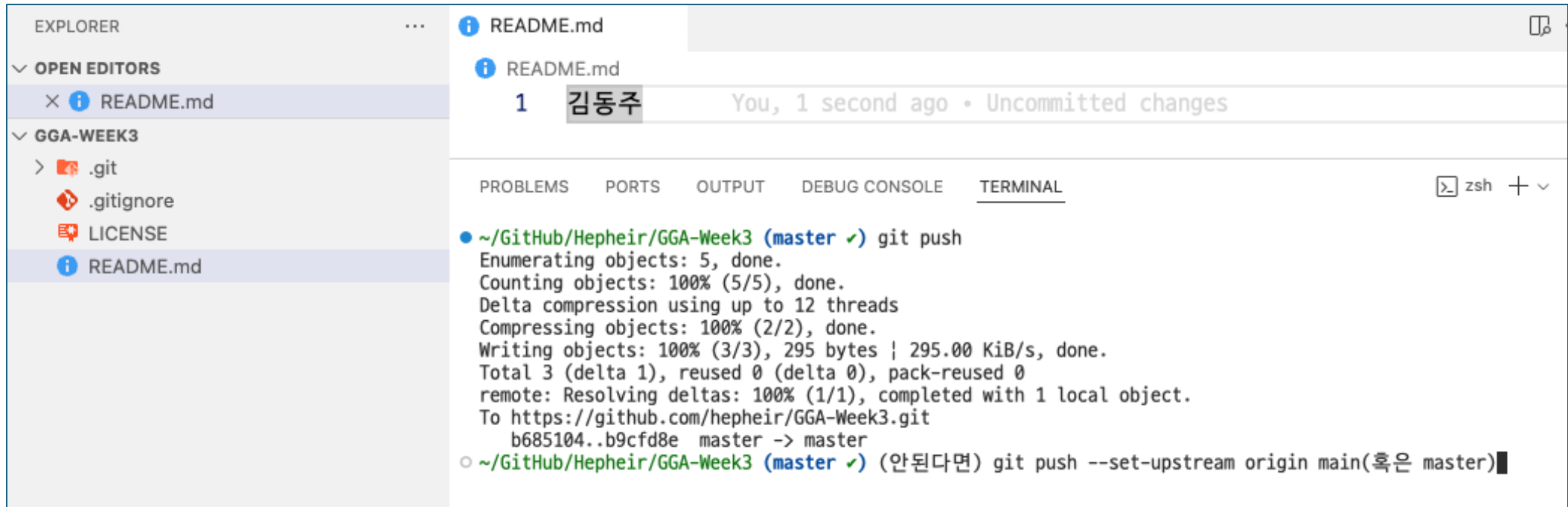


The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows the project structure with 'GGA-WEEK3' containing '.git', '.gitignore', 'LICENSE', and 'README.md'. The 'README.md' file is open in the editor. The editor shows the first line of the file with the text '김동주' (Kim Dongju) entered. A status bar at the top of the editor indicates 'You, 1 second ago • Uncommitted changes'. Below the editor, the TERMINAL panel is active, showing the following commands and output:

```
~/GitHub/Hepheir/GGA-Week3 (master x) git add README.md
~/GitHub/Hepheir/GGA-Week3 (master x) git commit -m "김동주라고 작성했다"
[master b9cfd8e] 김동주라고 작성했다
1 file changed, 1 insertion(+), 1 deletion(-)
~/GitHub/Hepheir/GGA-Week3 (master ✓) █
```


[실습] 팀원 2 먼저 push

1. [팀원 2]는 방금 변경한 내용을 `git push` 해주세요.
2. 브라우저로 GitHub에 접속하여 변경사항이 제대로 올라갔는지 확인.

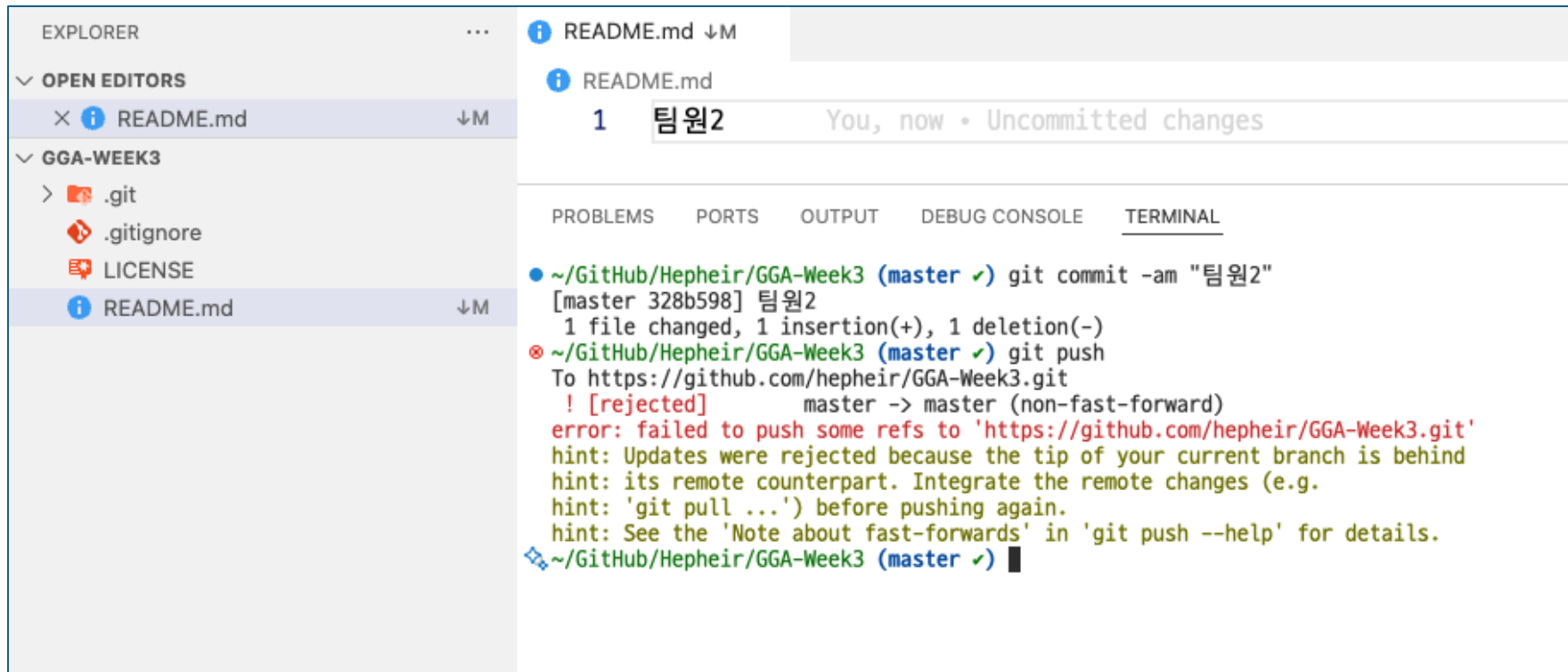


The screenshot shows the Visual Studio Code interface. On the left, the Explorer panel shows the file structure of a project named 'GGA-WEEK3', with 'README.md' selected. The Open Editors panel also shows 'README.md'. The main editor area displays the content of 'README.md', which includes the name '김동주'. Below the editor, the TERMINAL panel is active, showing the output of a 'git push' command. The output indicates that the push was successful, with objects enumerated, counted, and compressed, and the remote repository updated.

```
~/.GitHub/Hepheir/GGA-Week3 (master ✓) git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 295 bytes | 295.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/hepheir/GGA-Week3.git
   b685104..b9cfd8e  master -> master
○ ~/.GitHub/Hepheir/GGA-Week3 (master ✓) (안된다면) git push --set-upstream origin main(혹은 master)
```

[실습] 이번엔 팀원 1이 push

1. [팀원 1]도 마찬가지로 방금 변경한 내용을 `git push` 해주세요.
2. Push는 아래와 같은 메시지를 띄우며 실패할 것입니다.
(rejected: 거부 되었다는 뜻)



The screenshot shows the VS Code interface with the Explorer, Open Editors, and Terminal panels. The Explorer panel shows the project structure for GGA-WEEK3, including .git, .gitignore, LICENSE, and README.md. The Open Editors panel shows README.md with 1 line of code: 팀원2. The Terminal panel shows the output of a git commit and a git push command. The git push command failed with a 'rejected' message, indicating that the push was rejected because the tip of the current branch is behind its remote counterpart. The message suggests integrating remote changes (e.g., 'git pull ...') before pushing again.

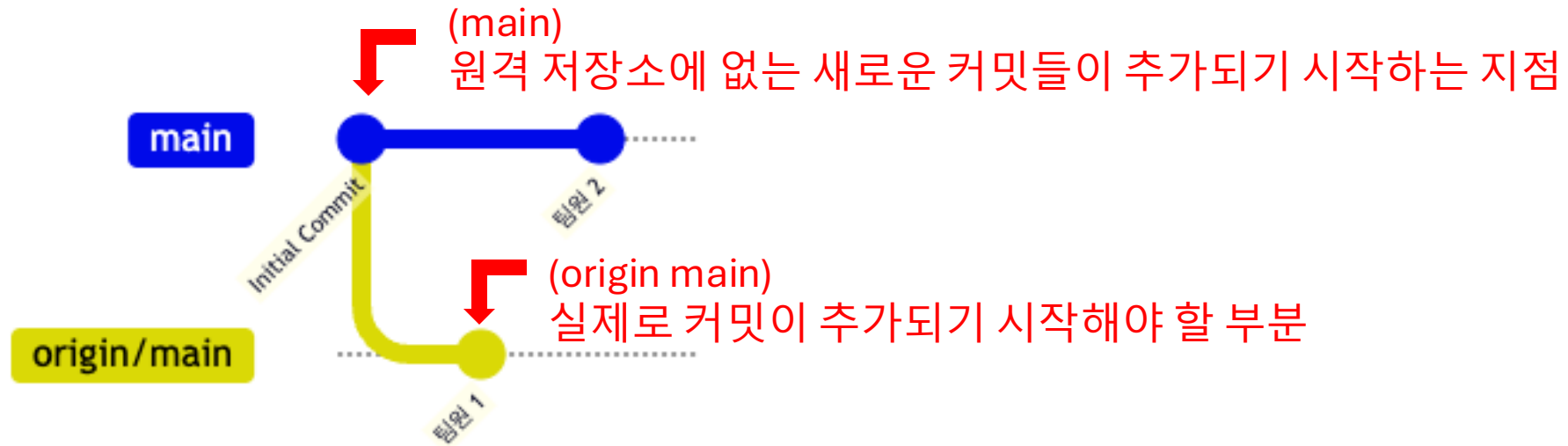
```
~ /GitHub/Hepheir/GGA-Week3 (master ✓) git commit -am "팀원2"
[master 328b598] 팀원2
1 file changed, 1 insertion(+), 1 deletion(-)
~ /GitHub/Hepheir/GGA-Week3 (master ✓) git push
To https://github.com/hepheir/GGA-Week3.git
! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/hepheir/GGA-Week3.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
~ /GitHub/Hepheir/GGA-Week3 (master ✓) █
```

[실습] 실패 이유 살펴보기

```
⊗ ~/GitHub/Hepheir/GGA-Week3 (master ✓) git push
To https://github.com/hepheir/GGA-Week3.git
! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/hepheir/GGA-Week3.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
❖ ~/GitHub/Hepheir/GGA-Week3 (master ✓) █
```

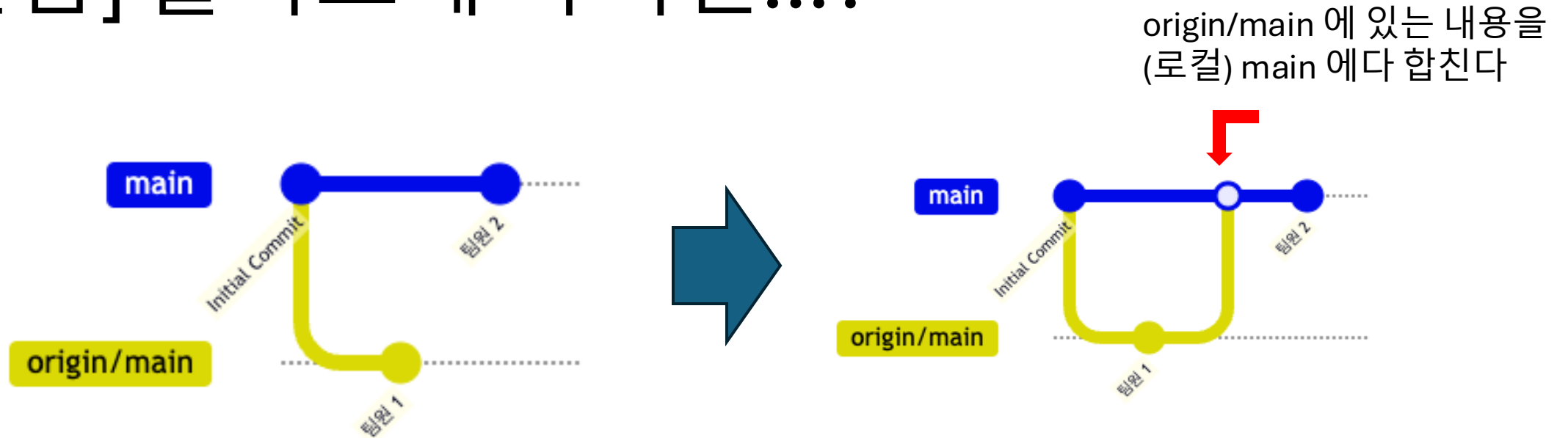
1. Tip of your current branch is behind its remote counterpart.
(지금 있는 브랜치가 가리키는 곳이, 원격 저장소에 있는 것보다 뒤쳐져있다.)
2. Integrate the remote changes ... before pushing again.
(다시 push 하기 전에 원격 저장소의 내용을 융합하라. 예) Git pull 등으로)

[실습] 실패 이유 살펴보기



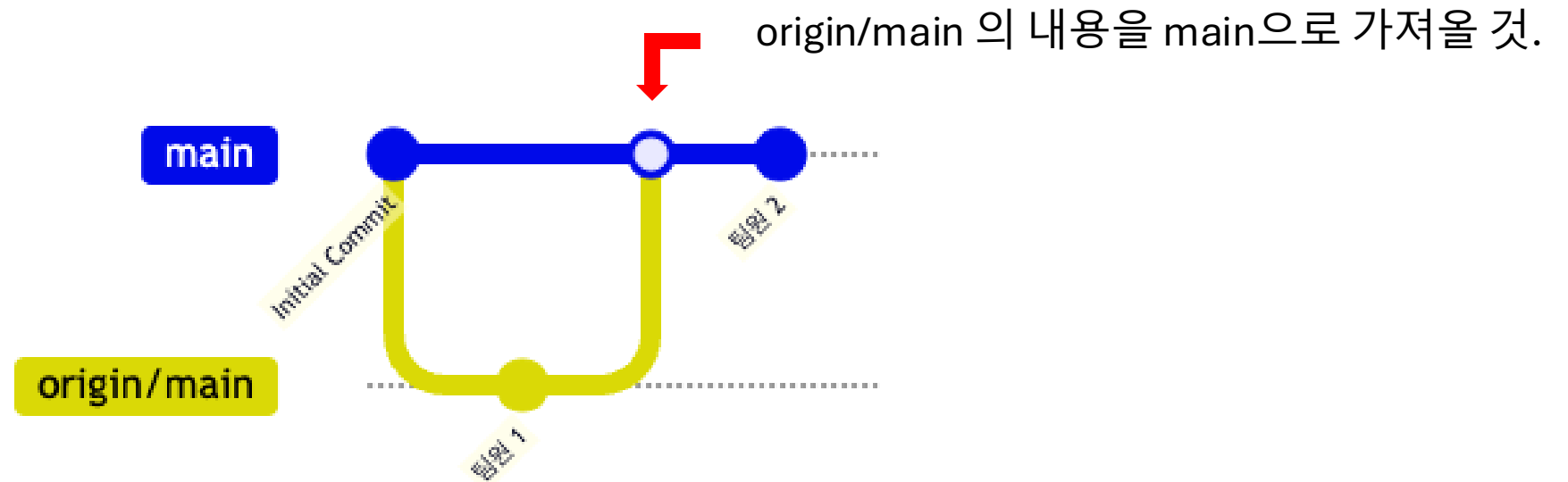
1. Tip of your current branch is behind its remote counterpart.
(지금 있는 브랜치가 가리키는 곳이, 원격 저장소에 있는 것보다 뒤쳐져있다.)
2. Integrate the remote changes ... before pushing again.
(다시 push 하기 전에 원격 저장소의 내용을 융합하라. 예) Git pull 등으로)

[실습] 올바르게 하려면...?



1. Tip of your current branch is behind its remote counterpart.
(지금 있는 브랜치가 가리키는 곳이, 원격 저장소에 있는 것보다 뒤쳐져있다.)
2. Integrate the remote changes ... before pushing again.
(다시 push 하기 전에 원격 저장소의 내용을 융합하라. 예) Git pull 등으로)

[실습] 올바르게 하려면...?

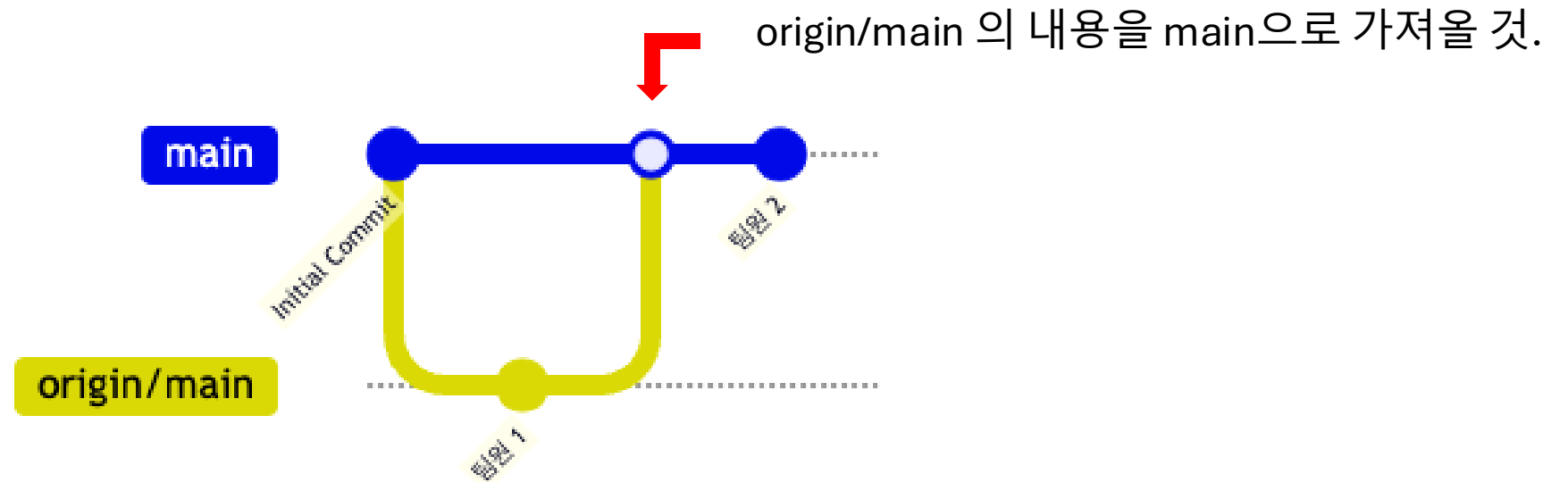


1. Current(내가 유지할 브랜치)와, Incoming(가져올 브랜치)를 정한다.
 1. 위 예시에서는 Current = main / Incoming = origin/main 이다.

[명령어] 브랜치 병합

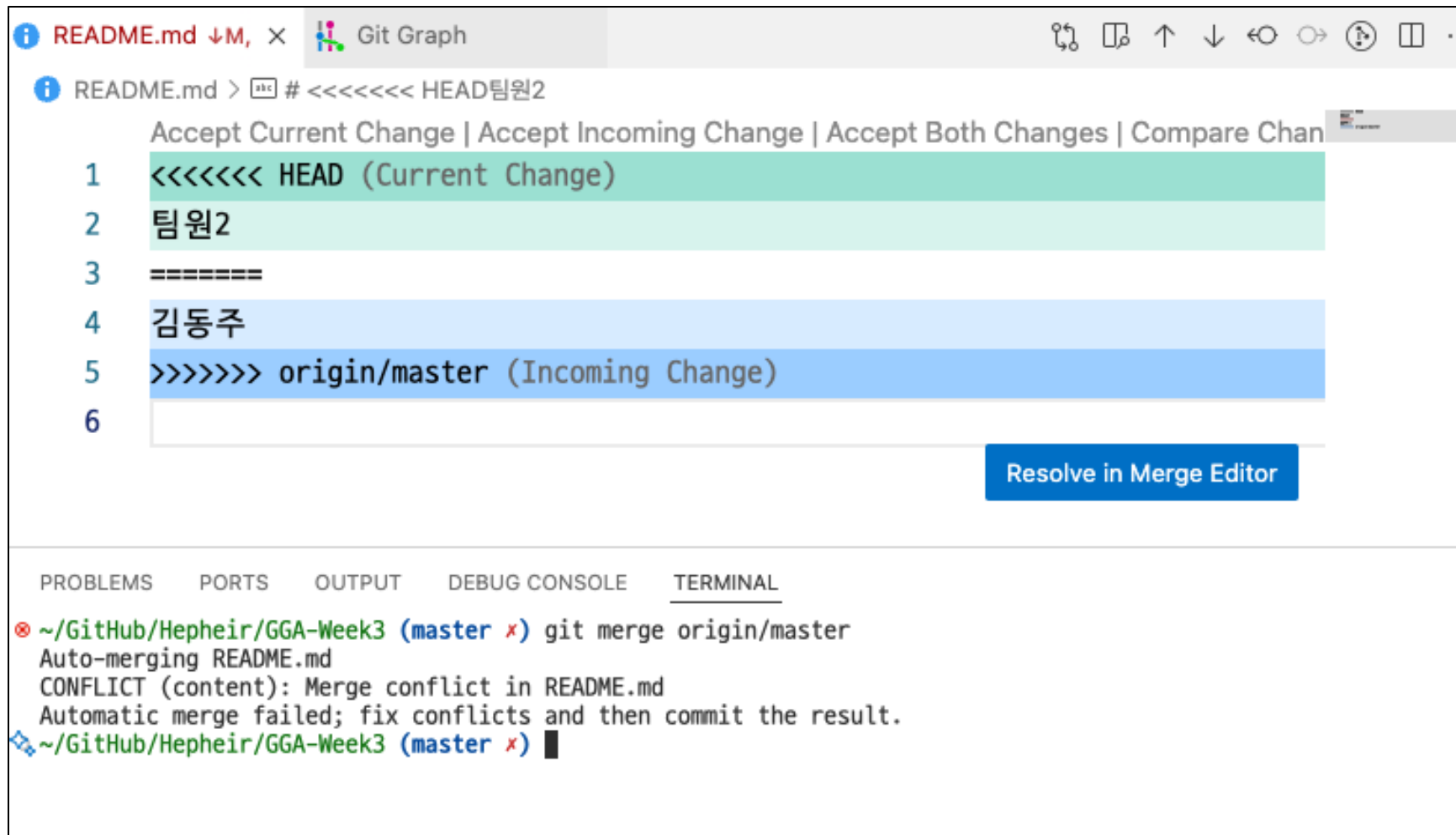
- **`git merge <branch_name>`**
 - 현재 있는 브랜치로 <branch_name> 브랜치의 변경사항을 가져와 병합한다.
- **`git merge --abort`**
 - 만약 병합과정 중에 병합 충돌이 발생하면, 이 명령을 실행하여 병합 전으로 되돌릴 수 있다.
- **`git merge --continue`**
 - 병합과정 중 발생한 병합 충돌을 해결한 뒤에 실행하는 명령어. 잠시 중단된 병합 작업을 이어서 진행한다.

[실습] 올바르게 하려면...?



1. Current(내가 유지할 브랜치)와, Incoming(가져올 브랜치)를 정한다.
 1. 위 예시에서는 Current = main / Incoming = origin/main 이다.
2. Current로 지정한 브랜치로 이동한다. (**git checkout** main)
3. Incoming 브랜치를 Current 브랜치로 병합한다. (**git merge** origin/main)

[실습] !!! 병합 충돌 (merge conflict) !!!

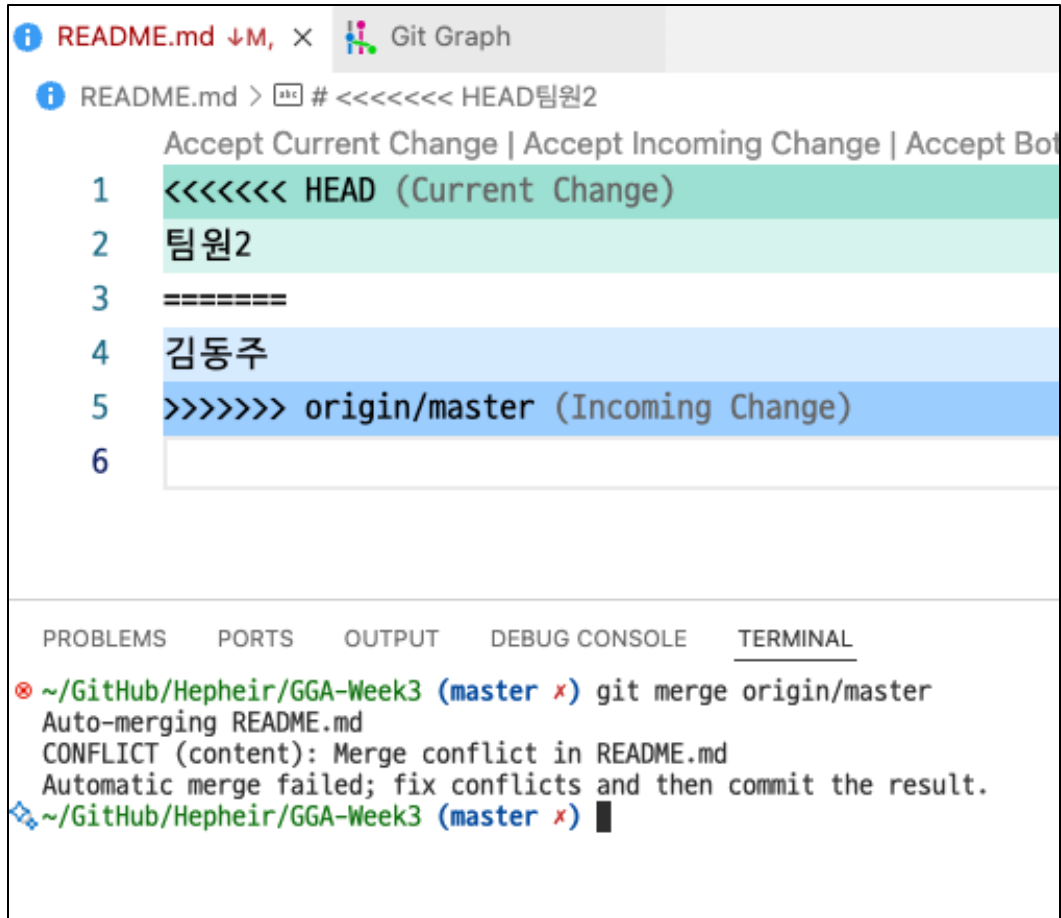


```
README.md ↓M, × Git Graph
README.md > # <<<<<<< HEAD팀원2
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Chan
1 <<<<<<< HEAD (Current Change)
2 팀원2
3 =====
4 김동주
5 >>>>>>> origin/master (Incoming Change)
6

Resolve in Merge Editor

PROBLEMS PORTS OUTPUT DEBUG CONSOLE TERMINAL
~/GitHub/Hepheir/GGA-Week3 (master x) git merge origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
~/GitHub/Hepheir/GGA-Week3 (master x) █
```

[실습] 병합 충돌이 발생하면



The screenshot shows a code editor with a merge conflict in `README.md`. The conflict is between the current branch (HEAD) and the incoming branch (origin/master). The conflict is resolved by accepting the incoming change, resulting in the text "김동주" being added to the file.

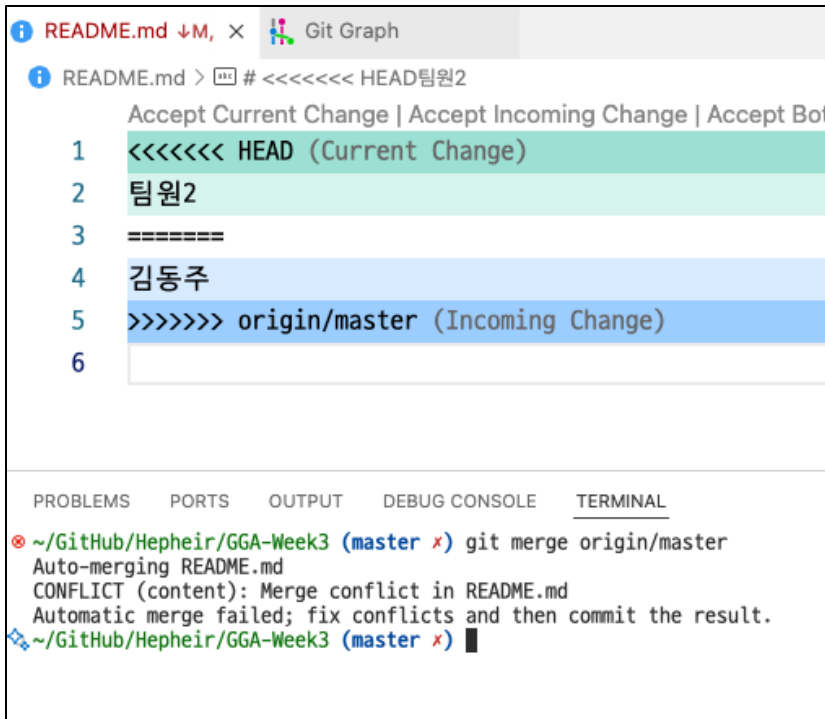
```
Accept Current Change | Accept Incoming Change | Accept Both Changes
1 <<<<<<< HEAD (Current Change)
2 팀원2
3 =====
4 김동주
5 >>>>>> origin/master (Incoming Change)
6
```

The terminal window shows the following output:

```
~/GitHub/Hepheir/GGA-Week3 (master x) git merge origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
~/GitHub/Hepheir/GGA-Week3 (master x) █
```

1. 가져올 브랜치와, 현재 브랜치의 변경사항들 중 겹치는 부분이 왼쪽과 같이 변경되어 표시된다.
2. “<<<<<< HEAD”와
~~~(현재 브랜치측 변경사항)  
“=====”  
~~~(가져올 브랜치측 변경사항)  
“>>>>>> <incoming_branch>”
형식으로 표시된다.

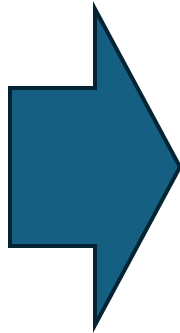
[실습] 병합 충돌 해결



```
README.md > # <<<<<<< HEAD팀원2
Accept Current Change | Accept Incoming Change | Accept Bot
1 <<<<<<< HEAD (Current Change)
2 팀원2
3 =====
4 김동주
5 >>>>>>> origin/master (Incoming Change)
6
```

PROBLEMS PORTS OUTPUT DEBUG CONSOLE TERMINAL

```
~/GitHub/Hepheir/GGA-Week3 (master x) git merge origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
~/GitHub/Hepheir/GGA-Week3 (master x) git add README.md
~/GitHub/Hepheir/GGA-Week3 (master x) git merge --continue
```



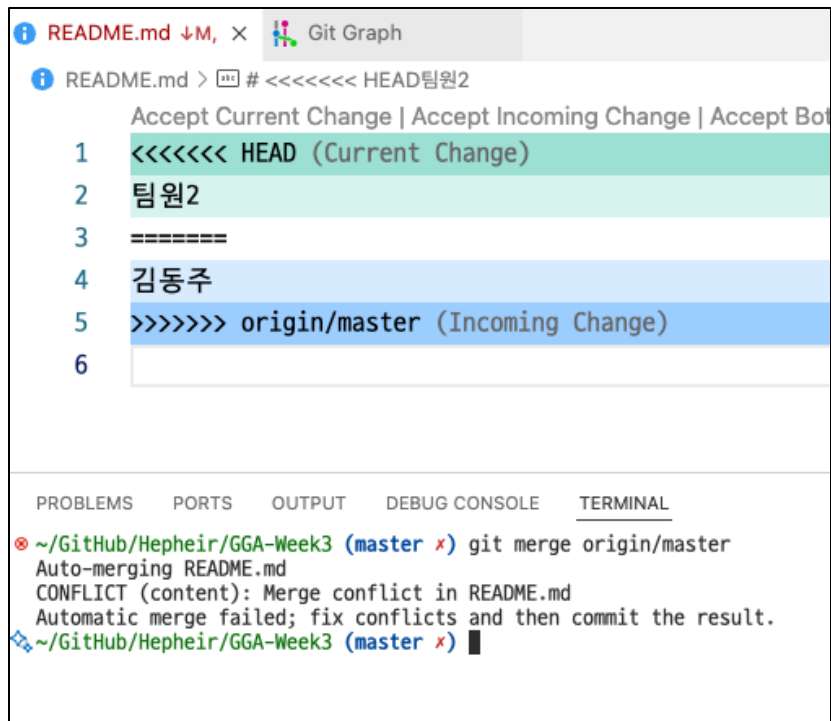
```
README.md
1 김동주
2 팀원2 You, 28 minutes ago • 팀원2
```

PROBLEMS PORTS OUTPUT DEBUG CONSOLE TERMINAL

```
~/GitHub/Hepheir/GGA-Week3 (master x) git merge origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
~/GitHub/Hepheir/GGA-Week3 (master x) git add README.md
~/GitHub/Hepheir/GGA-Week3 (master x) git merge --continue
```

1. 충돌한 부분의 내용을 알맞게 편집한 후, git add 로 수정한 파일을 stage 한다.
2. `git merge --continue` 명령을 입력하여 병합 작업을 이어나간다.

[리뷰] 병합 충돌은



```
README.md > # <<<<<<< HEAD팀원2
Accept Current Change | Accept Incoming Change | Accept Bot
1 <<<<<<< HEAD (Current Change)
2 팀원2
3 =====
4 김동주
5 >>>>>>> origin/master (Incoming Change)
6

PROBLEMS  PORTS  OUTPUT  DEBUG CONSOLE  TERMINAL
~/GitHub/Hepheir/GGA-Week3 (master x) git merge origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
~/GitHub/Hepheir/GGA-Week3 (master x) █
```

1. 병합 충돌은 서로 다른 브랜치에서 '같은 파일을 수정'할 때 주로 발생한다.
2. 항상 발생하는 것은 아니고,
 - * 파일 내에서도 변경한 부분이 겹칠 때,
 - * 혹은 겹치는지 여부를 판단하기 모호할 때 발생한다.

(정확히는 git의 diff로 보았을 때 수정한 라인 번호에 공통된 부분이 있으면 발생)

3. 병합 충돌이 발생한 부분(<<<<< ===== >>>>>)은 여러 개일 수도 있다.

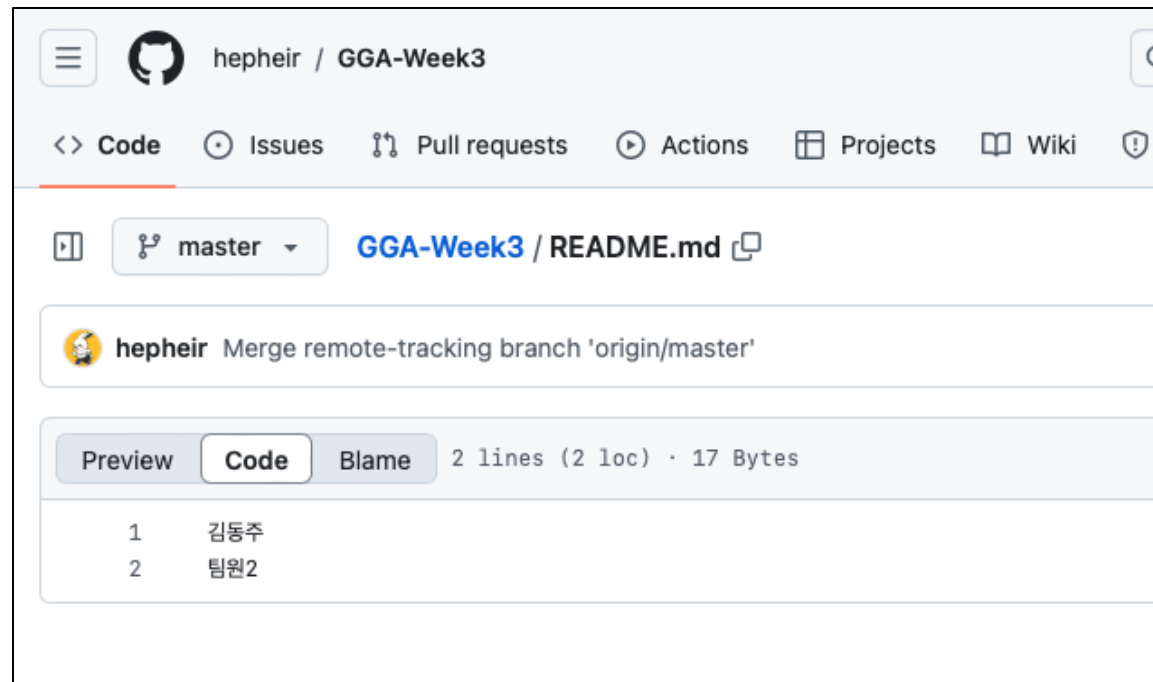
[실습] 다시 팀원 1이 push

1. 병합 충돌을 해결했다면, [팀원 1]은 해당 브랜치를 다시 Push해주세요.
2. 브라우저로 GitHub에 접속하여 변경사항이 제대로 올라갔는지 확인.

```
1 README.md
2 김동주
3 팀원2 You, 28 minutes ago • 팀원2
```

PROBLEMS PORTS OUTPUT DEBUG CONSOLE TERMINAL

```
~/GitHub/Hepheir/GGA-Week3 (master x) git merge origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
~/GitHub/Hepheir/GGA-Week3 (master x) git add README.md
~/GitHub/Hepheir/GGA-Week3 (master x) git merge --continue
[master 0428091] Merge remote-tracking branch 'origin/master'
~/GitHub/Hepheir/GGA-Week3 (master x) git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 544 bytes | 544.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/hepheir/GGA-Week3.git
b9cfd8e..0428091 master -> master
~/GitHub/Hepheir/GGA-Week3 (master x) █
```



Review

- 여러명의 사람들이 서로 다른 환경에서 변경사항을 만들고, 동일한 원격 저장소(remote)로 Push를 시도하면 실패할 수 있다.
- Merge Conflict (병합 충돌)은 언제 발생하는가?
- 병합 충돌을 해결하는 방법, 사용해야하는 명령어들
- 병합 충돌을 피하려면?