

# github으로 협업하기

김형준

# git commit, push, pull

이번 실습은 팀으로 진행하며, 각자 역할이 있습니다.

팀원들끼리 1번 2번 3번 역할을 분배하세요

원격 저장소



팀원 1



팀원 2



팀원 3

# git commit, push, pull

git commit: 커밋을 남기는 명령어. 커밋은 작업 내

git push: 내 로컬 환경에서 남긴 커밋을 원격 저장소에 업로드하는 명령어

git pull: 원격 저장소에 남은 커밋을 내 로컬 환경에 땡겨오는 명령어


# git commit, push, pull

1번 팀원분이 협업을 진행할 github repository를 만들어주세요.

repo 이름을 설정해주고, private와 add readme 옵션을 선택해서 만들어준 뒤, 팀원들을 초대하세요

---

Owner \*

 yeomko22 ▾

Repository name \*

github\_practice


✓ github\_practice is available.

Great repository names are short and memorable. Need inspiration? How about **cautious-adventure** ?


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.

---

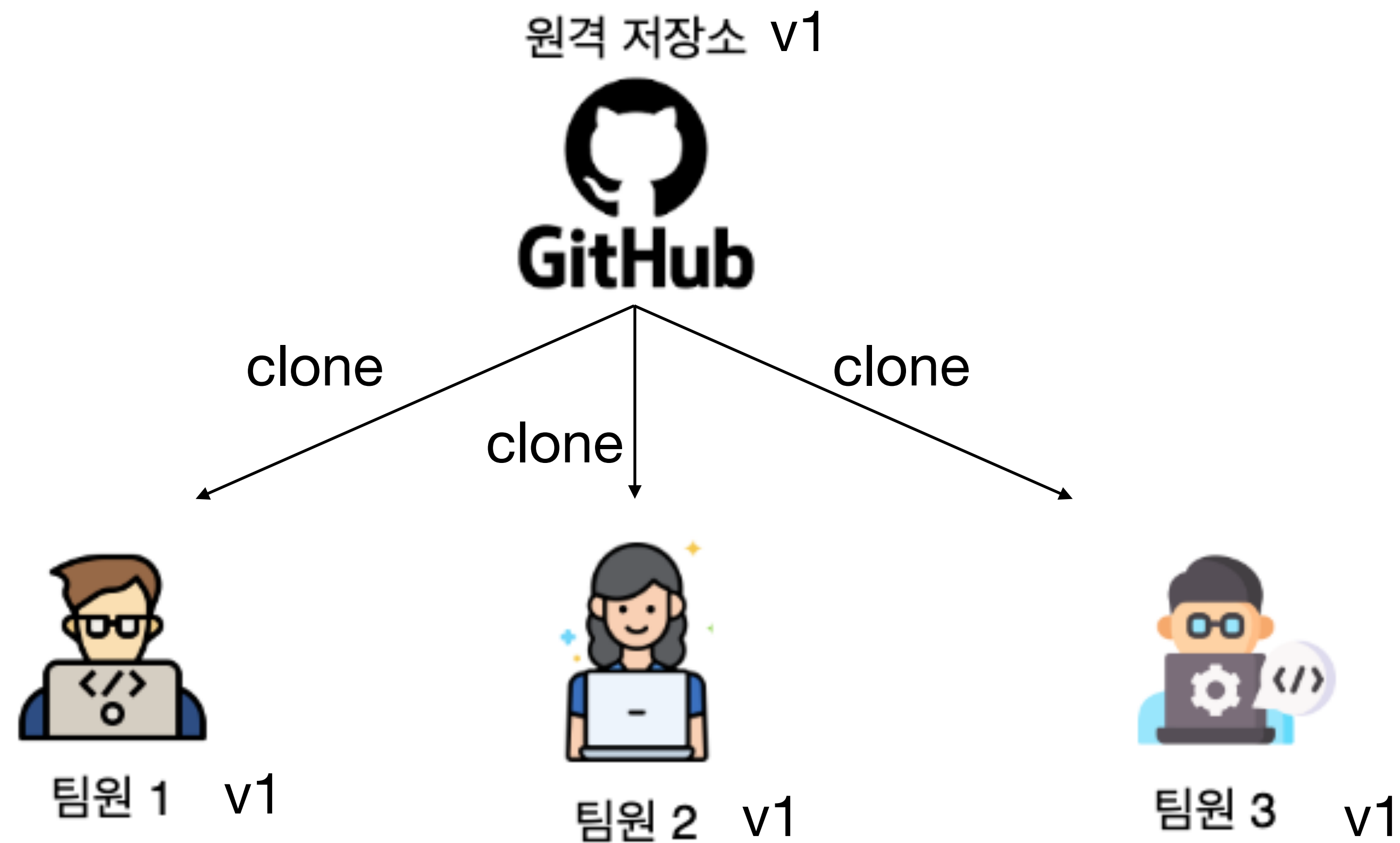
Initialize this repository with:

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

git commit, push, pull

Github 상에 초기 레포가 셋팅되었습니다. 이를 v1이라고 부르겠습니다.

이제 모든 팀원들이 각자의 환경에 github\_practice 레포를 clone 받아주세요



# git commit, push, pull

1번 팀원이 자신의 로컬 환경에서 Readme.md에 Hello world 라고 적어주고 커밋해보세요  
커밋이 하나 추가되었으니 이를 v2라고 부르겠습니다.

원격 저장소 v1



readme.md 수정  
git add readme.md  
git commit



팀원 1 v2



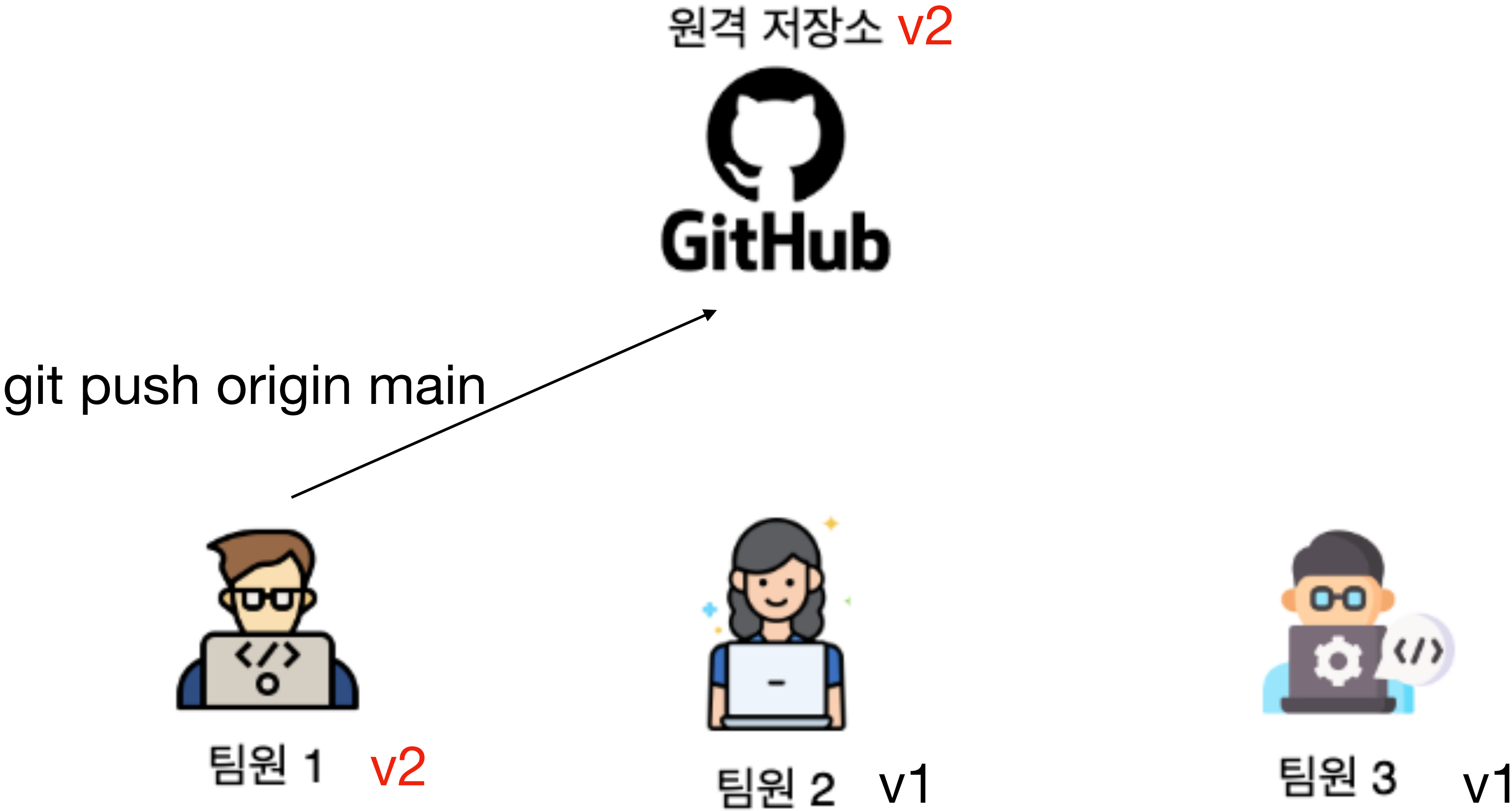
팀원 2 v1



팀원 3 v1

git commit, push, pull

1번 팀원이 커밋을 완료했다면, 이를 원격 저장소로 푸시해보세요



# git commit, push, pull

푸시가 정상적으로 완료되었다면 아래 처럼 hello world 메시지가 보일 것입니다.

main ▾


1 branch

0 tags

Go to file


Add file ▾

<> Code ▾



yeomko22 Modified readme.md

502d544 now 2 commits



README.md

Modified readme.md

now

README.md

github\_practice

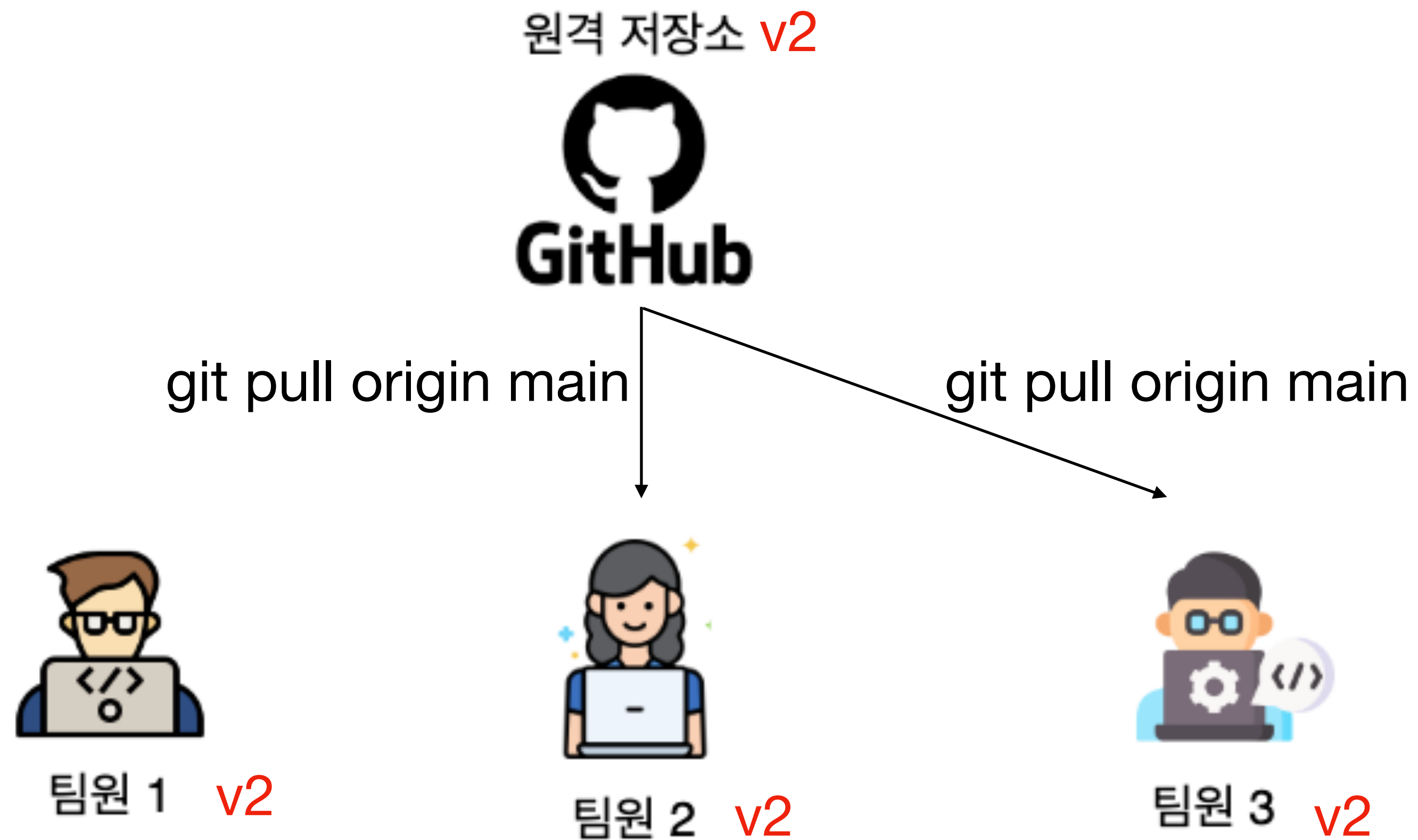
hello world



# git commit, push, pull

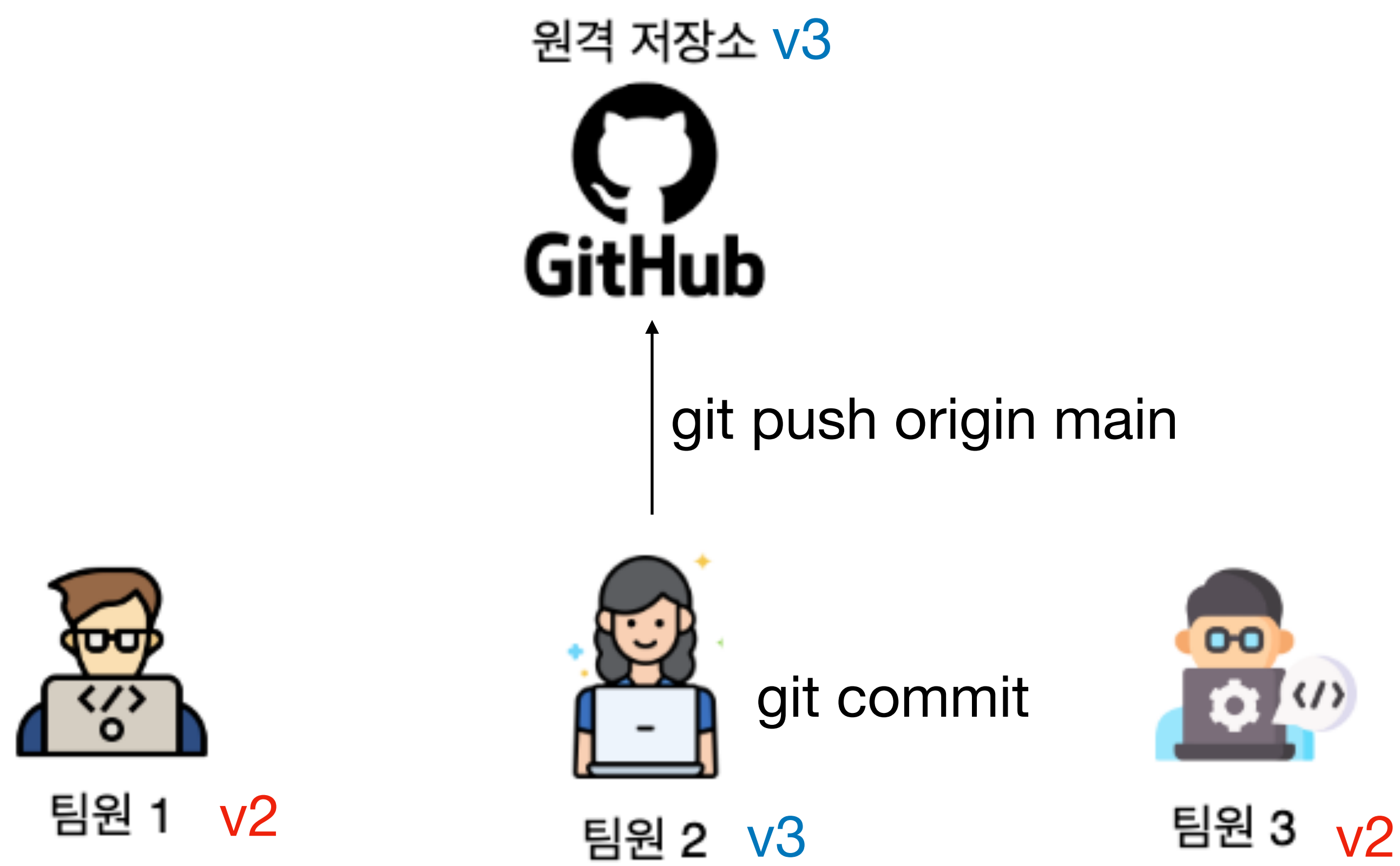
1번 팀원의 로컬 환경과 원격 레포는 v2이지만 팀원 2와 3의 로컬 환경은 여전히 v1입니다.

git pull origin main 명령어로 v2를 받아오세요



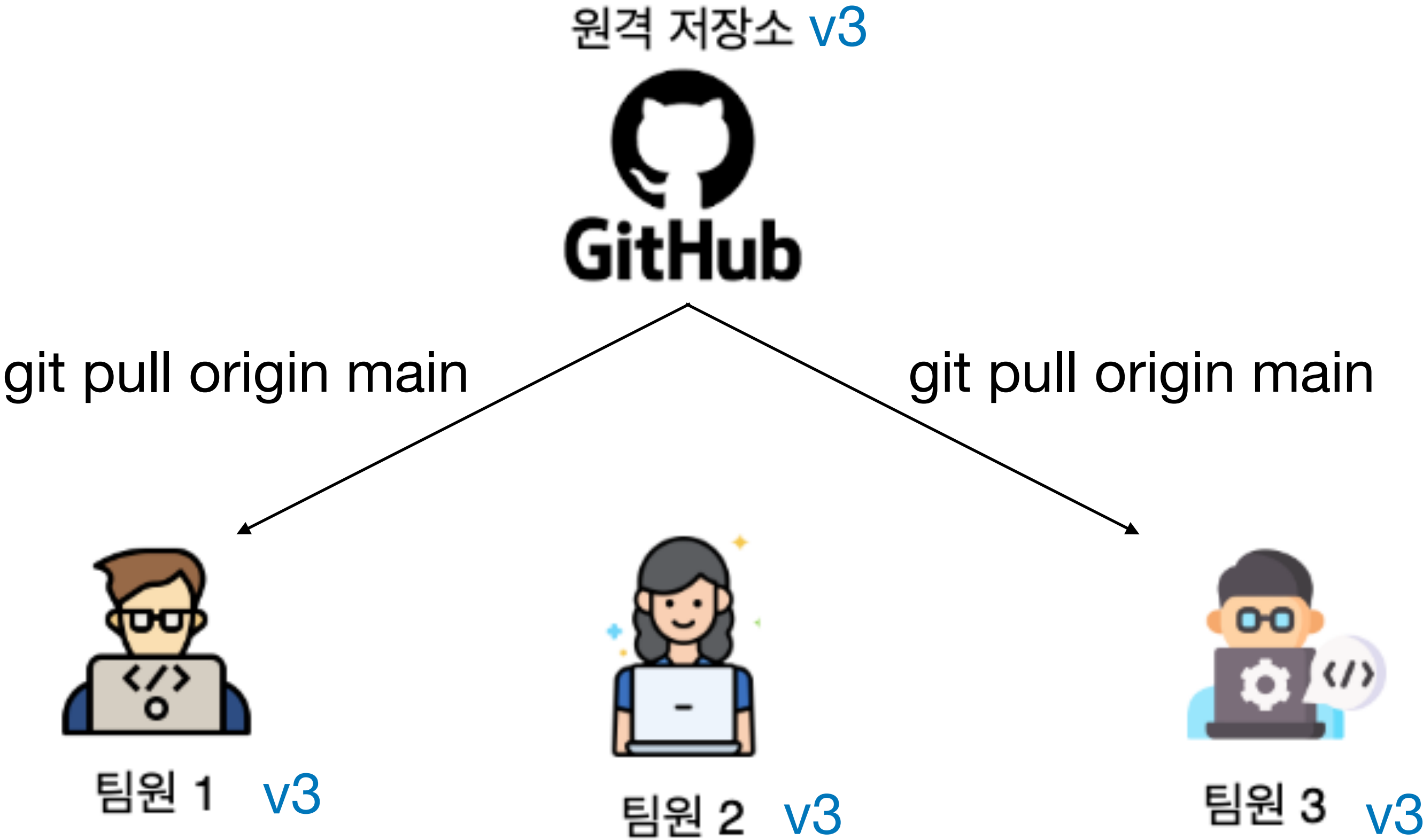
# git commit, push, pull

이번에는 2번 팀원이 hello world 문구를  
nice to meet you로 변경한 뒤에 커밋하고 푸시해보겠습니다.



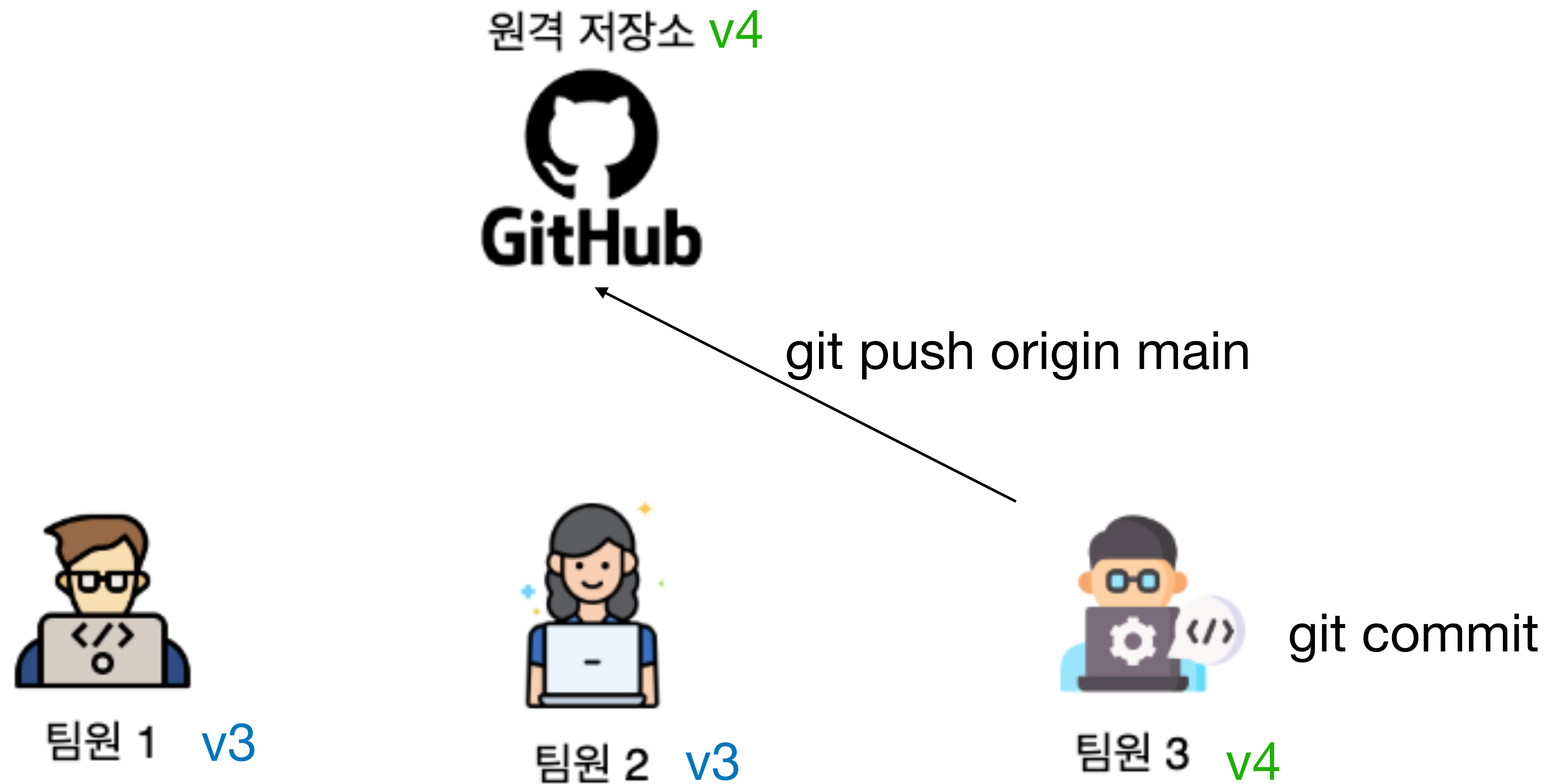
git commit, push, pull

마찬가지로 잘 푸쉬되었는지 확인해보고, 나머지 두 팀원들은 pull 받아보세요



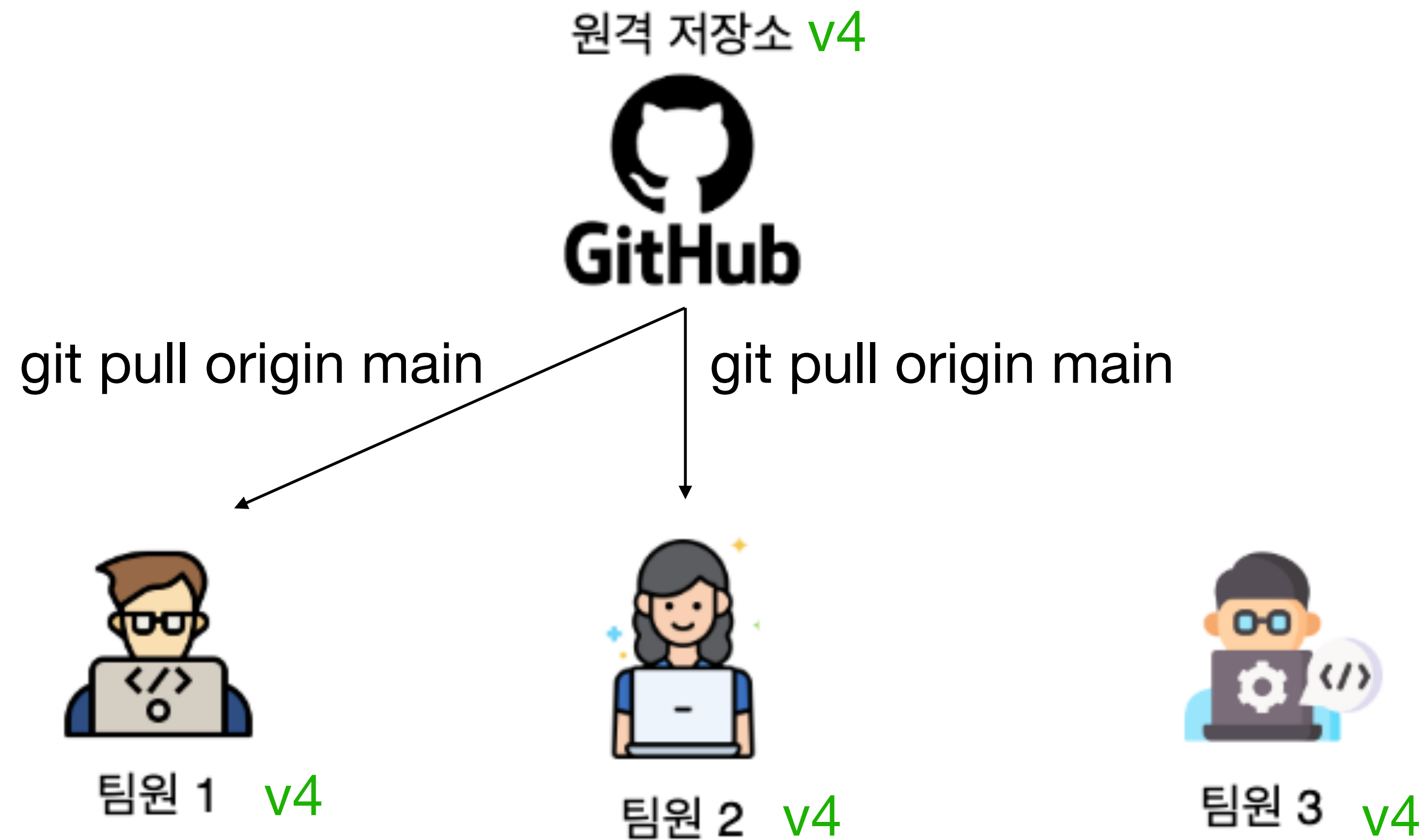
# git commit, push, pull

마지막으로 3번 팀원은 readme.md에 nice to meet you 아래에 I'm happy를 추가한 뒤,  
커밋하고 푸시를 해보세요



git commit, push, pull

마지막으로 3번 팀원은 readme.md에 nice to meet you 아래에  
I'm happy를 추가한 뒤, 커밋하고 푸시를 해보세요



# branch, pull request

## branch

- 여러 사람이 동시에 작업할 때, 서로의 작업에 영향을 주지 않기 위해서 분리하는 버전

## git switch

- 로컬 환경에서 브랜치를 전환할 때 사용하는 명령어
- `git switch -c “브랜치명”`을 사용하면 내 로컬 환경에 새 브랜치를 만들 수 있음

## pull request

- 특정 브랜치에 업로드 된 작업 내역을 다른 브랜치에 합치고 싶을 때 사용하는 기능
- 주로 각자 나눠서 작성한 내용을 main 브랜치에 합칠 때 사용

## code review

- 다른 사람의 작업 내역을 보고, 이 코드는 이렇게 수정하면 좋겠다고 커멘트를 남기는 과정

# branch, pull request

모든 팀원이 자신의 로컬 환경에서 `git switch -c` “자기가 쓸 브랜치 명”으로 브랜치 생성

원격 저장소



팀원 1

`git switch -c “p1”`



팀원 2

`git switch -c “p2”`



팀원 3

`git switch -c “p3”`

# branch, pull request

모든 팀원이 자신의 로컬 환경에서 `git switch -c` “자기가 쓸 브랜치 명”으로 브랜치 생성



팀원 1

`git switch -c "p1"`



팀원 2

`git switch -c "p2"`



팀원 3

`git switch -c "p3"`



# branch, pull request

git status를 입력했을 때, 자신이 새로 생성한 브랜치가 나오면 성공

```
> git switch -c "p1"  
새로 만든 'p1' 브랜치로 전환합니다  
> git status  
현재 브랜치 p1  
커밋할 사항 없음, 작업 폴더 깨끗함
```

# branch, pull request

1번 팀원이 자기 브랜치에서 간단한 streamlit 앱 띄우는 파일 작성해서 커밋한 뒤에 푸쉬

```
import streamlit as st

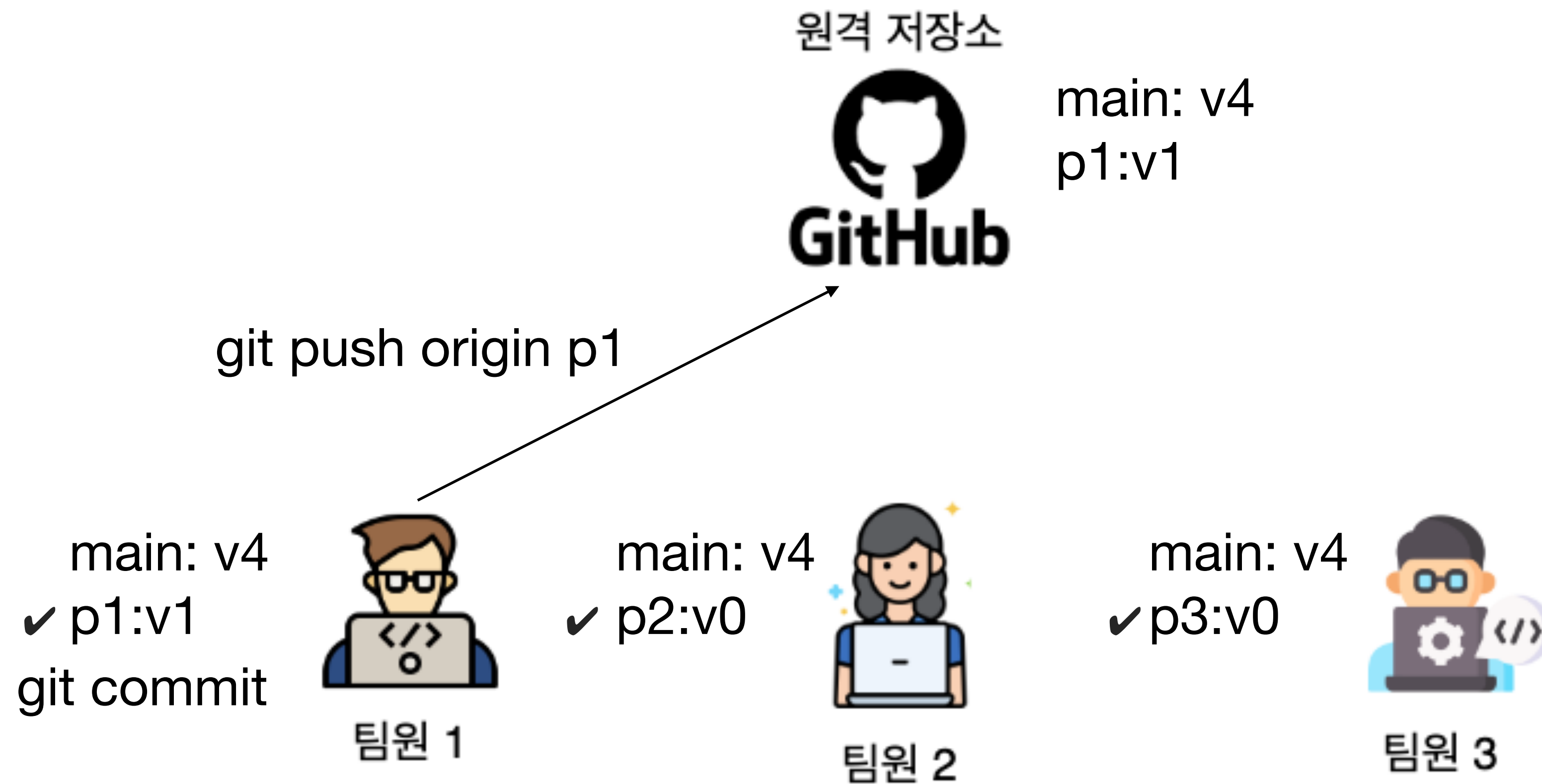
st.title("hello world")
st.subheader("nice to meet you")
st.text("practice text")
```

home.py

# branch, pull request

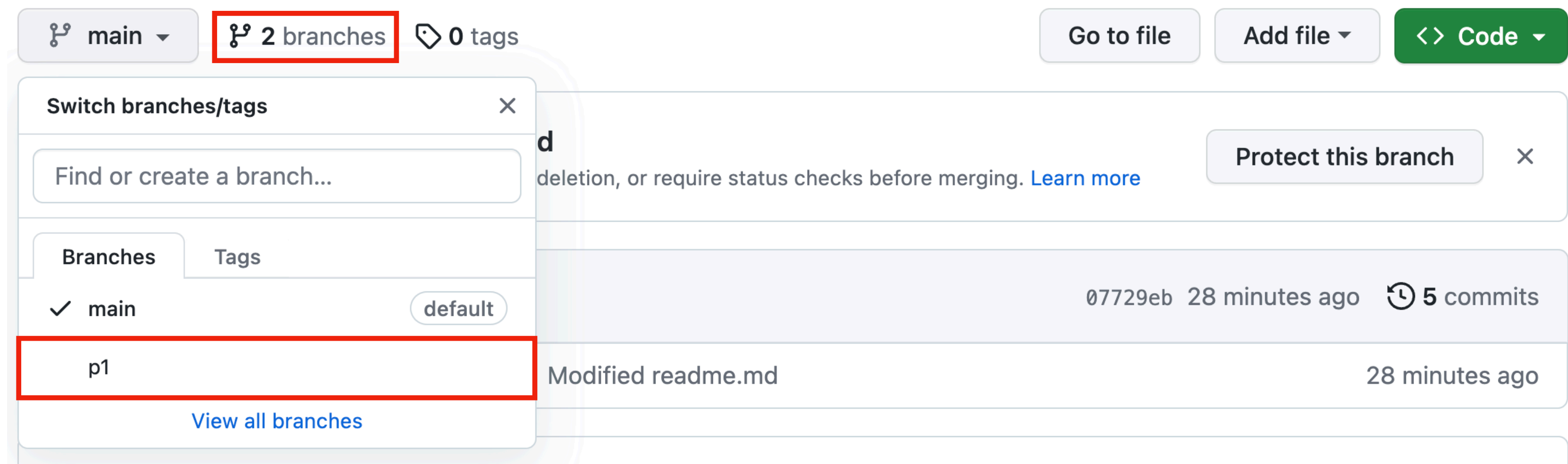
1번 팀원이 자기 브랜치에서 간단한 streamlit 앱 띄우는 파일 작성해서 커밋한 뒤에 푸쉬

푸쉬 할 때는 자기가 만든 브랜치 명을 사용할 것




# branch, pull request


원격 저장소에 p1 브랜치가 생성되었다면 성공




# branch, pull request

원격 저장소에서 p1 브랜치를 선택하면 home.py를 볼 수 있음


 p1 ▾

 2 branches


 0 tags


Go to file


Add file ▾



 Code ▾

This branch is [1 commit ahead](#) of main.

 [Contribute](#) ▾

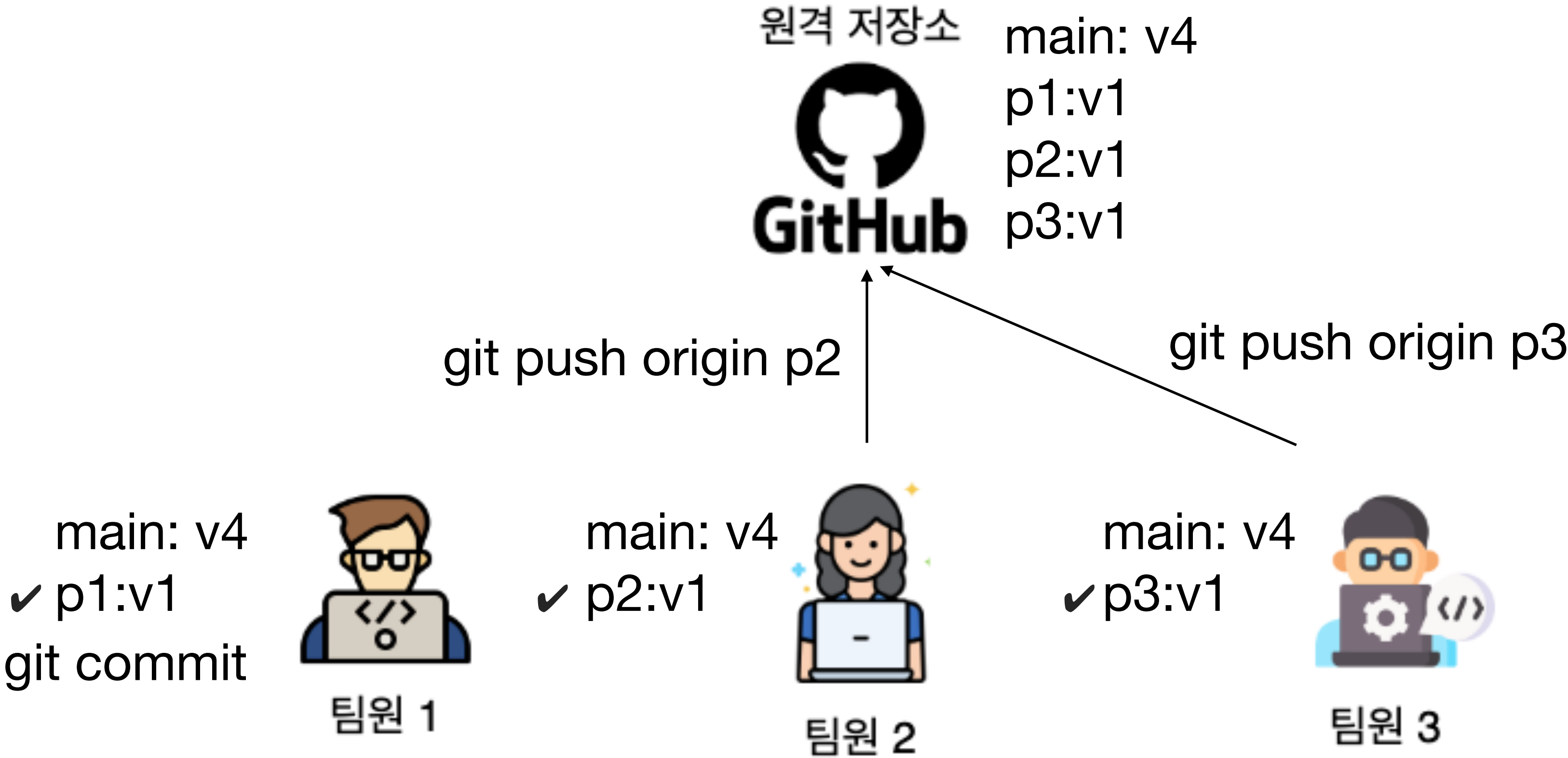
 **yeomko22** Add home.py

c7dca58 2 minutes ago  6 commits

 README.md	Modified readme.md	30 minutes ago
 <b>home.py</b>	Add home.py	2 minutes ago

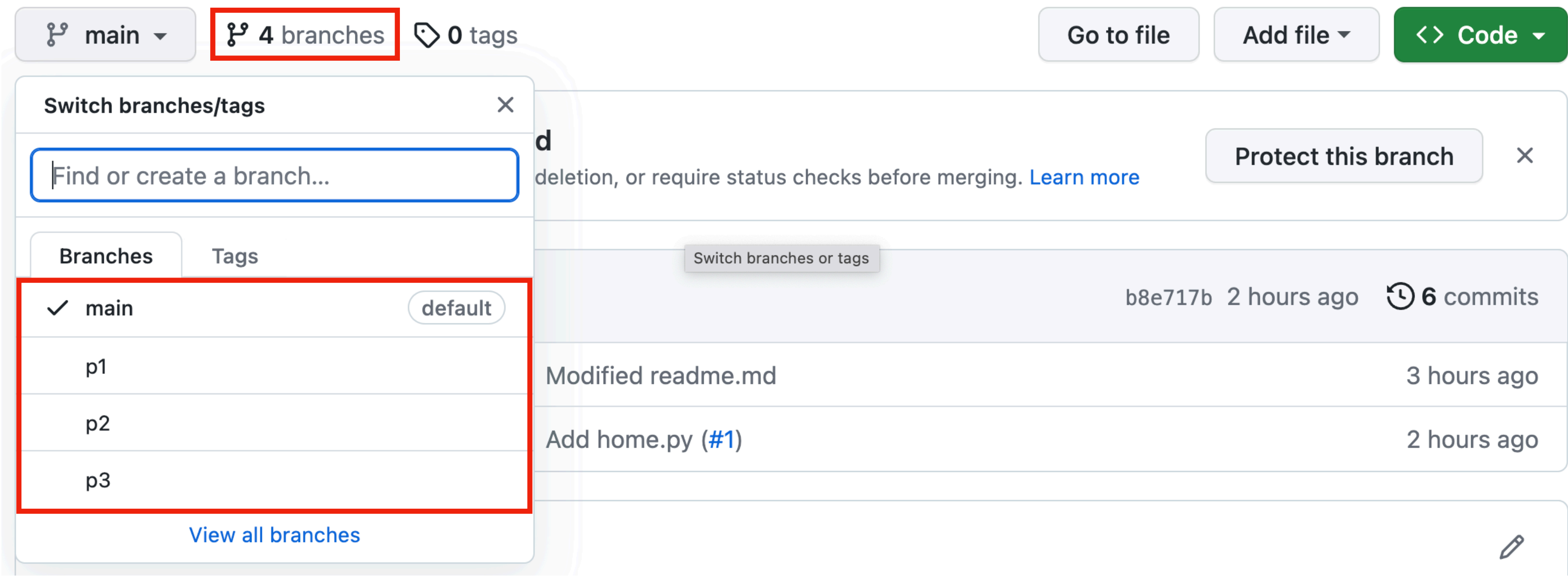
# branch, pull request

2번과 3번 팀원은 각각 pages 폴더 아래에 p2.py와 p3.py를 만들어서  
간단하게 텍스트를 보여주는 페이지를 만들고, 커밋한 다음 푸쉬할 것




# branch, pull request

원격 레포에서 총 4개의 브랜치를 확인할 수 있으면 성공



# branch, pull request

p1 브랜치에 작업한 내역을 main 브랜치에 합치고 싶다면 pull request를 생성

 p1 had recent pushes 7 minutes ago

Compare & pull request


Filters ▾

is:pr is:open

Labels 9

Milestones 0

New pull request





main 브랜치에 합치도록 설정해주고 변경 사항을 간단히 적어줌

코드 리뷰를 요청하고 싶다면 reviewer에 팀원들 지정 (급하다면 리뷰 생략 가능)

base: main compare: p1 ✓ **Able to merge.** These branches can be automatically merged.

Add home.py

Write Preview H B I

앱의 기본 화면인 home.py를 추가했습니다.

**Reviewers**   
No reviews

**Assignees**   
No one—assign yourself

**Labels**   
None yet

리뷰를 요청받은 팀원은 file changed 탭으로 이동해서 코드 밑에 커멘트를 남길 수 있음

2번 3번 팀원들이 커멘트를 남겨줌

Conversation 0

Commits 1

Checks 0

Files changed 1

+6 -0

Changes from all commitsFile filterConversationsJump to0 / 1 files viewedReview changes

6 home.py

View fullViewed

@@ -0,0 +1,6 @@

1 + import streamlit as st

2 +

3 + st.title("hello world")

커멘트 남기고 싶은 라인 옆 + 버튼을 누름

WritePreview

🔗 H B I ≡ <> 🔗 ≡ ≡ ≡ @ ↗ ↶

앱 제목을 변경하면 좋겠습니다.


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

CancelAdd single commentStart a review

# branch, pull request

코드 리뷰를 받은 뒤에 변경 요청 사항을 반영하여 1번 팀원의 로컬 환경에서 다시 커밋

원격 저장소



main: v4  
p1:v1  
p2:v1  
p3:v1

main: v4  
✓ p1:v2  
git commit



팀원 1

main: v4  
✓ p2:v1



팀원 2

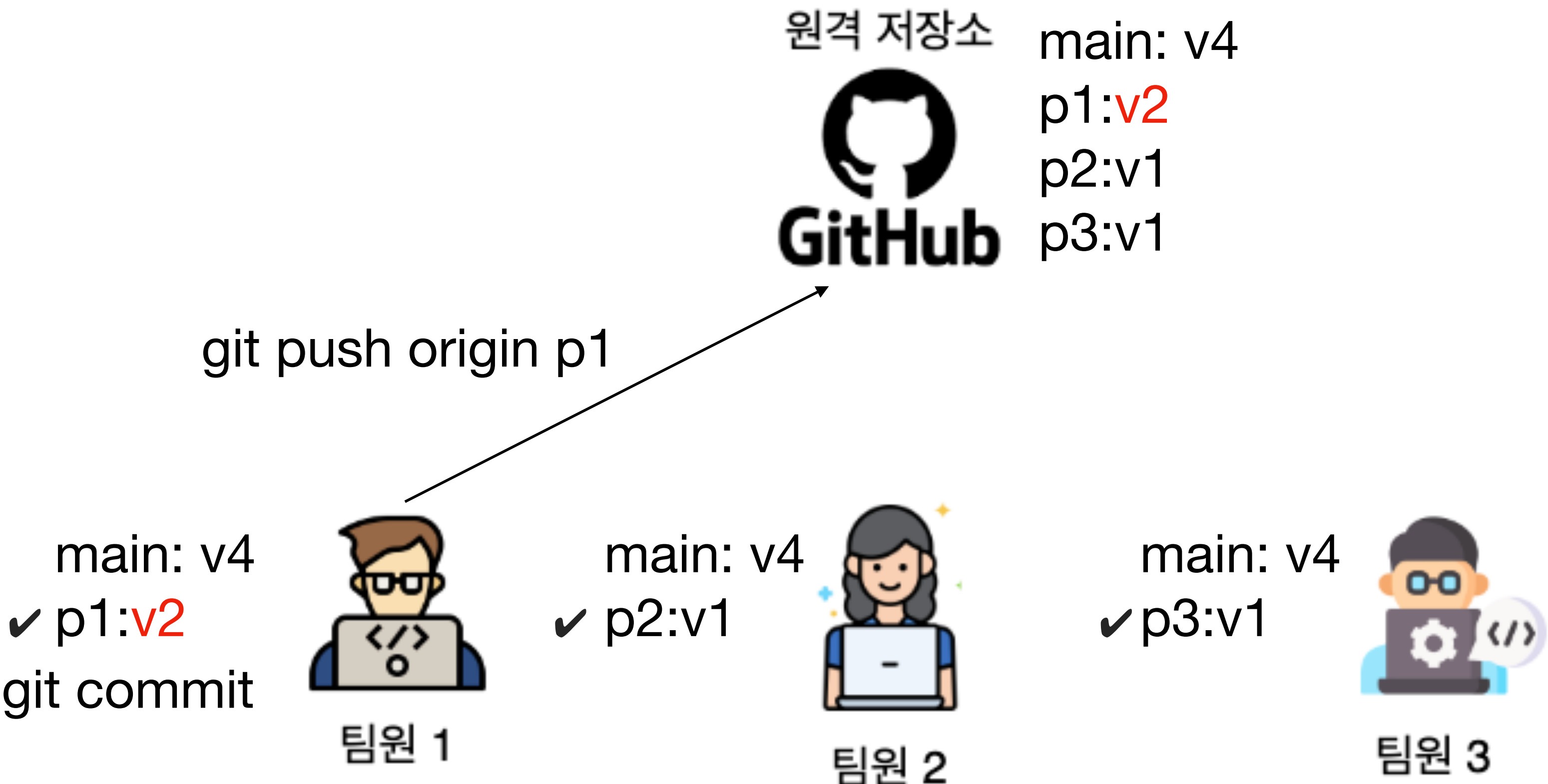
main: v4  
✓ p3:v1



팀원 3

# branch, pull request

코드 리뷰를 받은 뒤에 변경 요청 사항을 반영하여 1번 팀원의 로컬 환경에서 다시 커밋



# git commit, push, pull

코드에 문제가 없다면 2번 3번 팀원이 Approve를 눌러줌

Conversation 0Commits 1Checks 0Files changed 1+6-0

Changes from all commitsFile filterConversationsJump to0 / 1 files viewedFinish your review 1

6 home.py

@@ -0,0 +1,6 @@

1 + import streamlit as s

2 +

3 + st.title("hello world

4 + st.subheader("nice to

5 + st.text("practice tex

6 +

yeomko22 Pending

앱 제목을 변경하면 좋겠습니다.

Reply...

Finish your review

WritePreview

H B I ≡ <> ↻ ⋮ ⋮ ⋮ @ ↗ ↶

Leave a comment

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

☒ Comment

Submit general feedback without explicit approval.

☐ Approve

Submit feedback and approve merging these changes.

☐ Request changes

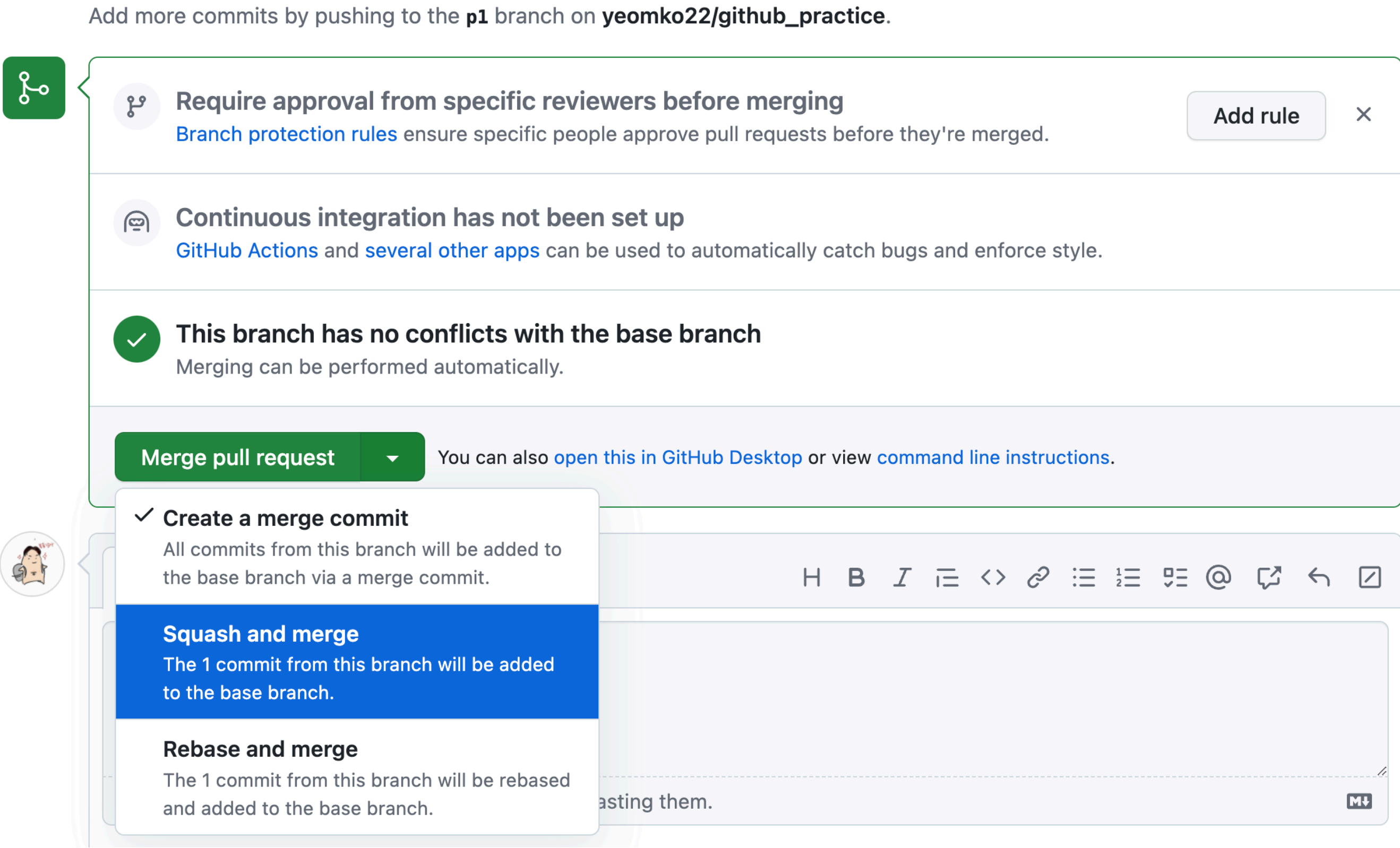
Submit feedback that must be addressed before merging.

Submit review 1 pending comment

# git commit, push, pull

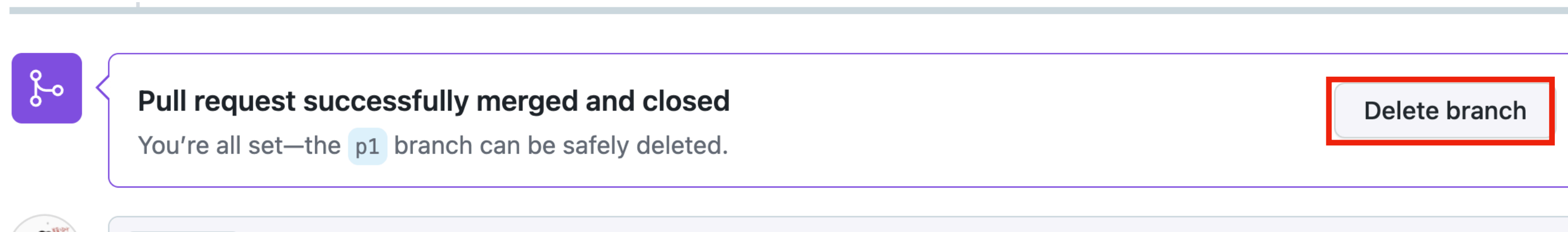
팀원들로부터 Approve를 받았다면 이제 main 브랜치에 p1 브랜치를 합칠 수 있음

이 때, squash merge를 선택하여 합쳐줌



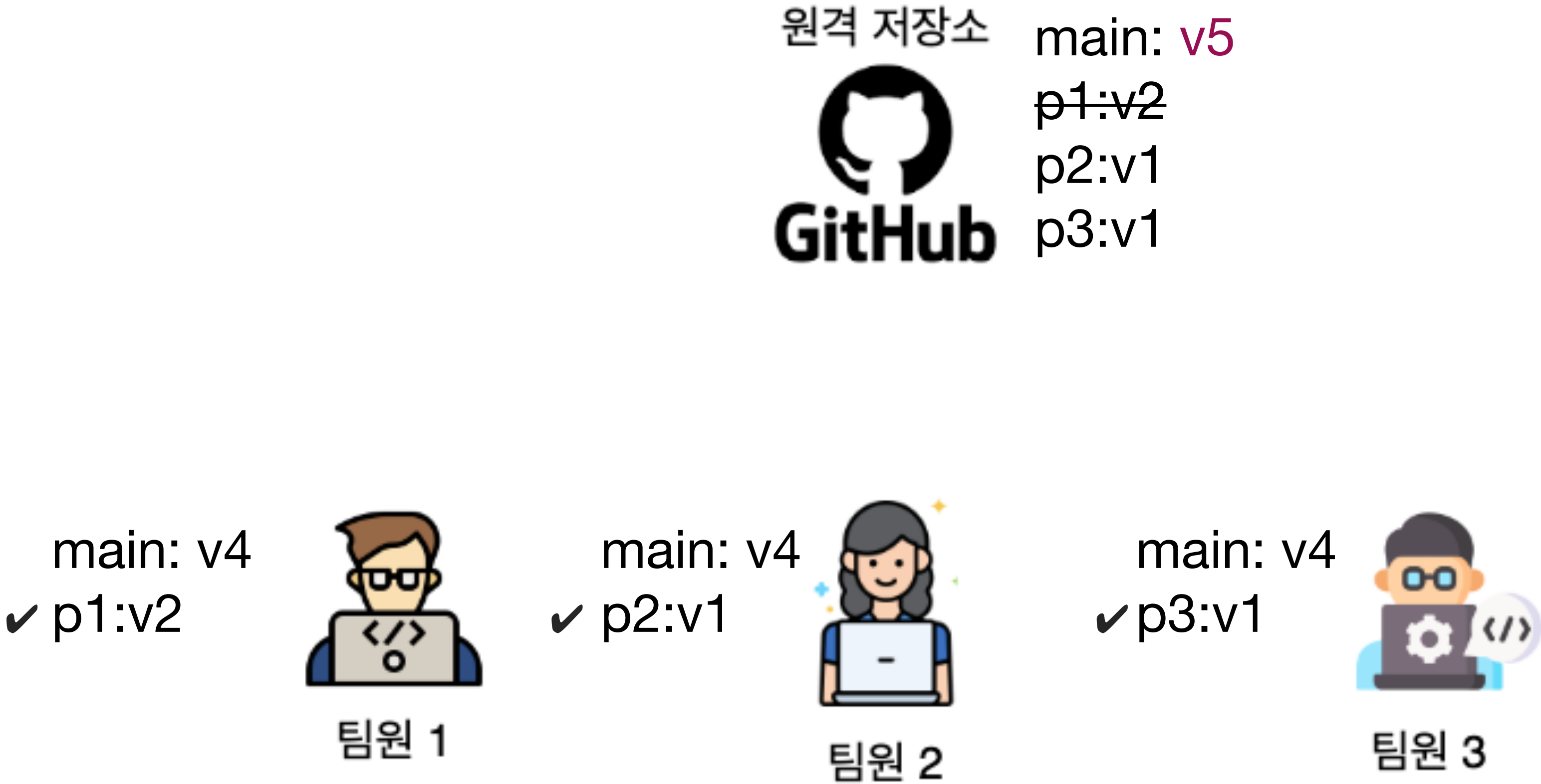
# git commit, push, pull

그러면 p1 브랜치에서 작업한 home.py가 main 브랜치에 합쳐짐  
merge된 브랜치는 원격 레포에서 지워주는게 좋은 습관



git commit, push, pull

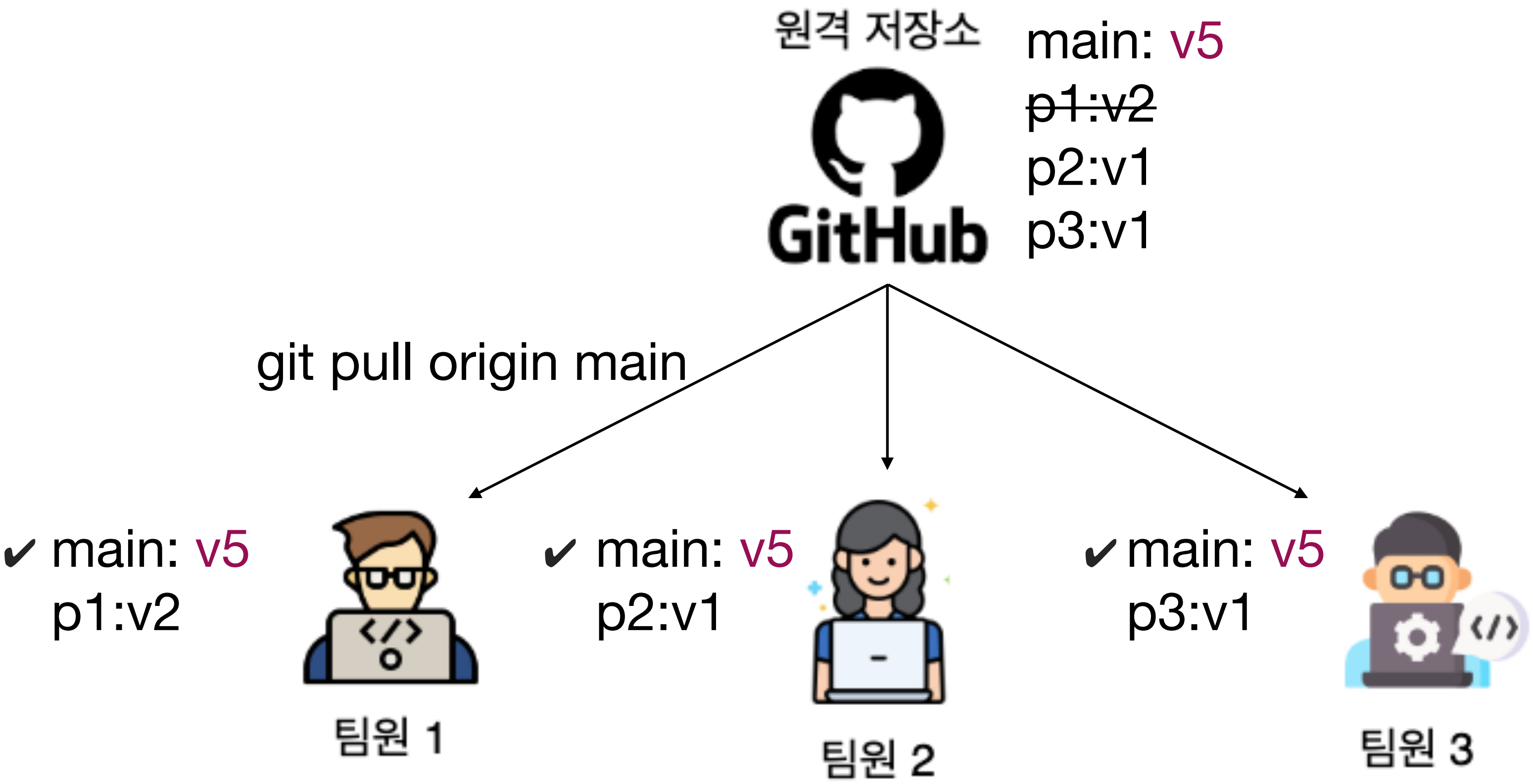
원격 저장소에 p1 브랜치를 main 브랜치에 merge하면 main 브랜치의 버전이 하나 올라감





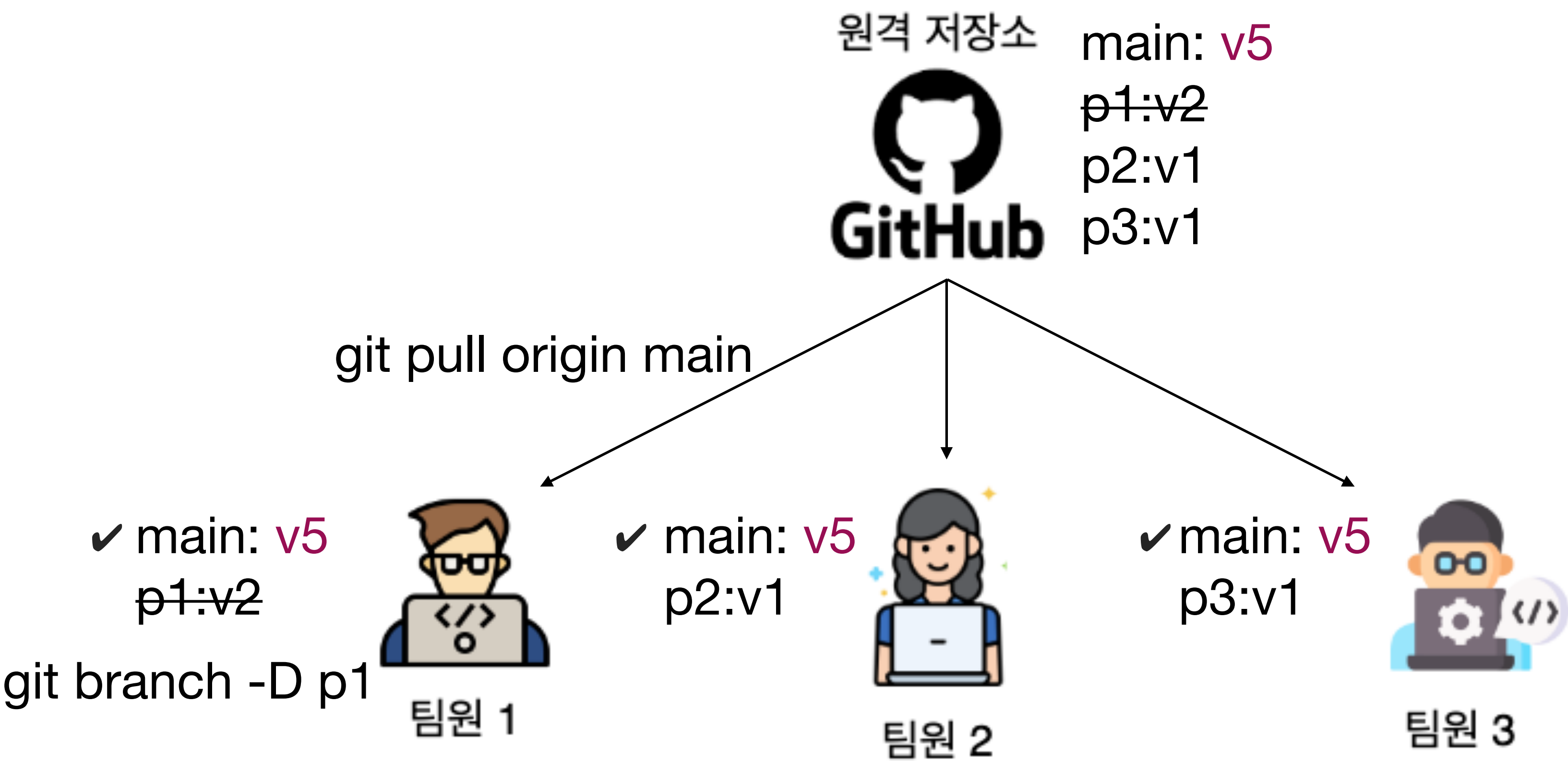
# git commit, push, pull

1, 2, 3번 팀원 모두 git switch main 명령어를 이용해서 main 브랜치로 돌아온 뒤,  
git pull origin main 명령어를 이용해서 main 브랜치의 버전을 당겨옵니다.



# git commit, push, pull

1번 팀원은 git branch -D 명령어를 이용하여 자신의 로컬환경에서도 p1 브랜치를 지워줍니다.  
추가적으로 작업을 하고 싶다면, 새롭게 당겨온 main에서 브랜치를 새로 따서 작업을 하면 됩니다.



# git rebase

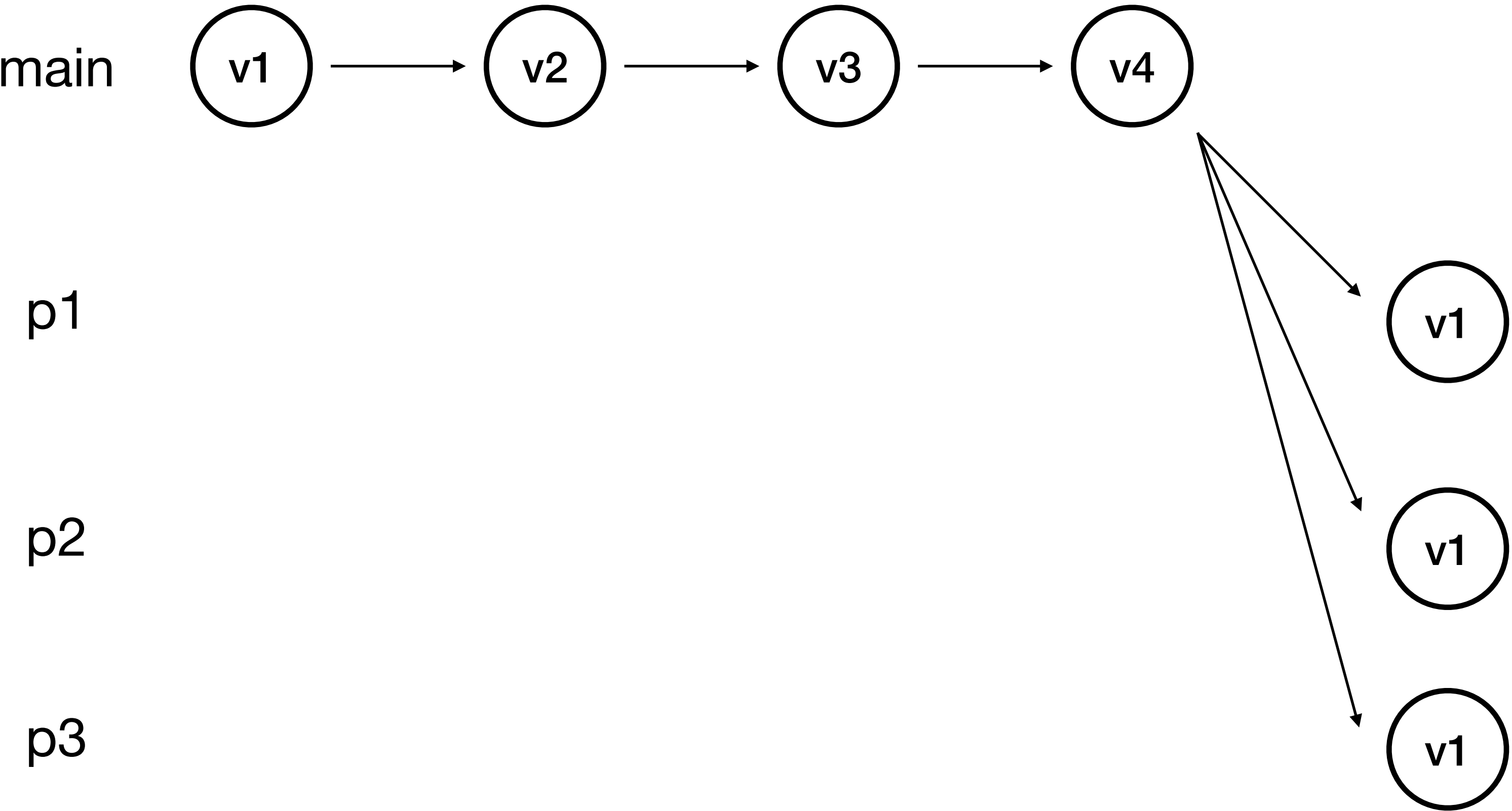
git rebase

: 원래 브랜치를 만들었던 브랜치에 새로운 커밋이 추가된 경우,

이를 현재 브랜치에 추가해주는 명령어

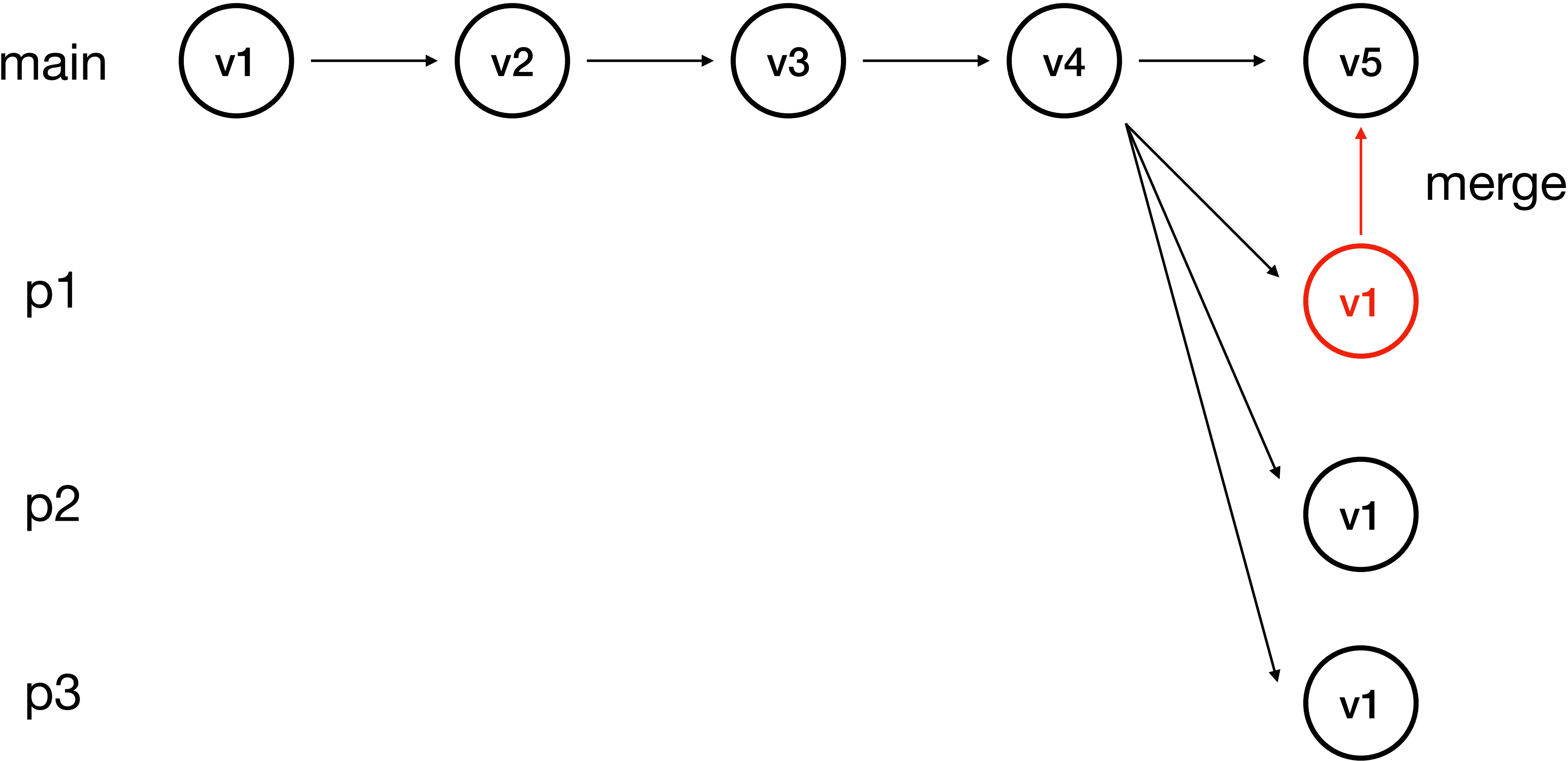
# git rebase

기존에 main 브랜치의 v4에서 p1, p2, p3 세 브랜치를 생성하여 각자 작업을 진행



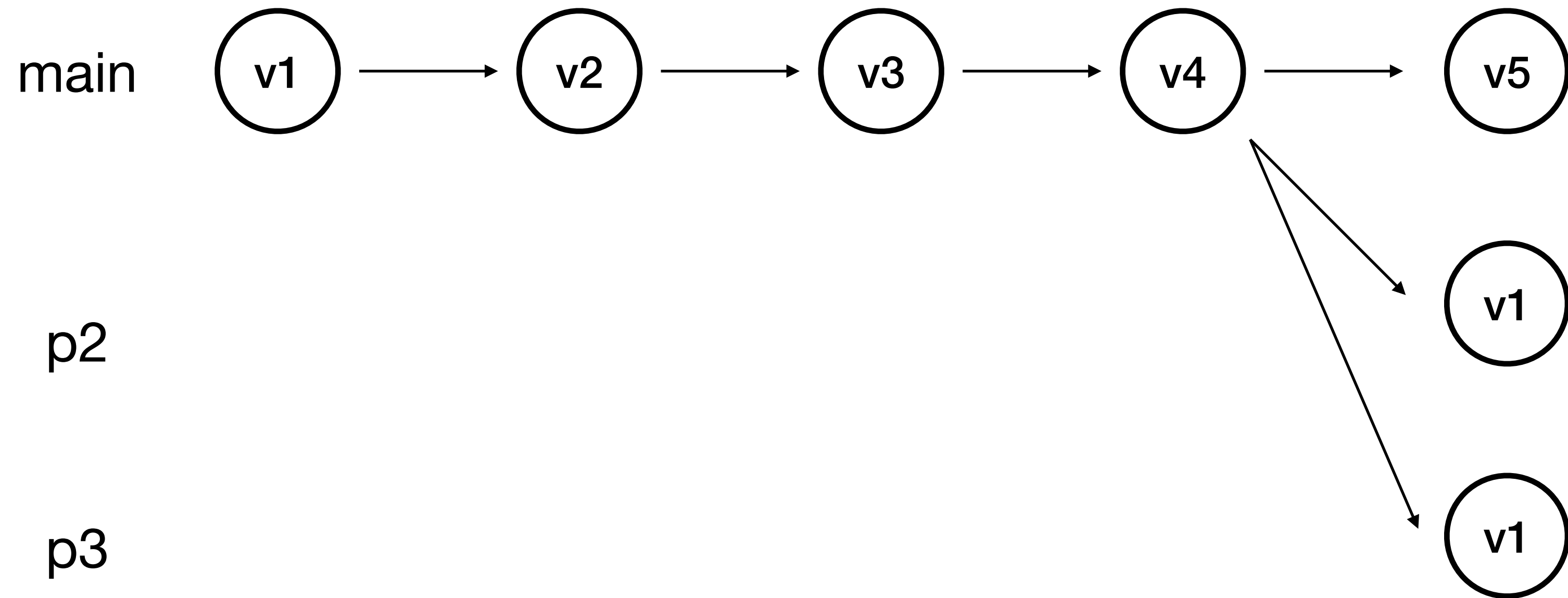
# git rebase

pull request와 merge를 통해서 p1 브랜치 작업물이 main에 반영됨



# git rebase

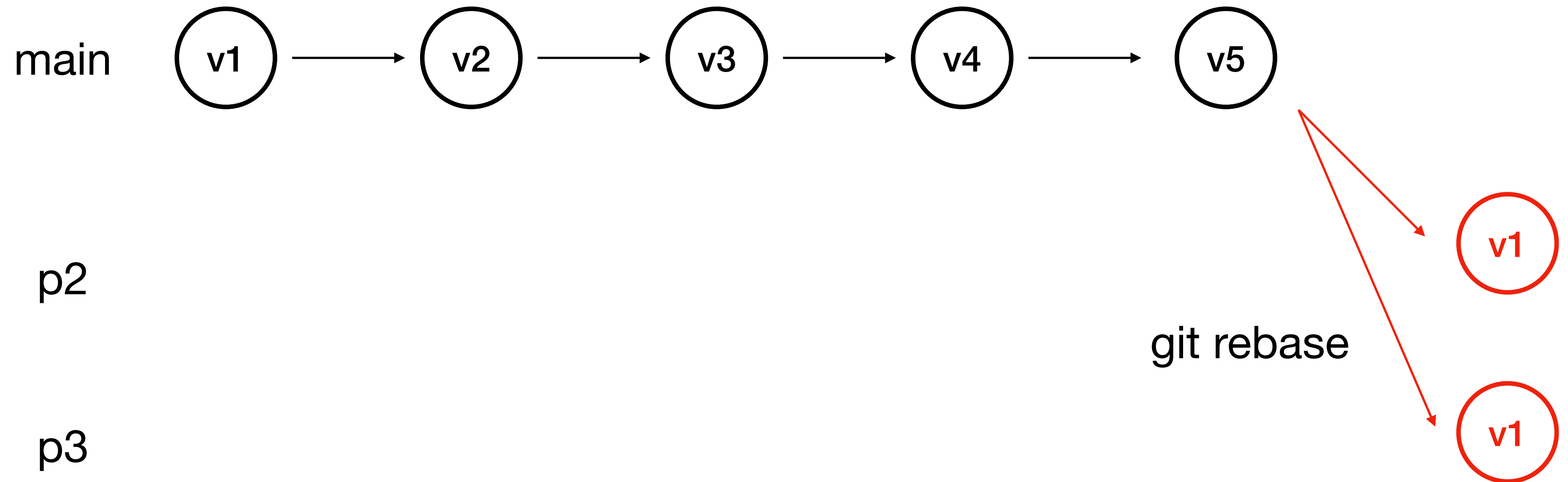
그런데 여전히 p2와 p3는 main 브랜치의 v4를 기준으로 하고 있어서  
v5에 새로 추가된 코드가 반영되지 못한 상태입니다.



# git rebase

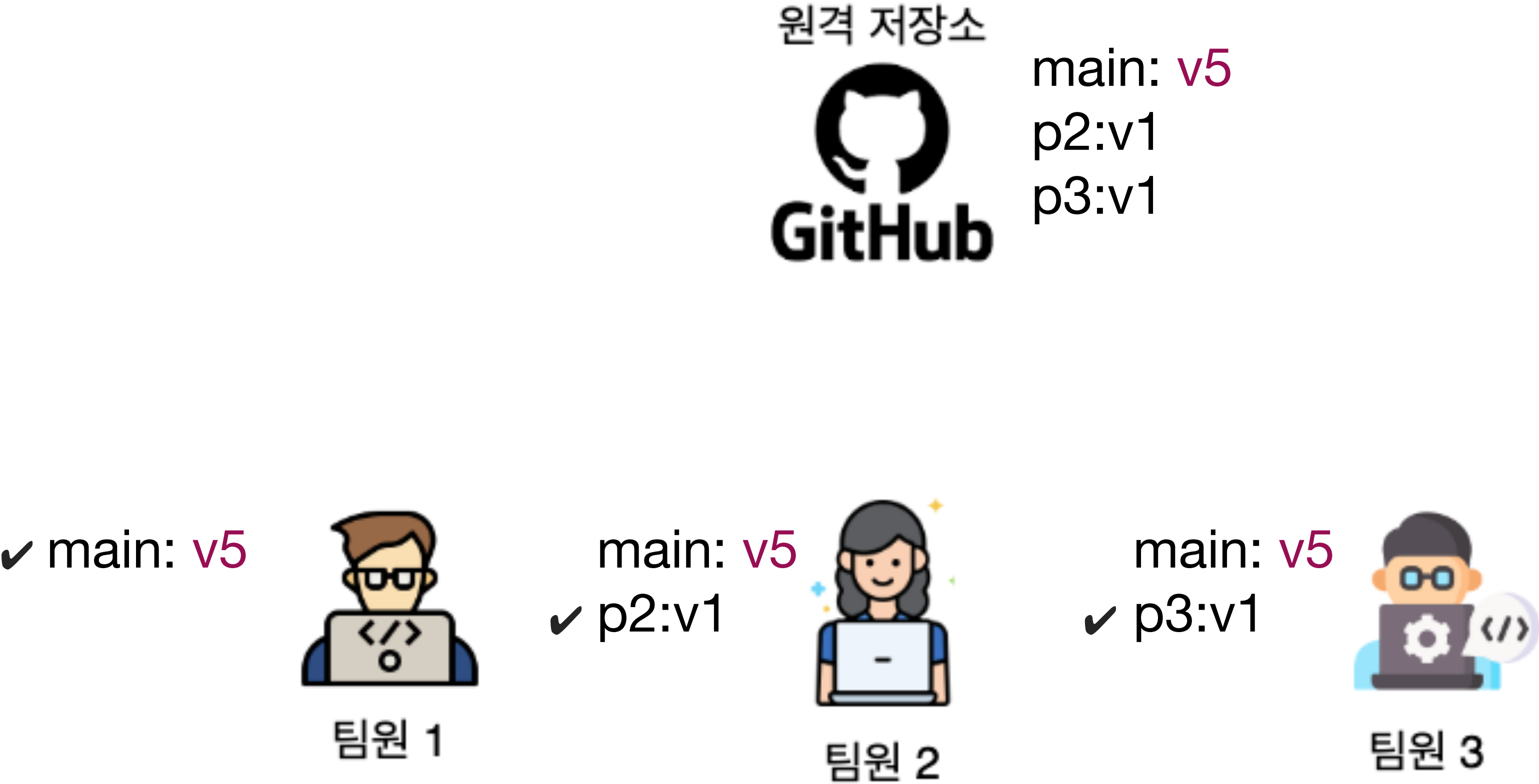
이를 git rebase 명령어를 이용하여 마치 v5를 기준으로 p2와 p3 브랜치를 만든 것 처럼

새로 추가된 커밋을 반영해주겠습니다.



git commit, push, pull

2번과 3번 팀원은 git switch 명령어를 이용하여 자신이 만든 브랜치로 이동합니다.





git commit, push, pull

그 다음 git rebase 명령어를 사용하여 p2, p3 브랜치가  
각각 main 브랜치의 v5 커밋을 반영하도록 합니다.

원격 저장소



main: v5  
p2:v1  
p3:v1

✓ main: v5



팀원 1

main: v5  
✓ p2:v1

git rebase main



팀원 2

main: v5  
✓ p3:v1

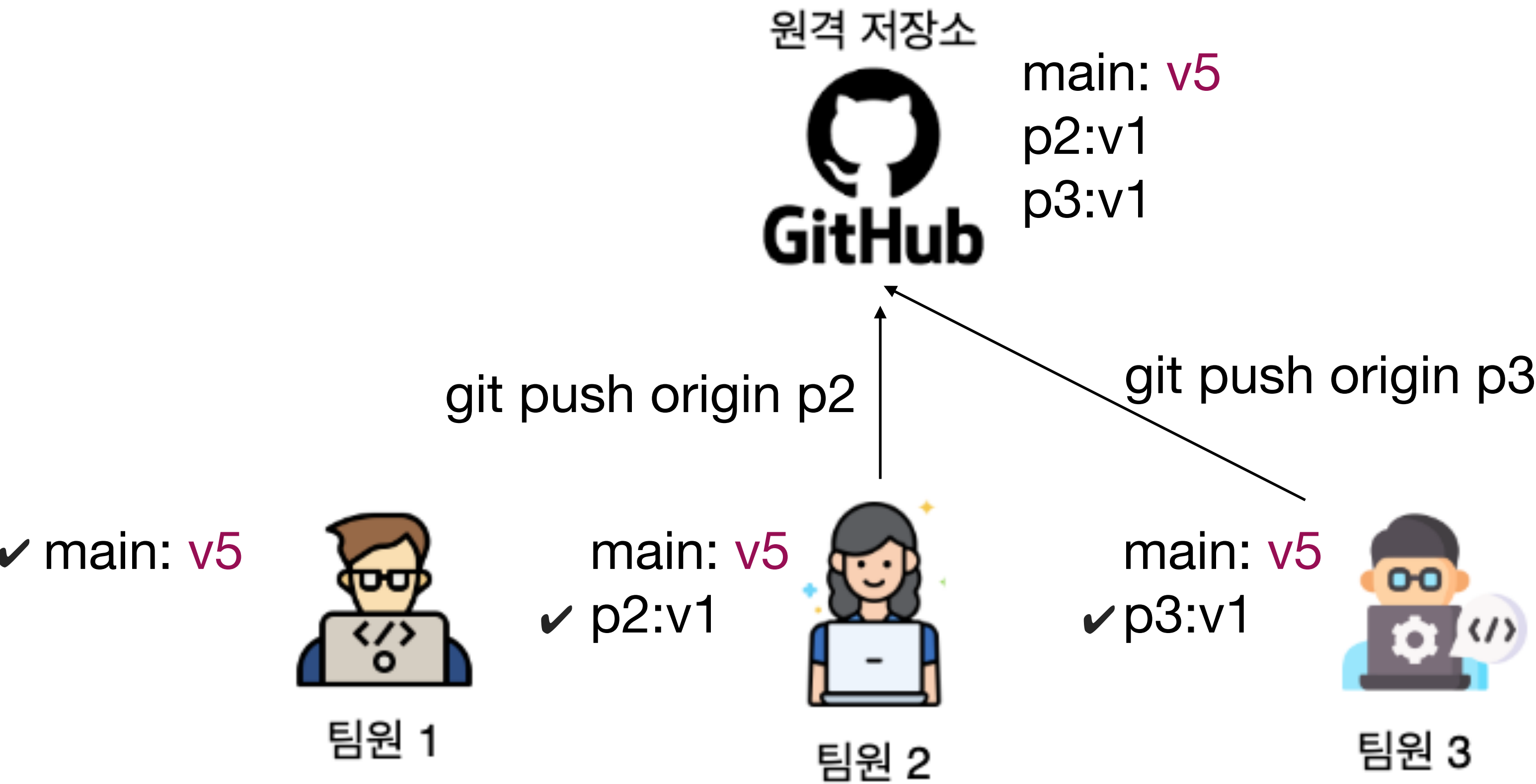
git rebase main



팀원 3

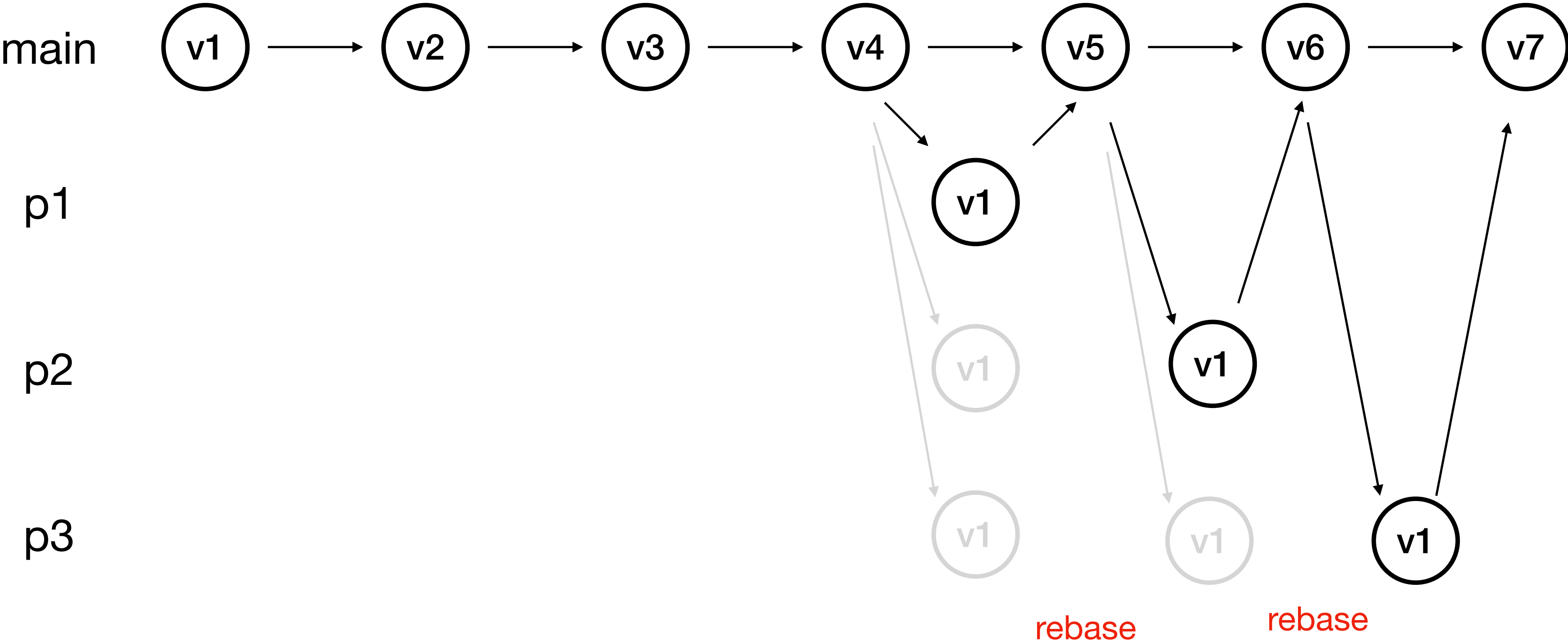
# git commit, push, pull

그 다음에 다시 git push 명령어를 이용해서 변경된 이력을 원격 저장소에 푸시해주고  
pull request로 p2와 p3도 각각 main에 merge 해줍니다.



# git commit, push, pull

최종적으로 main 브랜치의 커밋 이력은 이렇게 만들어집니다.



# git을 이용한 협업 프로세스

github으로 협업을 할 때는 다음과 같은 요령으로 진행하시면 됩니다.

1. main 브랜치에서 각자 작업할 브랜치 꺾어서 작업
2. 작업이 완료되면 커밋한 뒤 푸쉬
3. 내 작업물을 main에 합치고 싶다면 pull request 생성
4. 코드 리뷰한 뒤, main에 merge
5. main 브랜치에 커밋이 추가되면 이를 pull 받은 뒤, 내 브랜치 rebase
6. 내 작업물이 merge 되었다면 최신 main 브랜치에서 새로 브랜치를 꺾은 뒤 작업