

# #1. git으로 협업하기

- ##1. add, commit, push
- ##2. 원격 저장소와의 상호작용
- ##3. 분기 나누기 (branch)
- ##4. pull request (협업) 실습

Git?

# 버전 관리 시스템

# 버전 관리 시스템

되돌리고

백업하고

협업하기 위한 도구

의미로 곱씹어보는

add, commit, push

버전이 만들어지는 두 개의 단계

## 버전이 되기까지 거쳐가는 세 개의 공간

Working directory (작업공간)

Staging Area

Repository

버전이 만들어지는 두 개의 단계

## 버전이 되기까지 거쳐가는 세 개의 공간

Working directory (작업공간)

- 내가 코드작업을 하는 공간
- 파일들이 생성/수정/삭제되는 공간
- 즉, 변경사항이 생기는 공간



버전이 만들어지는 두 개의 단계

## 버전이 되기까지 거쳐가는 세 개의 공간

Working directory (작업공간)



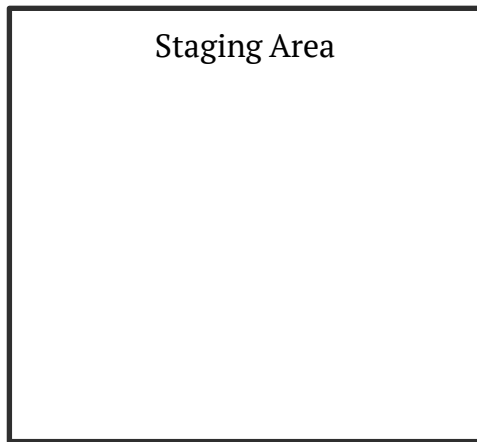
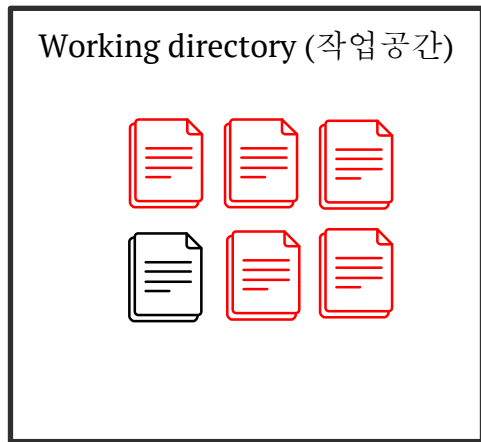
: 변경된 파일

Q. Working Directory의 모든 변경사항들을  
버전으로 만들어야 할까?

A. 변경사항들 중 다음 버전이 될 파일들을  
선별해서 선별된 파일들을 버전으로 만들자!

버전이 만들어지는 두 개의 단계

## 버전이 되기까지 거쳐가는 세 개의 공간



- 버전이 될 후보들이 올라오는 공간
- Working directory에서 선별

: 변경된 파일

버전이 만들어지는 두 개의 단계

버전이 되기까지 거쳐가는 세 개의 공간

내 컴퓨터 속  
저장소!

Working directory (작업공간)



Staging Area



Repository

Version 4.0

Version 3.0

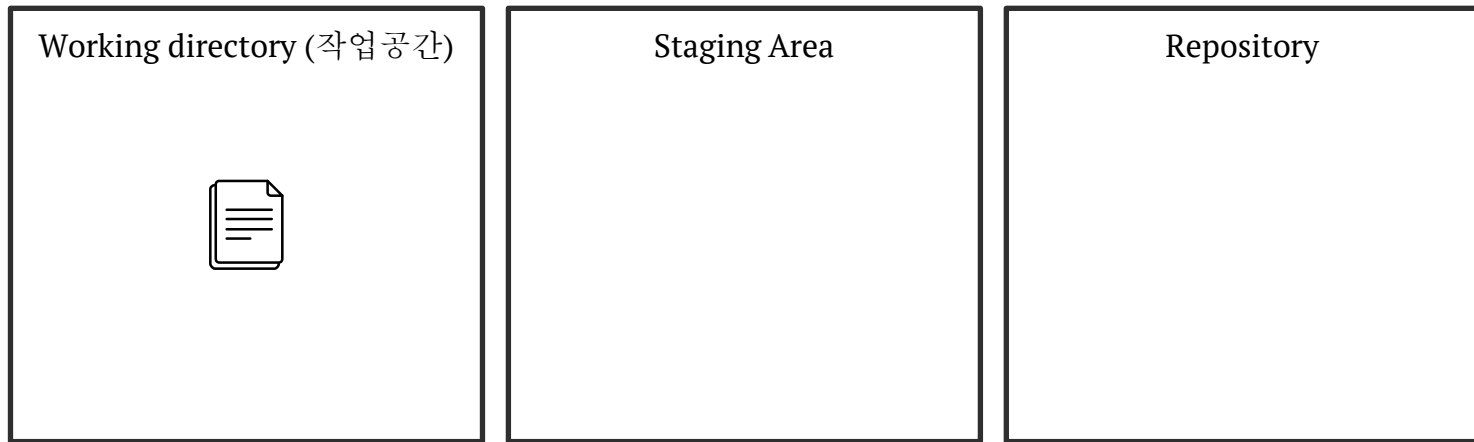
Version 2.0

Version 1.0

: 변경된 파일

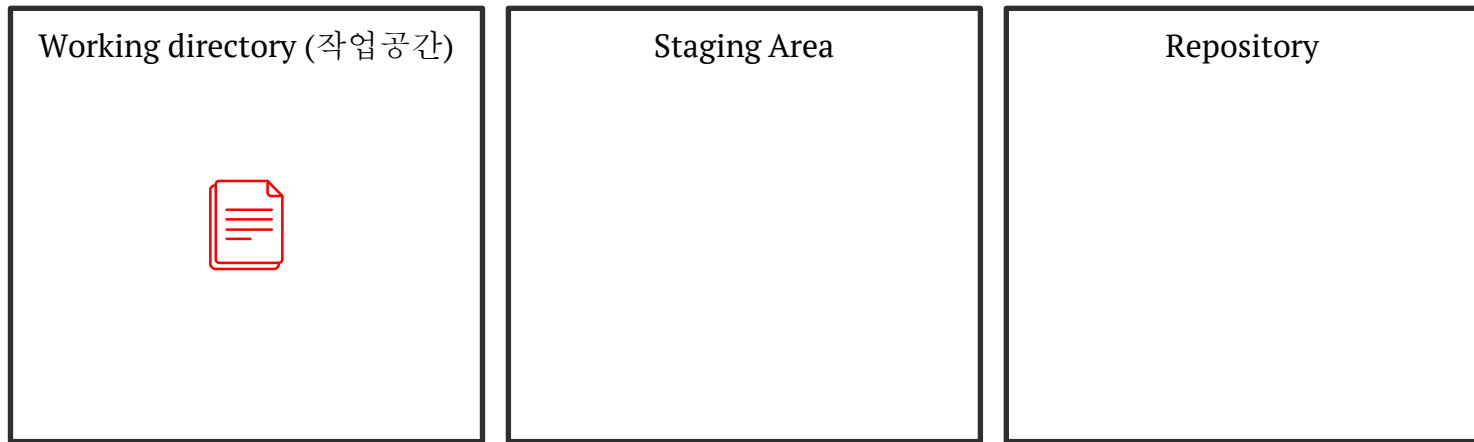
버전이 만들어지는 두 개의 단계

## 버전이 되기까지 거쳐가는 세 개의 공간



버전이 만들어지는 두 개의 단계

## 버전이 되기까지 거쳐가는 세 개의 공간

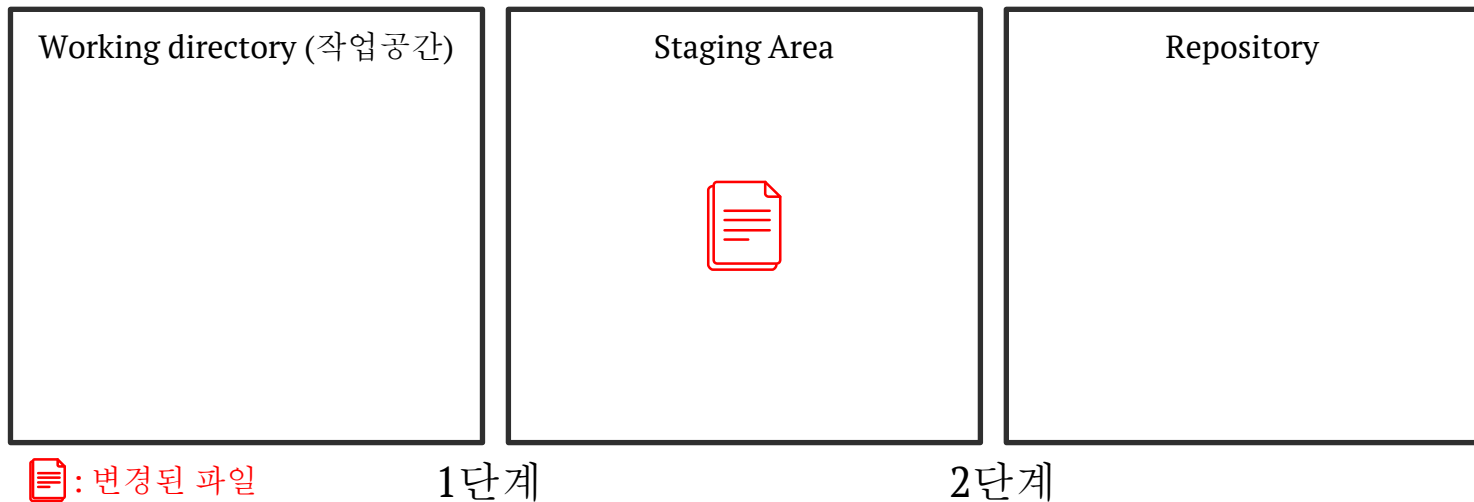


: 변경된 파일

1단계

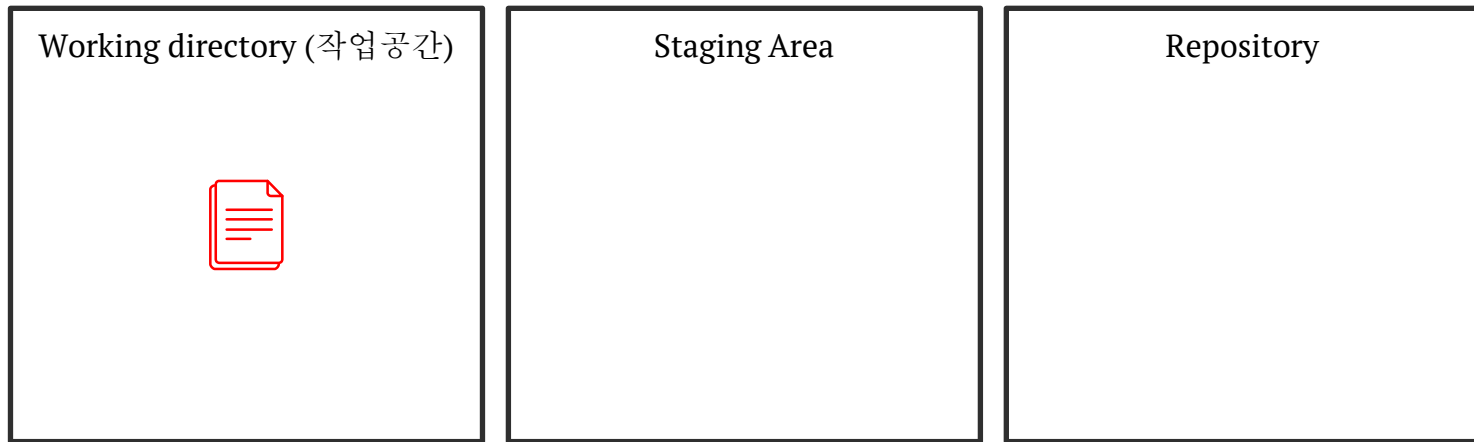
버전이 만들어지는 두 개의 단계

## 버전이 되기까지 거쳐가는 세 개의 공간



버전이 만들어지는 두 개의 단계

명령어 : git <명령어>

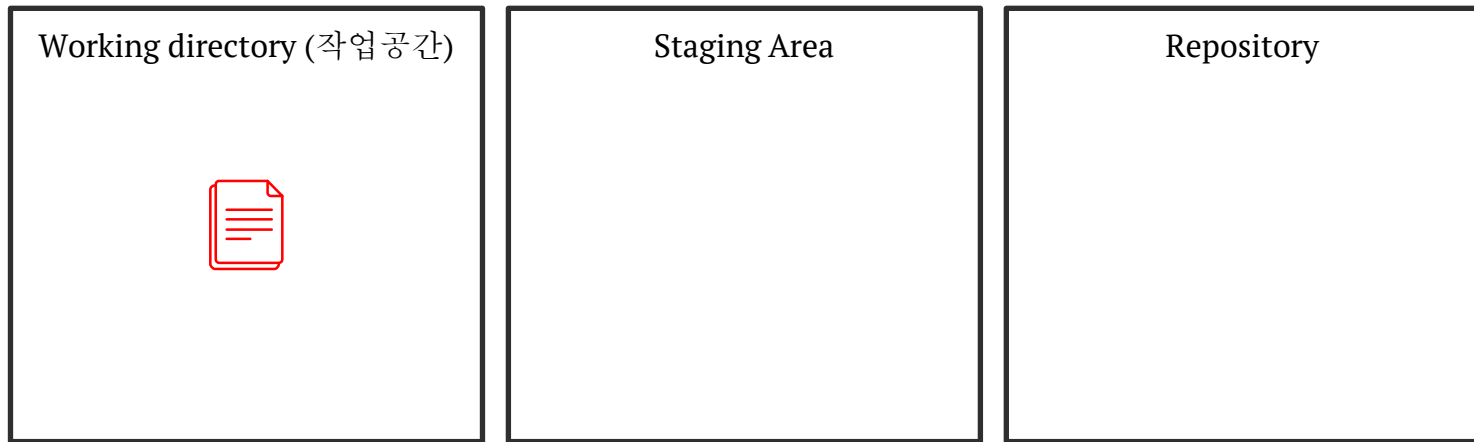


 : 변경된 파일

1단계

버전이 만들어지는 두 개의 단계

명령어 : git add



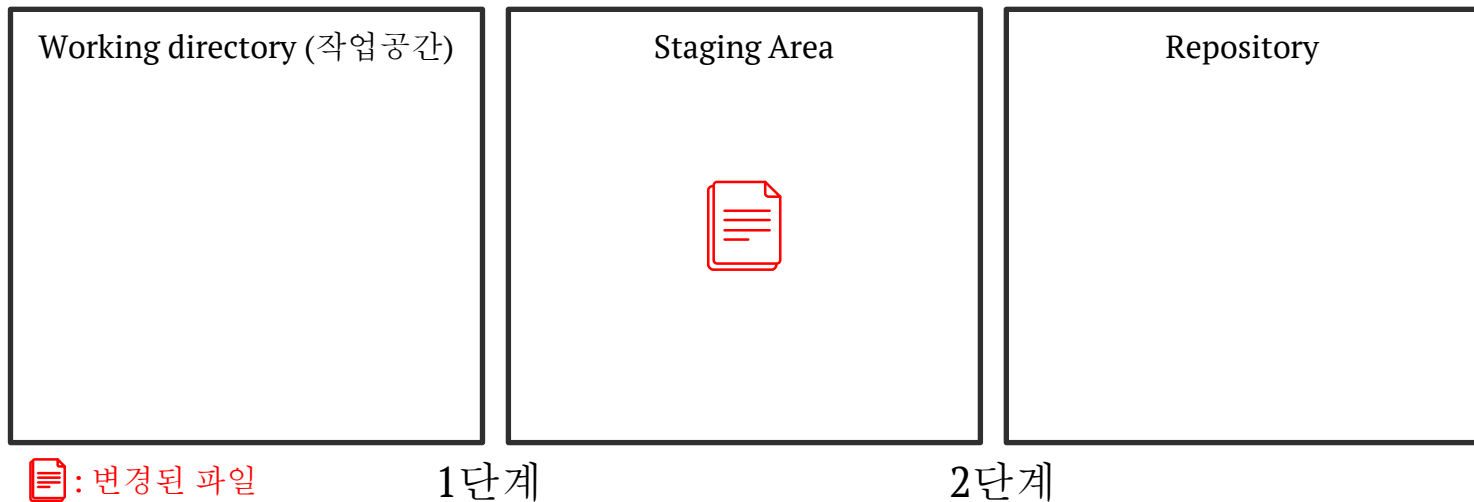
 : 변경된 파일

1단계



버전이 만들어지는 두 개의 단계

명령어 : `git commit -m "commit message"`



## 용어 정리

Commit message??

이 변화가 (이 버전이) 의미하는 것을 담은 메세지  
어떤 유의미한 변화가 이 버전에 담겼는가를 담은 메세지

## 용어 정리



KakaoTalk

Version 4.3.5, 23.0 MB

Nov 5, 2014

OPEN

- Bug fixes



Tumblr

Version 3.7.2, 20.9 MB

Nov 5, 2014

OPEN

Not a big update. You know, bug fixes.

Main one was an annoying thing with iCloud keychains where you couldn't be logged in on more than one device at a time. Annoying! We fixed it.

Okay, bye!


이 밖에

지금부터 이 폴더에서 버전관리 시작할거다!: `git init`

지금 상황은?: `git status`

지금까지의 버전은? : `git log`

뭐가 달라졌어? : `git diff`



실습해봅시다

## 복습해봅시다

Working directory (작업공간)

Staging Area

Repository

Version 3.0

Version 2.0

Version 1.0

## 의문점

`git commit` 명령어를 통해 버전을 만들면  
내 컴퓨터 속 저장소(=local 저장소)에만 저장되는데,

어떻게 다른 사람과 원격으로 협업을 할 수가 있는 거죠?

## 의문점

**git commit** 명령어를 통해 버전을 만들면  
내 컴퓨터 속 저장소 (=local 저장소) 에만 저장되는데,

효율적으로 백업했다고 볼 수 있나요?  
노트북에 커피 한 번 쏟으면 말짱 꽂인데?



## Solution



각자의 컴퓨터에만 존재하는  
버전(Local에서 만들어준 버전)을 저장/관리해주는 서비스  
~~개발자 취업 필수 포트폴리오..~~

## 용어 정리

github에 코드를 업로드한다

## 용어 정리

github에 코드를 업로드한다

## 용어 정리

github에 코드를 **push**한다

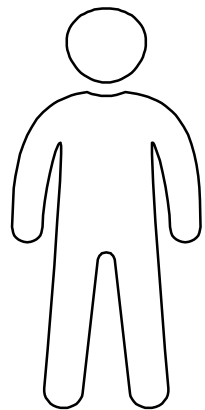
## 용어 정리

github에 코드를 push한다

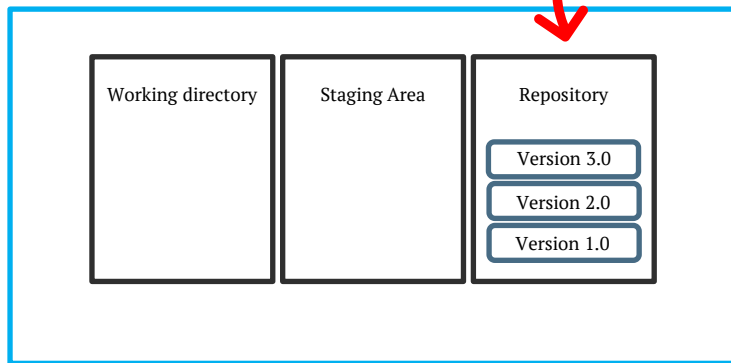
## 용어 정리

원격저장소에 코드를 **push**한다

Github에 코드를 밀어넣는다?



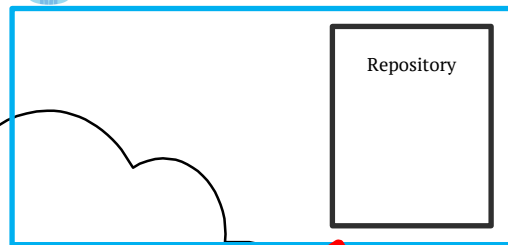
내 컴퓨터



“Local 저장소”



인터넷 세상 어딘가의  
(Github가 관리하는) 컴퓨터

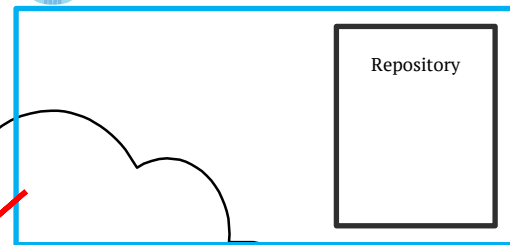
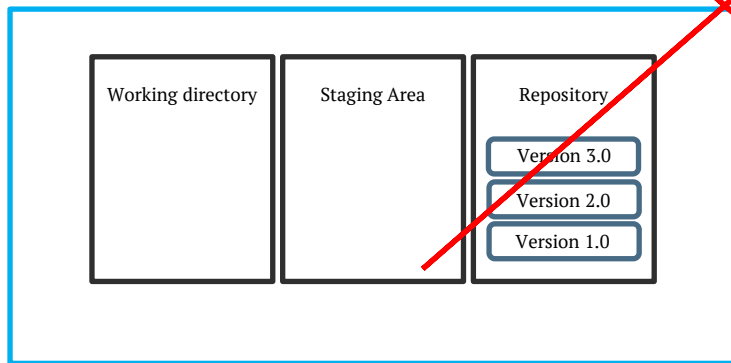
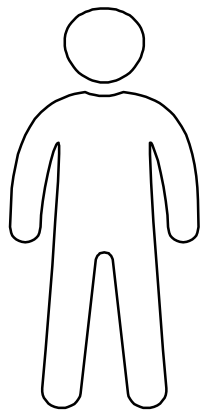


“원격 저장소”

(= remote 저장소)

Github에 코드를 밀어넣는다?

내 컴퓨터



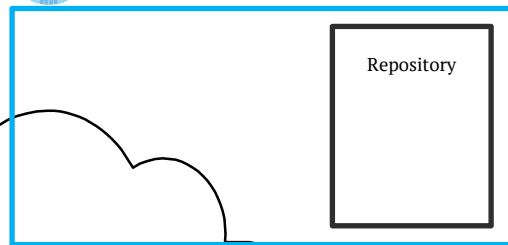
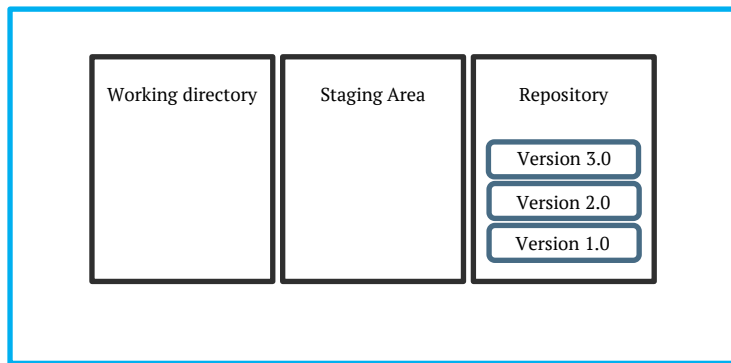
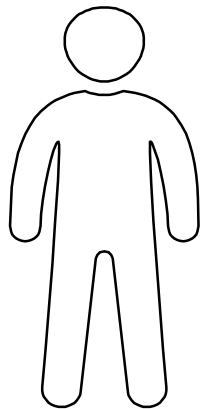
인터넷 세상 어딘가의  
(Github가 관리하는) 컴퓨터





Github에 코드를 밀어넣는다?

내 컴퓨터

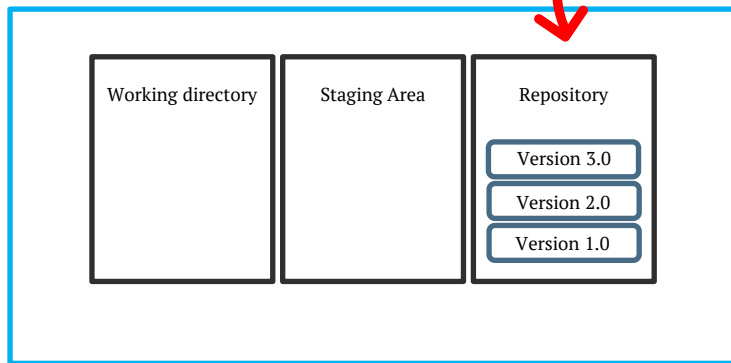


인터넷 세상 어딘가의  
(Github가 관리하는) 컴퓨터

Github에 코드를 밀어넣는다?



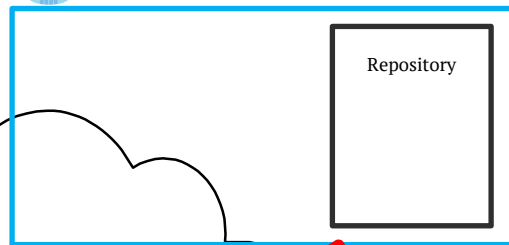
내 컴퓨터

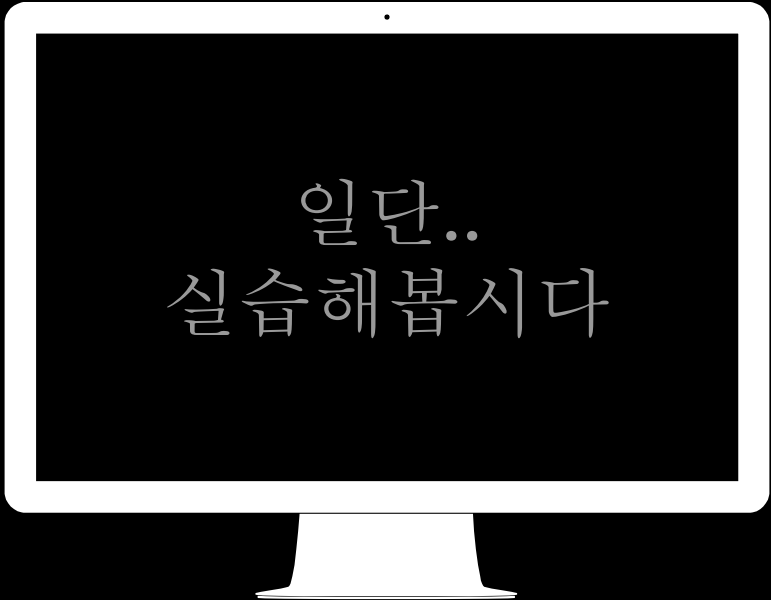


“Local 저장소”

인터넷 세상 어딘가의  
(Github가 관리하는) 컴퓨터

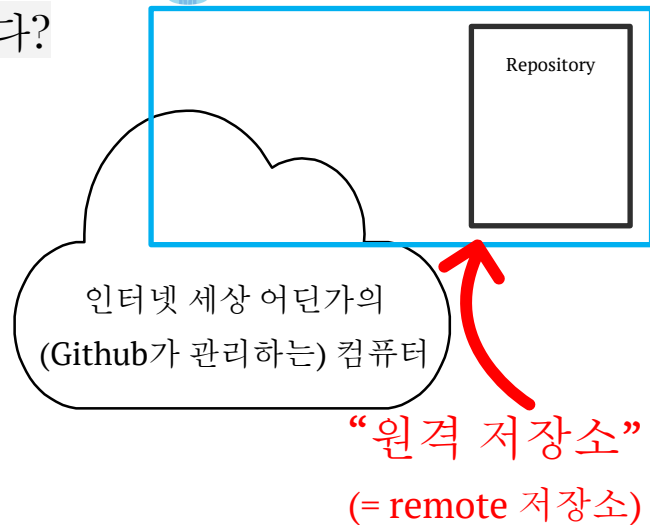
“원격 저장소”  
(= remote 저장소)





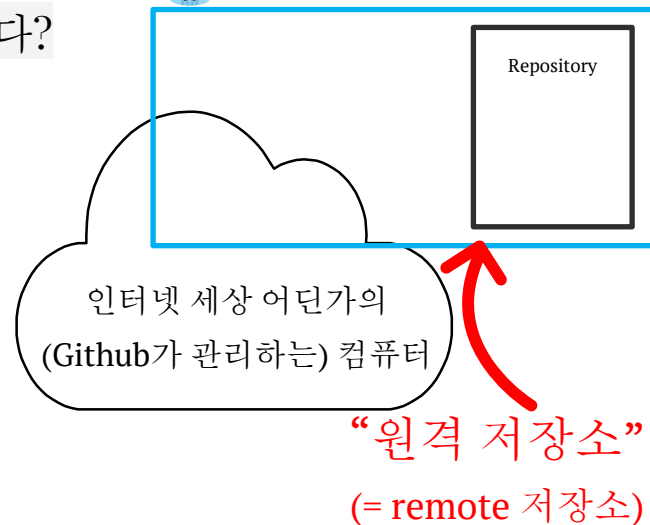
일단..  
실습해봅시다

Github에 코드를 밀어넣는다?



내 컴퓨터(local repository)는 지금 여기, 내 눈앞에 있는데,  
인터넷 세상 어딘가에 있는 원격 저장소(remote repository)는  
어디에 있는줄 어떻게 알고  
내 컴퓨터의 코드를 push하지?

Github에 코드를 밀어넣는다?



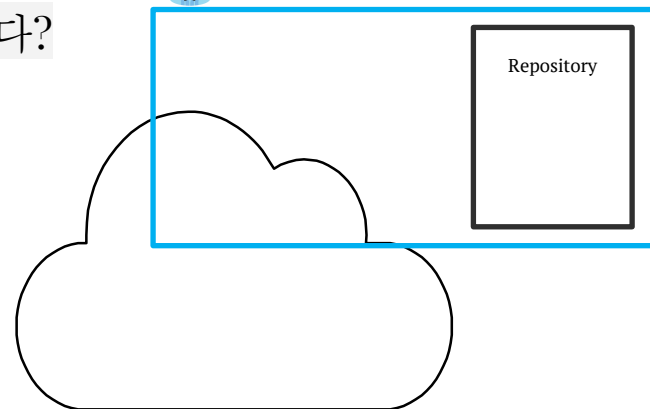
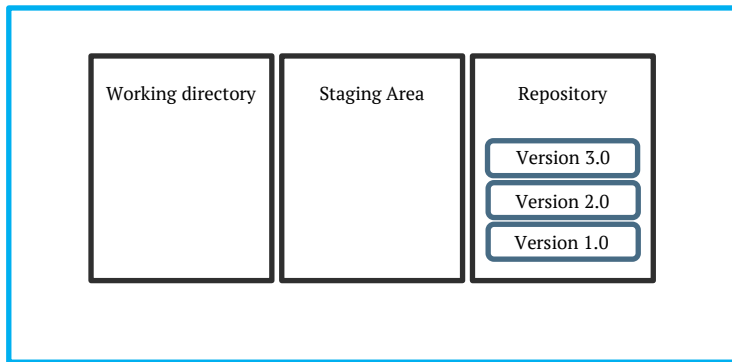
원격 저장소에 push하려면 원격 저장소 **등록**을 해 주어야 한다

```
$ git remote add <이름> <원격 저장소의 URL>
```

Github에 코드를 밀어넣는다?

이제부터 저 원격저장소랑  
**origin**이라는 이름으로  
상호작용할거야

내 컴퓨터



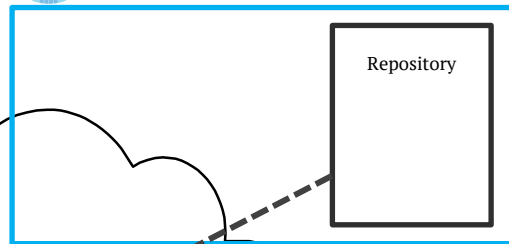
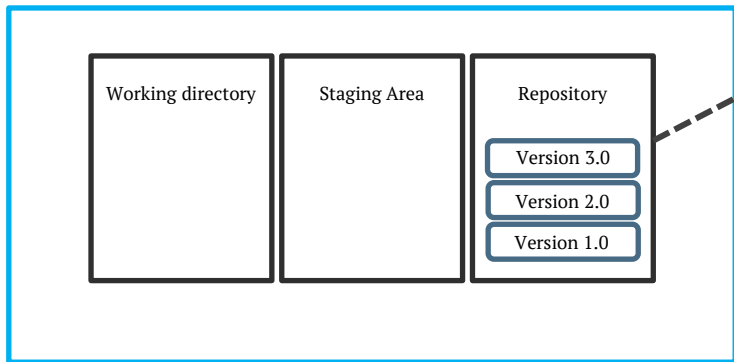
<https://github.com/solux/solux-test.git>



Github에 코드를 밀어넣는다?

이제부터 저 원격저장소랑  
**origin**이라는 이름으로  
상호작용할거야

내 컴퓨터



<https://github.com/solux/solux-test.git>

```
$ git remote add origin https://github.com/solux/solux-test.git
```

## 원격 저장소와의 상호작용

내 컴퓨터와 연결된 원격 저장소 **조회하기**

내 저장소 내용 원격으로 **밀어넣기**

원격 저장소 내용 내 저장소로 **당겨오기**

원격 저장소 내용 내 저장소로 **복사하기**



## 원격 저장소와의 상호작용

내 컴퓨터와 연결된 원격 저장소 **조회하기**: git **remote**

내 저장소 내용 원격으로 **밀어넣기**: git **push**

원격 저장소 내용 내 저장소로 **당겨오기**: git **pull**

원격 저장소 내용 내 저장소로 **복사하기**: git **clone**



실습해봅시다

“

*branch* 나누기

“

*Branch* 를 나눈다

==

여기서 *분기*를 나눈다

“

그럼 분기를 왜 나누지?



“

Git을 이용한 버전관리의 원칙!

1. 분기를 나누고
2. 각자 작업해서
3. 합친다!

“

기능을 시험삼아 추가해보려고

1. 분기를 나누고
2. 각자 작업해서
3. 합친다!

“

기능을 시험삼아 추가해보려고  
여기서부터 역할을 나누어 협업하려고  
등등

1. 분기를 나누고
2. 각자 작업해서
3. 합친다!

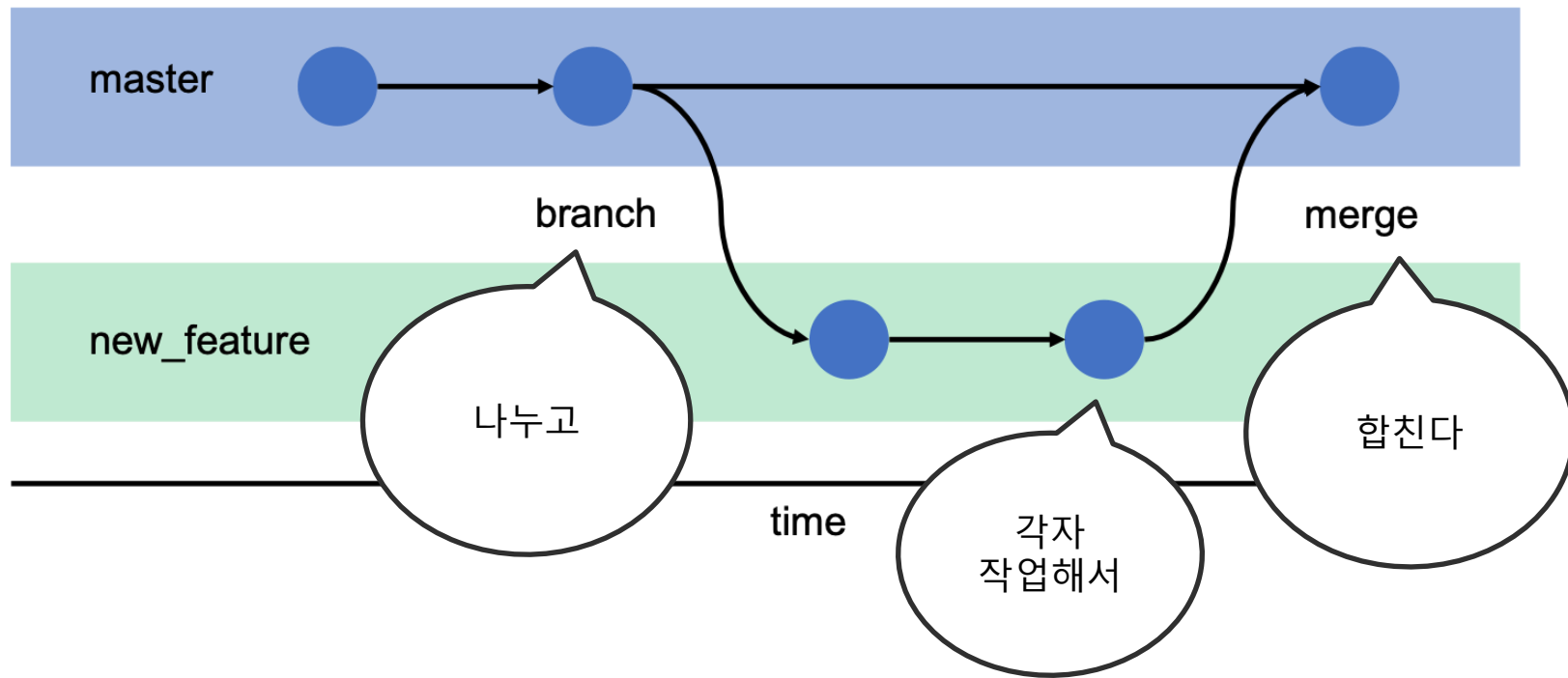




“

기능을 시험삼아 추가해보려고  
여기서부터 역할을 나누어 협업하려고  
등등

## Branch가 필요한 이유



## Branch 명령어

“나누고”

\$ git **branch** [branch 이름]

하나라도 버전이 있는 상태에서 **branch** 나누어야 합니다

## Branch 명령어

“들어가서 각자 작업 후”

```
$ git checkout [branch 이름]
```

```
$ git checkout -b [branch 이름]
```

## Branch 명령어

“합친다”

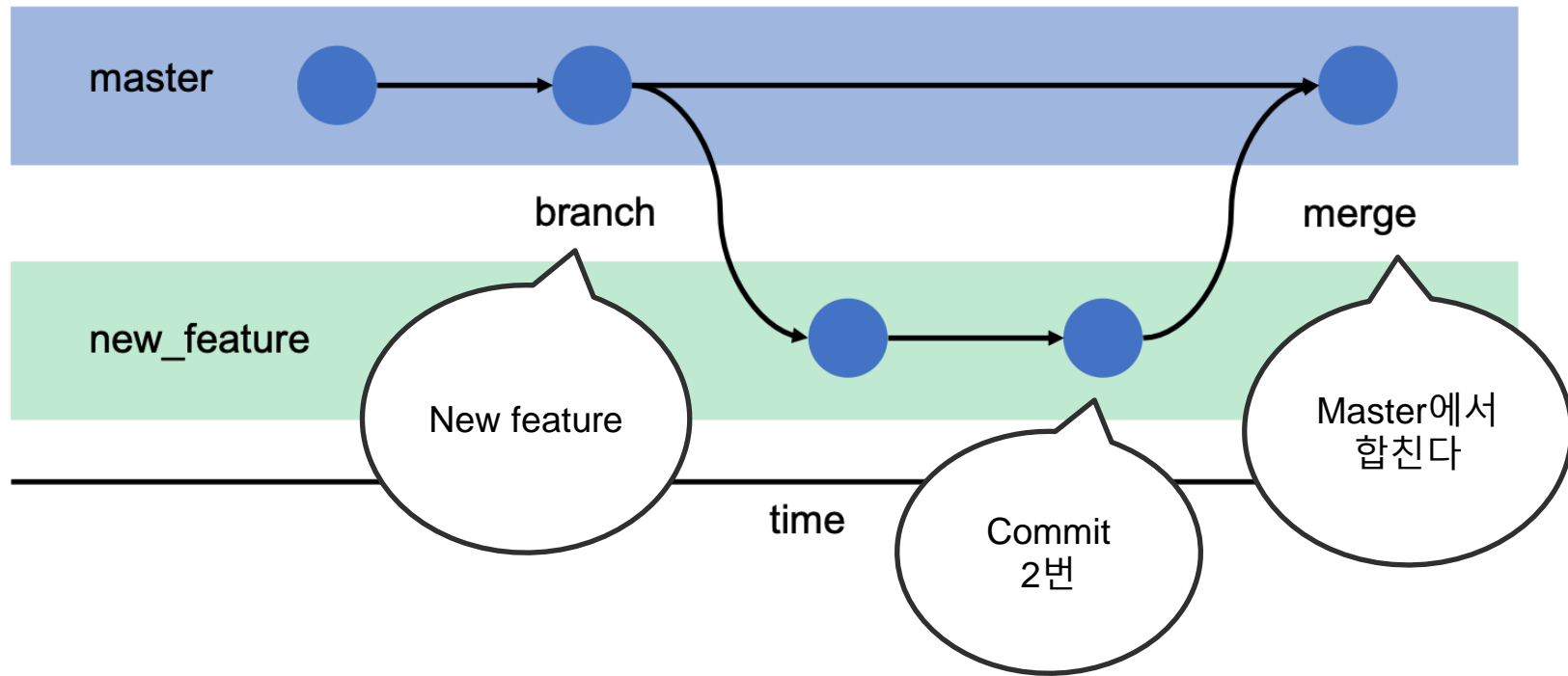
\$ git merge [branch 이름]

“재 나한테 합쳐줘”



실습해봅시다

## Branch가 필요한 이유

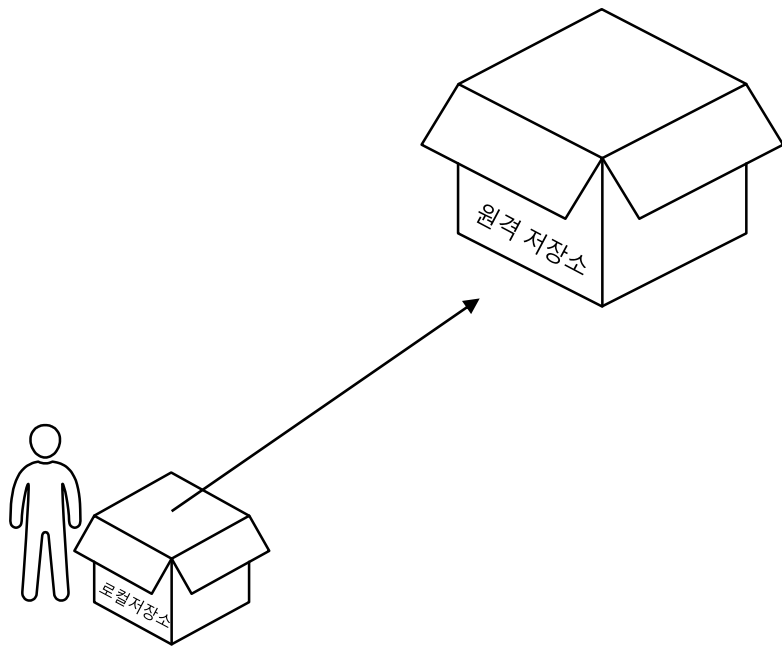


“

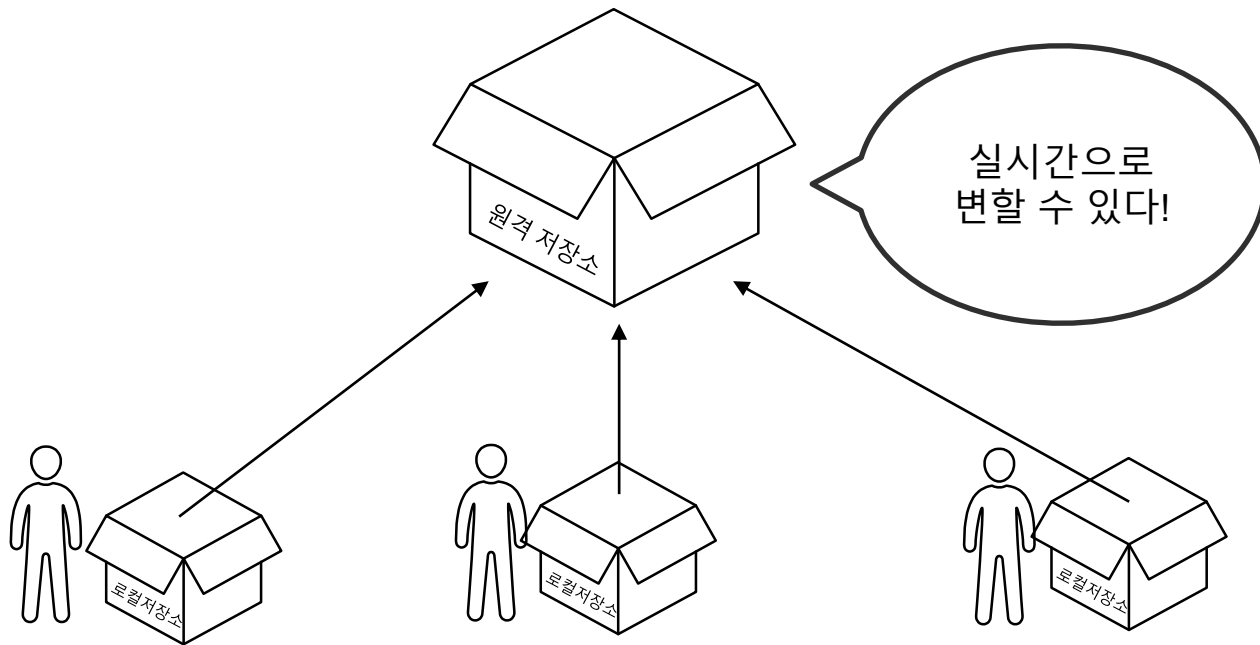
*협업의 원리를 시나리오별로 알아봅시다*



## 나 혼자 쓰는 Github (원격저장소)



## 여럿이 같이 쓰는 Github (원격저장소)



## 협업의 세 가지 시나리오

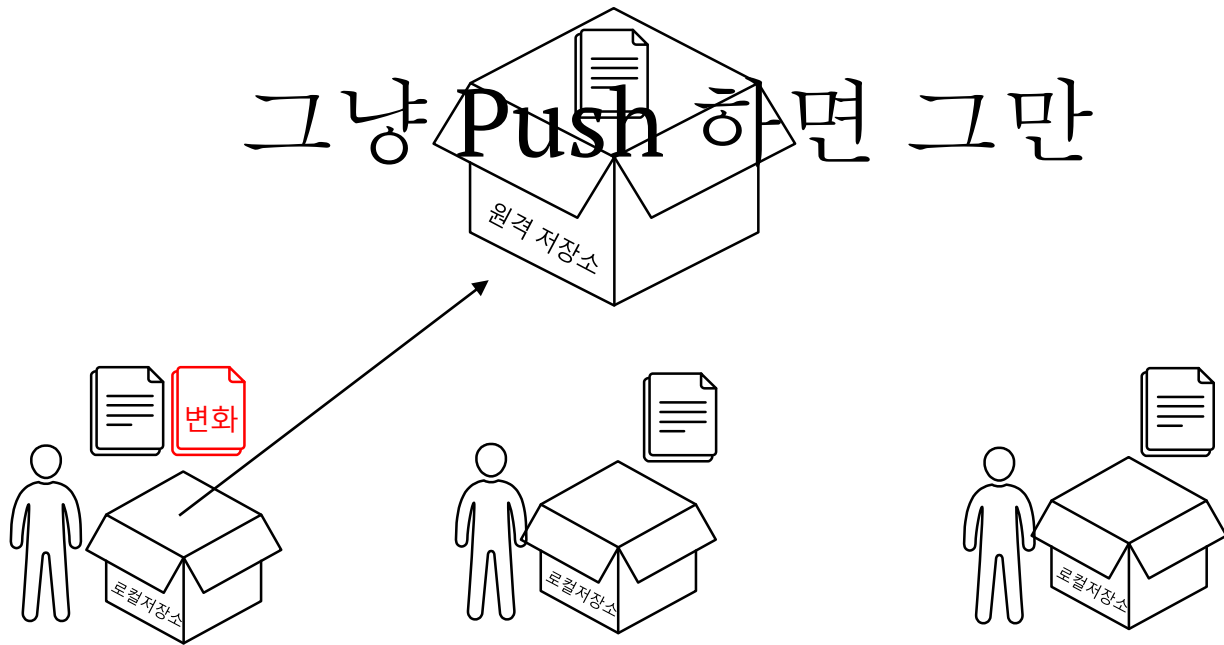
내 로컬 저장소는 **변했는데** 원격 저장소는 **변함 없는 경우**

내 로컬 저장소는 **변함 없는데** 원격 저장소는 **변한 경우**

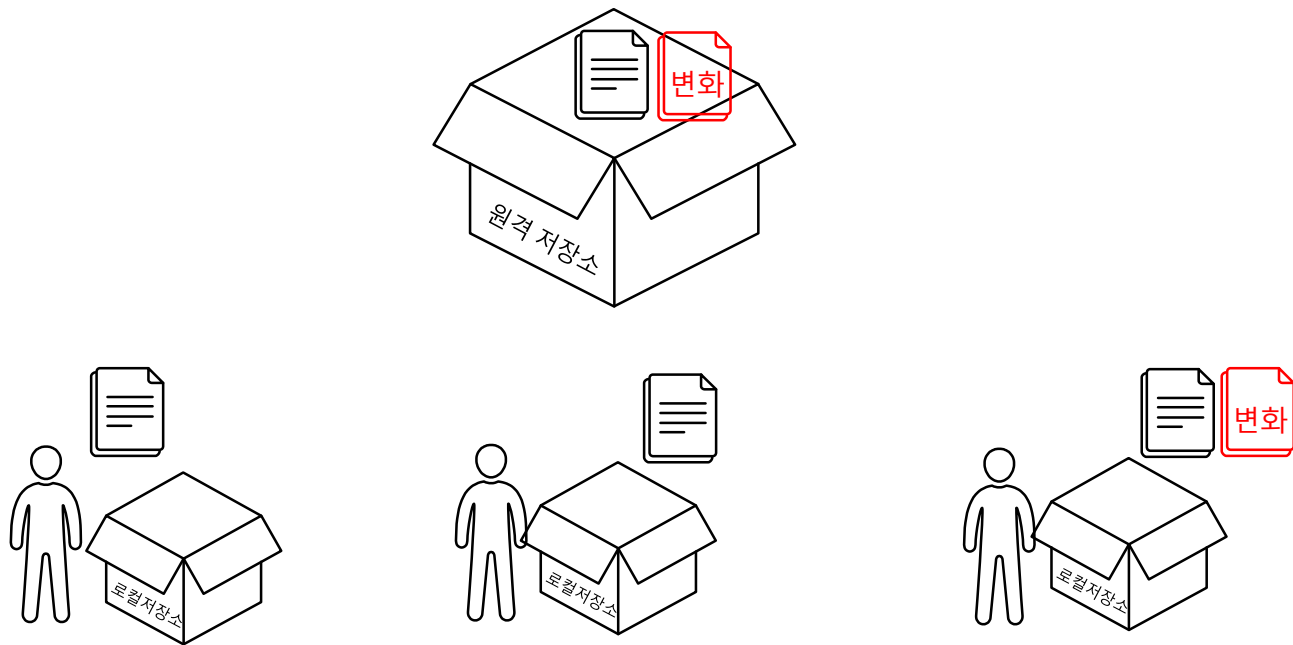
내 로컬 저장소도 **변했는데** 원격 저장소도 **변한 경우**

내 로컬 저장소는 변했는데 원격 저장소는 변함 없는 경우

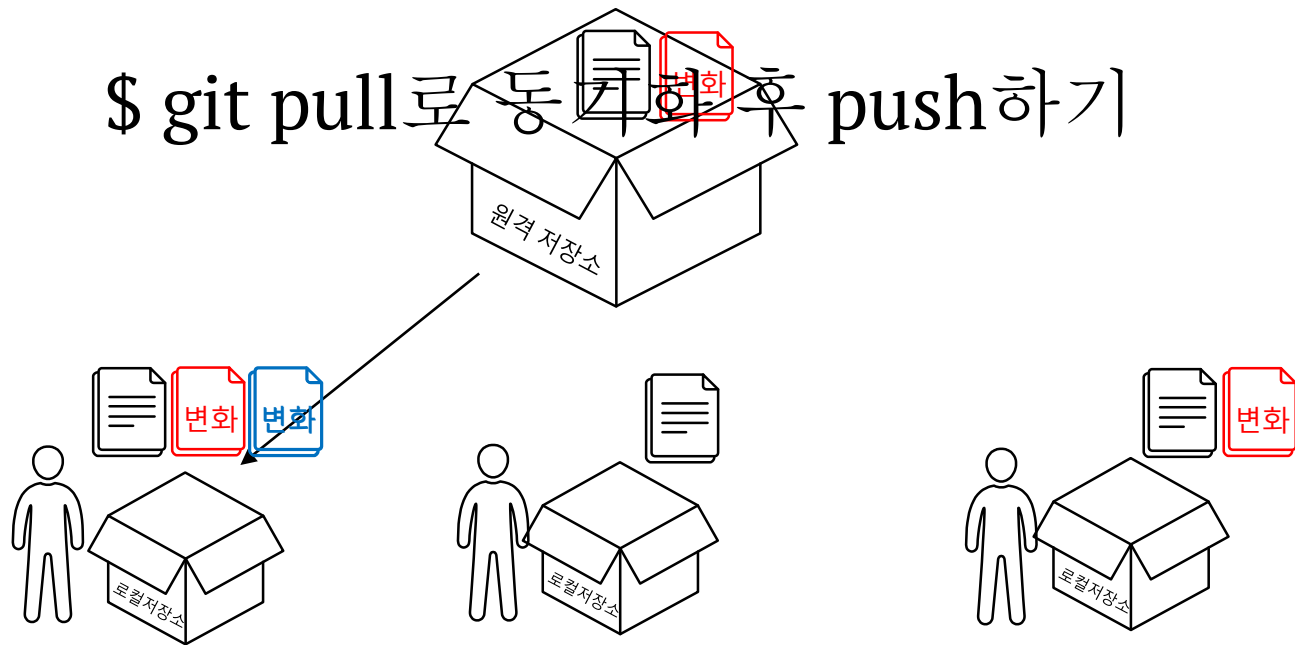
그냥 Push 하면 그만



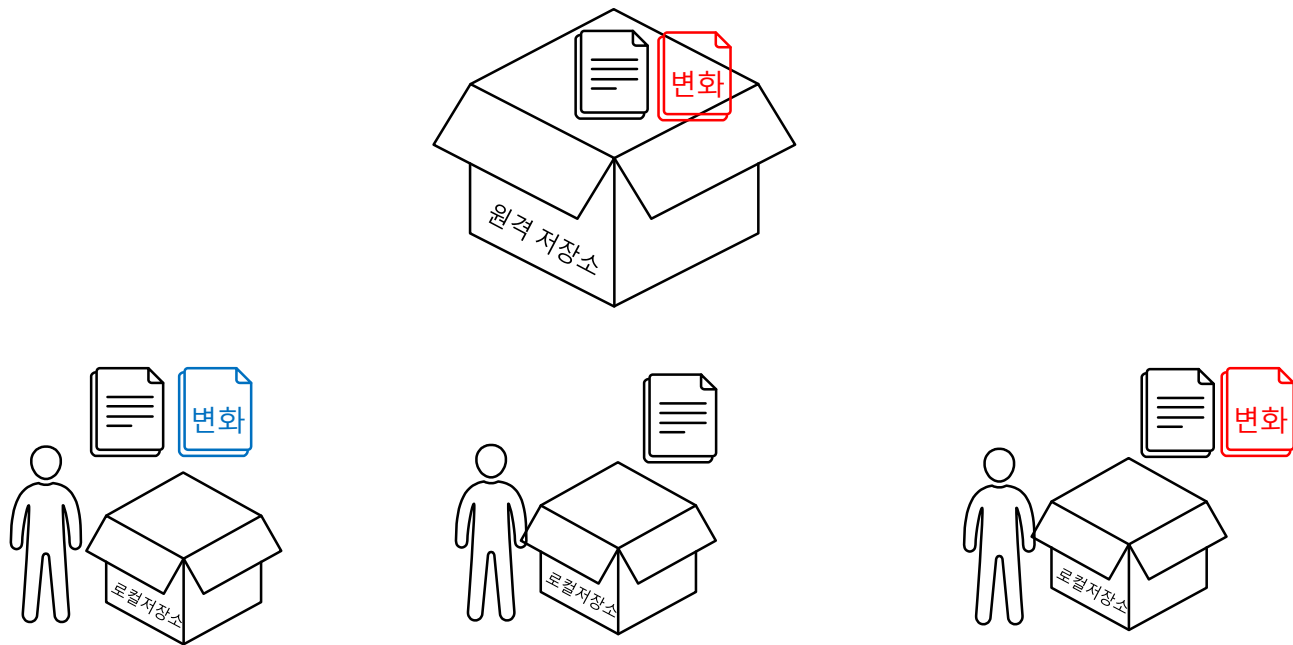
내 로컬 저장소는 변함없는데 원격 저장소는 변한 경우



내 로컬 저장소는 변함없는데 원격 저장소는 변한 경우



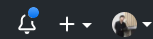
## 내 로컬 저장소 변했는데 원격 저장소도 변한 경우



내 로컬 저장소 변했는데 원격 저장소도 변한 경우

1. Collaborator
2. pull request 를 통한 merge
- ~~3. rebase~~



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)[kangtegong / develup-javascript](#)[Unwatch](#)

3

[Unstar](#)

11

[Fork](#)

9

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)[Options](#)[Manage access](#)[Security & analysis](#)[Branches](#)[Webhooks](#)[Notifications](#)[Integrations](#)[Deploy keys](#)[Secrets](#)[Actions](#)[Moderation](#)[Interaction limits](#)

## Who has access

### PUBLIC REPOSITORY



This repository is public and visible to anyone.

[Manage](#)

### DIRECT ACCESS



0 collaborators have access to this repository. Only you can contribute to this repository.

## Manage access



You haven't invited any collaborators yet

[Invite a collaborator](#)

내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request

## 2. Pull Request 보내기

== branch 나눠서 작업 후 요청 보내기

## 내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request

1. 내 계정으로 **깃고온다**
2. 내 컴퓨터로 **코드를 받아온다**
3. 새로운 branch (**분기**)만든다
4. **내가 만든 branch**에서 작업을 수행한다
5. **내 계정에 push**한다
6. Pull request

## 내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request

1. **Fork**
2. 내 컴퓨터로 **코드를 받아온다**
3. 새로운 **branch (분기)** 만든다
4. **내가 만든 branch**에서 작업을 수행한다
5. **내 계정에 push**한다
6. Pull request

## 내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request

1. **Fork**
2. `git clone <url>`
3. 새로운 branch (**분기**)만든다
4. **내가 만든 branch**에서 작업을 수행한다
5. **내 계정에 push**한다
6. Pull request

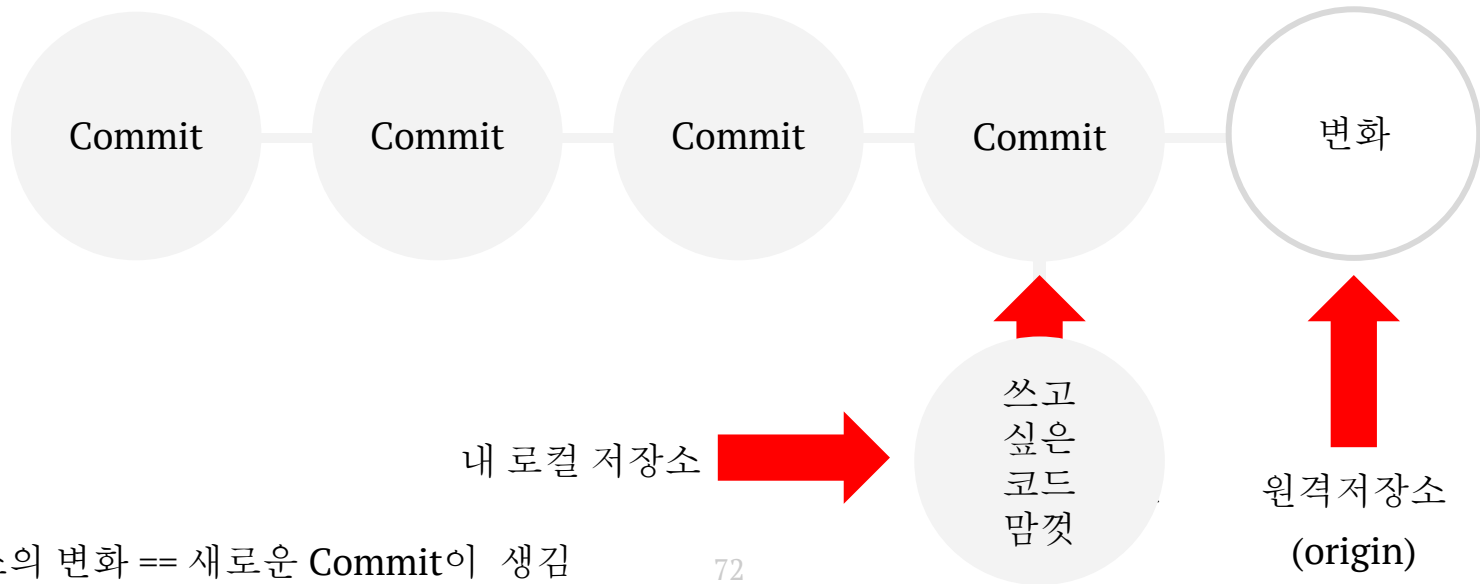
## 내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request

1. **Fork**
2. `git clone <url>`
3. `git branch <my_branch>`
4. 내가 만든 **branch**에서 작업을 수행한다
5. 내 계정에 **push**한다
6. Pull request

## 내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request

1. **fork**
2. `git clone <url>`
3. `git branch <my_branch>`
4. 내가 만든 **branch**에서 작업을 수행한다
5. `git push origin mybranch`
6. Pull request

## 내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request

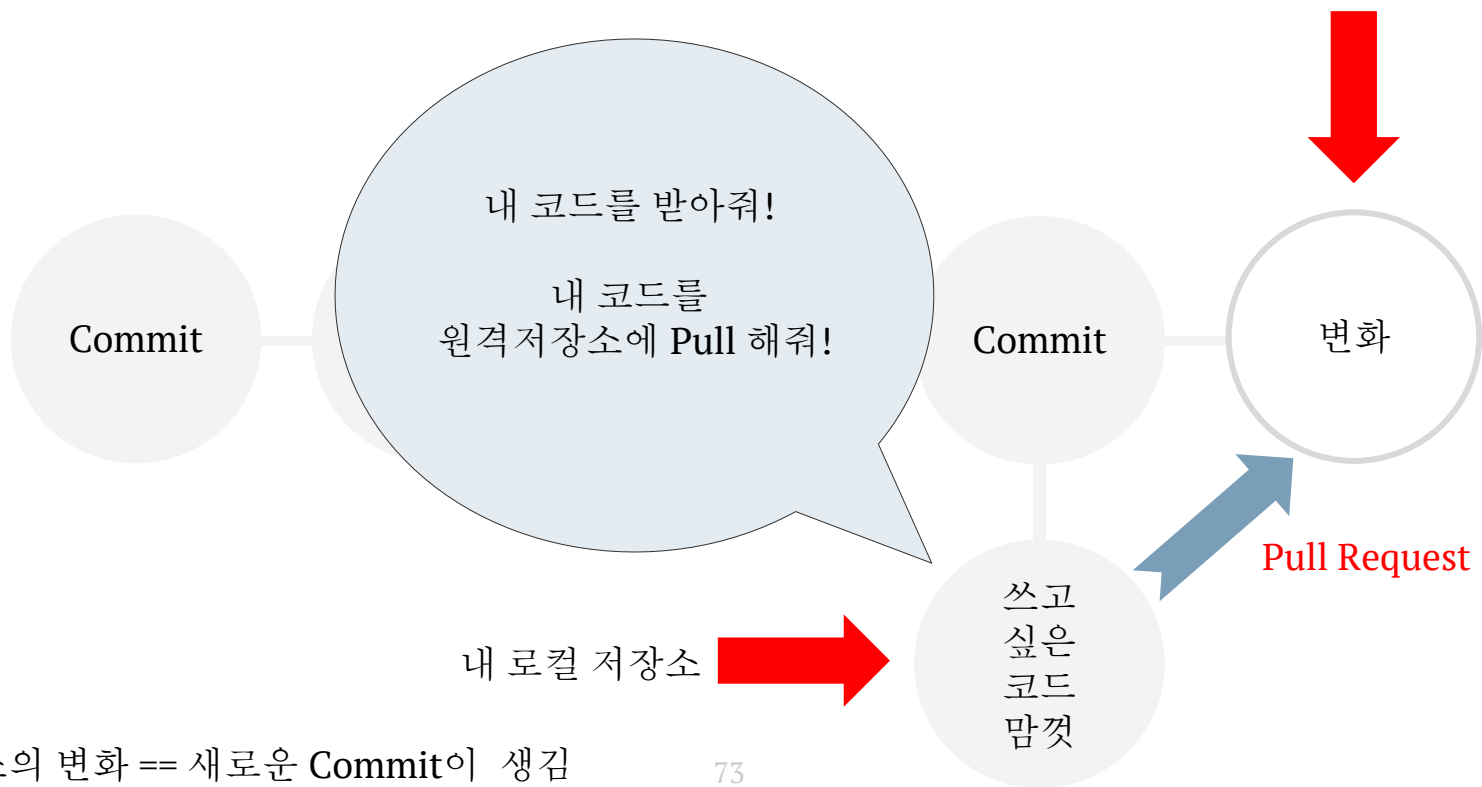


\* 저장소의 변화 == 새로운 Commit이 생김



## 원격저장소 (origin)

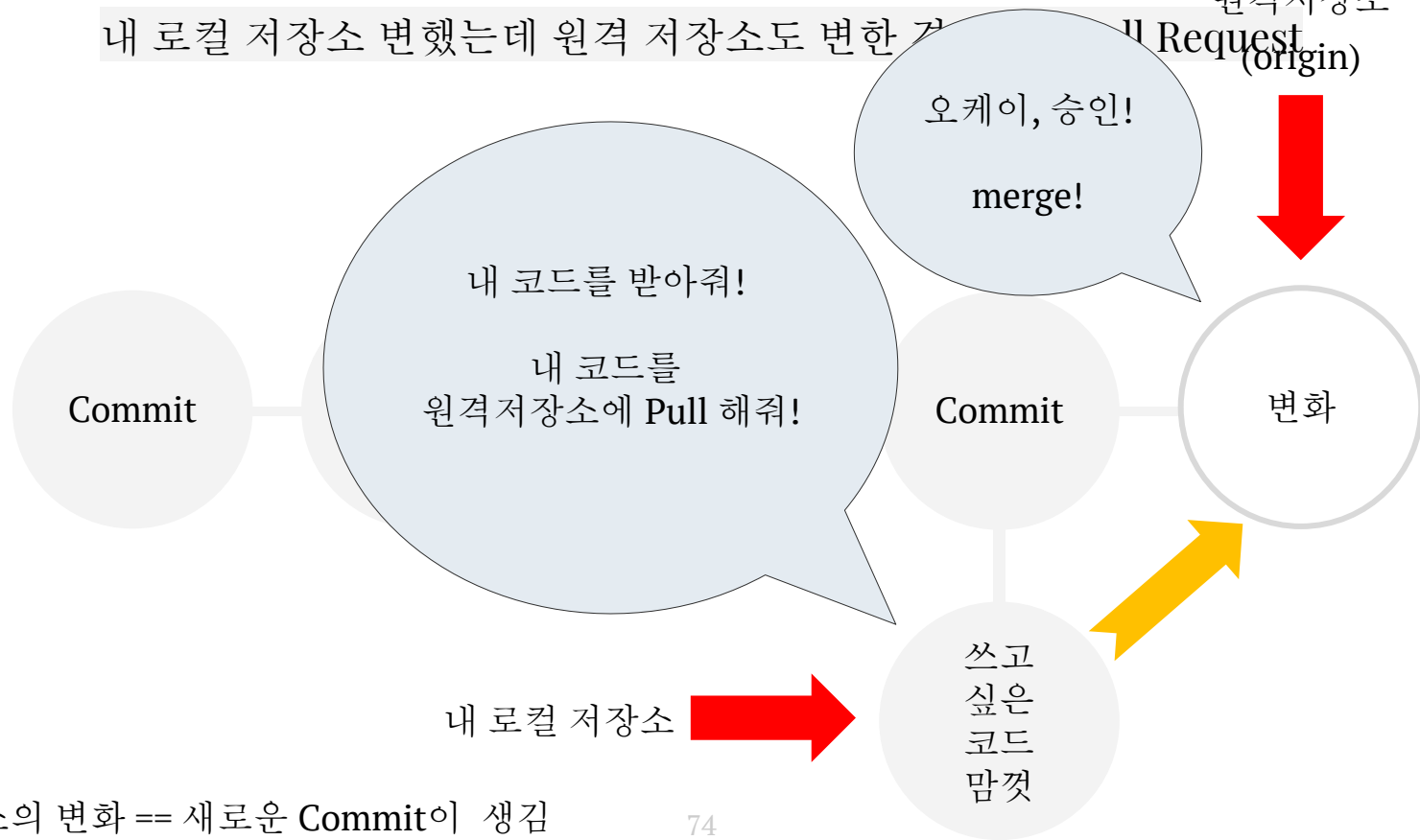
### 내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request



\* 저장소의 변화 == 새로운 Commit이 생김

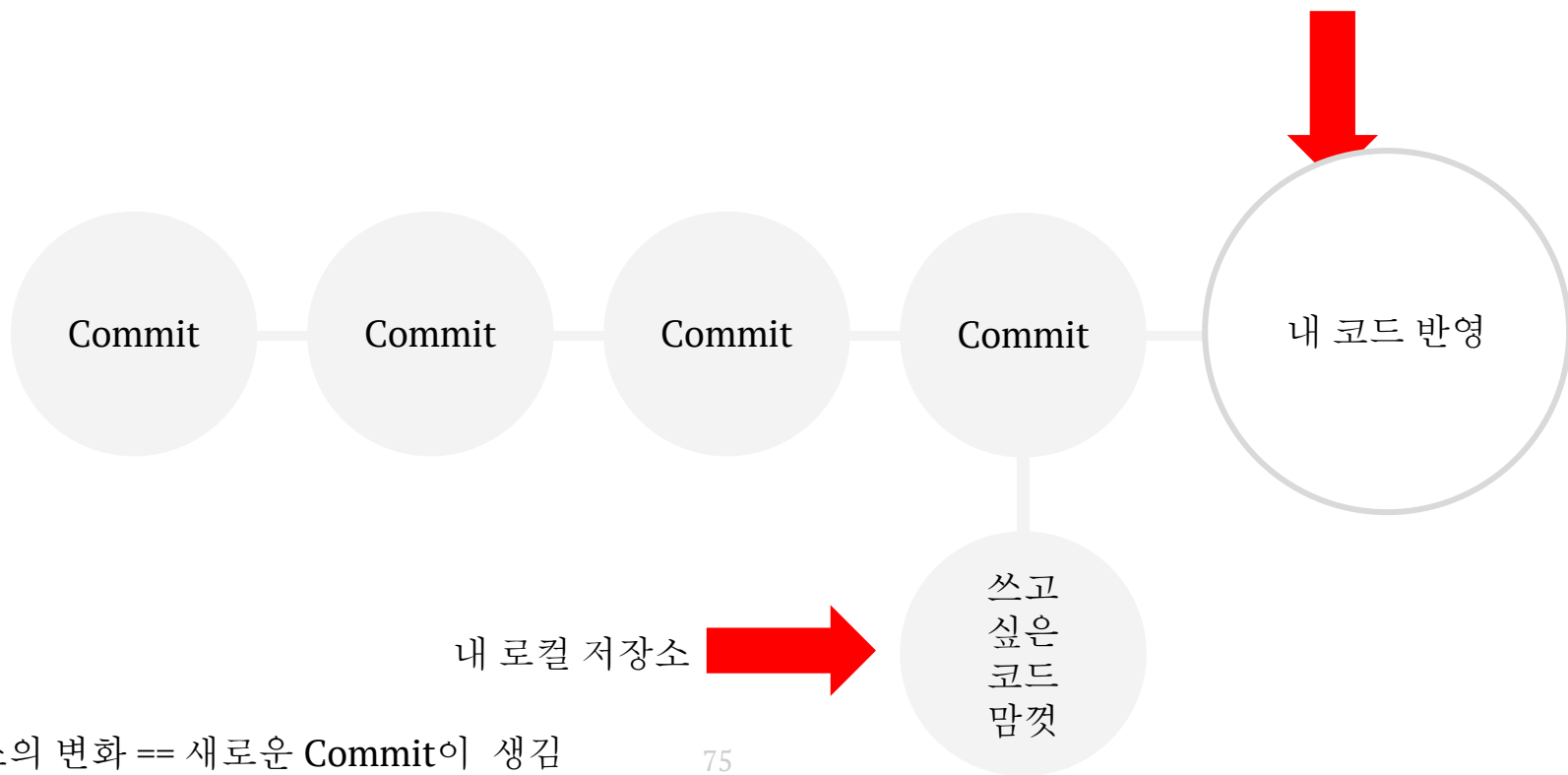
원격저장소  
Pull Request  
(origin)

내 로컬 저장소 변했는데 원격 저장소도 변한 거



\* 저장소의 변화 == 새로운 Commit이 생김

원격저장소  
(origin)  
내 로컬 저장소 변했는데 원격 저장소도 변한 경우 - 2. Pull Request



\* 저장소의 변화 == 새로운 Commit이 생김



실습해봅시다

## 협업의 세 가지 시나리오

내 로컬 저장소는 **변했는데** 원격 저장소는 **변함 없는 경우**

내 로컬 저장소는 **변함 없는데** 원격 저장소는 **변한 경우**

내 로컬 저장소도 **변했는데** 원격 저장소도 **변한 경우**

“

*Add commit push*

*원격 저장소와의 상호작용*

*Branch 나누기*

*Pull request 보내기*

git, github 더 나아가기

git으로 버전 되돌리기

git, github 더 나아가기

추가 학습자료

git 공식 매뉴얼

<https://git-scm.com/book/en/v2>



git, github 더 나아가기

추가 학습자료

강의 자료 및 문의사항

<https://github.com/kangtegong/SOLUX-git>

Lecturer github: <https://github.com/kangtegong>

Email: [tegongkang@gmail.com](mailto:tegongkang@gmail.com)