

# Git / Github

source code management system

**Kosslab**

이호연 (@DanielTimLee)

# | Speaker



**이호연** (Daniel T. Lee)

- Kosslab Software Engineer
- Opensource Developer (Linux Kernel – BPF, ufttrace, etc.)

# How to learn Git?

# | How to learn?

**Git**

+

add

commit

push

pull

reflog

show

status

log

shortlog

diff

clone

checkout

merge

fetch

rebase

reset

stash

blame

cherry-pick

submodule

# | How to learn?

百聞而不如一見

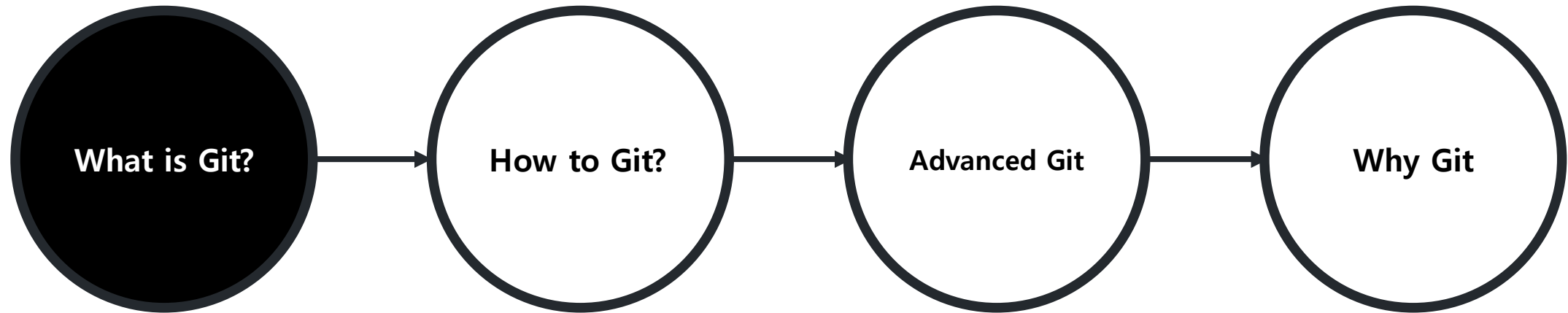
백문이 불여일견  
(백번 듣는 것보다 한 번 보는 것이 낫다)

# How to learn?

百聞而不如一 

백문이 불여일 **깃**  
(백번 듣는 것보다 **일단 써보면서** 이해해자)

# Contents



# What is Git?



# Git 이란?

깃

소프트웨어



깃은 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템이다. 소프트웨어 개발에서 소스 코드 관리에 주로 사용되지만 어떠한 집합의 파일의 변경사항을 지속적으로 추적하기 위해 사용될 수 있다. [위키백과](#)

원작자: [리누스 토르발스](#)

작성 언어: [C](#), [펄](#), [Tcl](#), [파이썬](#)

# Git 이란?

깃  
소프트웨어

???



깃은 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템이다. 소프트웨어 개발에서 소스 코드 관리에 주로 사용되지만 어떠한 집합의 파일의 변경사항을 지속적으로 추적하기 위해 사용될 수 있다. 위키백과

원작자: 리누스 토르발스

작성 언어: C, 펄, Tcl, 파이썬

# | Git 이란?

**Git** == 버전 관리하는 도구

# | Git 이란?

**Git** == 버전 관리하는 도구

- 개발과정, 소스파일 등을 관리
- history 관리: 특정시점으로 복구


# 왜 Git을 써야 될까요?

[illegible]


# 왜 Git을 써야 될까요?


# Version 1




 포스터 시안

## 포스터 시안

 최종완료1 - 복사본 (2)


 최종완료

 진짜최종완료

**Ps 진짜 진짜 최종완라**

**진짜진짜진짜최종완료**


 중간저장3

 중간저장2

**중간저장**


**완료**

**마지막**


 □ L O □ L O

끝

 asdasdasda

 1231231234444444444444444

[illegible]

 123123123123

## Version 2









..??



# Final Version?

?!

# 왜 Git을 써야 될까요?

 포스터 시안  
 포스터 시안  
 최종완료1 - 복사본 (2)  
 최종완료  
 진짜최종완료  
 진짜진짜최종완랴  
 진짜진짜진짜최종완료  
 중간저장3  
 중간저장2  
 중간저장  
 완료  
 마지막  
 □ ◻ ○ □ ◻ ○  
 끝  
 asdasdasda  
 123123123444444444444444444444  
 1111111111111111111111111111111111...  
 123123123123

# 문제점?

- 버전 간 **변경사항**을 알 수 없다
- 버전마다 생성되는 **새로운 파일**..

# 왜 Git을 써야 될까요?

## 문제점?

☑ 버전 간 **변경사항**을 **알 수 없다**

☐ 버전마다 생성되는 **새로운 파일..**

```
def galaxy_order_id
-   xml_body.at_xpath('GalaxyOrderID').try(:text).try(:to_i)
+   xml_body.at_xpath('GalaxyOrderID')&.text&.to_i
end

def errors
@@ -42,8 +46,8 @@ module Egalaxy
class Error
  def self.from_xml(xml)
    new(
-       code: xml.at_xpath('ErrorCode').try(:text).try(:to_i),
-       text: xml.at_xpath('ErrorText').try(:text)
+       code: xml.at_xpath('ErrorCode')&.text&.to_i,
+       text: xml.at_xpath('ErrorText')&.text
    )
  end
end
```



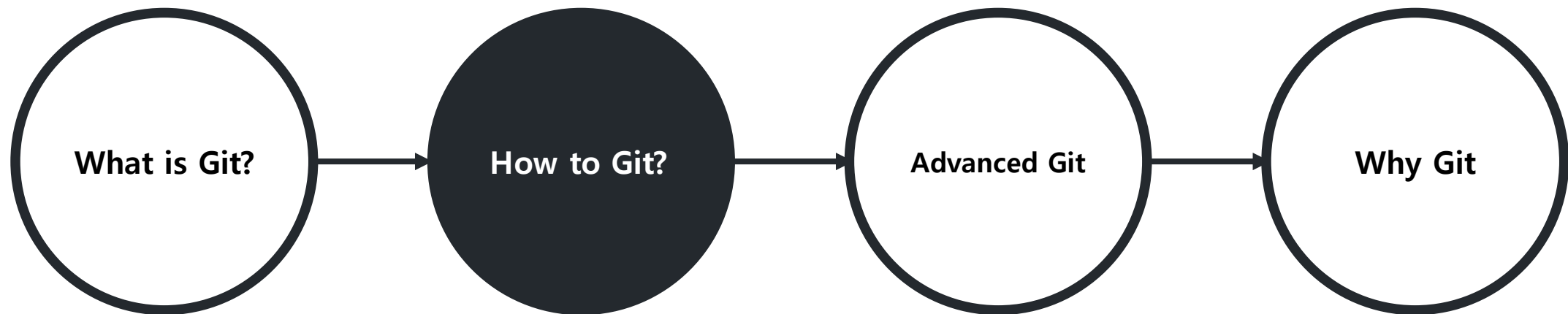
# | 왜 Git을 써야 될까요?

## 문제점?

- ☐ 버전 간 변경사항을 알 수 없다
- ☒ 버전마다 생성되는 **새로운 파일**..



# Contents



# | Git 실습



<https://github.com/danieltl/git-tutorial>

# Git 실습 - 준비하기 1



## 1. 예제소스 다운

<https://git.io/fjdvc>



## 2. Git 설치하기

<https://git-scm.com/downloads>



## 3. Editor 다운받기

<https://code.visualstudio.com/>


<https://atom.io/>



## 4. Github 회원가입


<https://github.com/join>


# Github 회원가입


 Why GitHub? ▾ Enterprise Explore ▾ Marketplace Pricing ▾  / Sign in Sign up

## Join GitHub

The best way to design, build, and ship software.

 **Step 1:**  
Set up your account

 **Step 2:**  
Choose your subscription

 **Step 3:**  
Tailor your experience

### Create your personal account

**Username \***  
  
This will be your username. You can add the name of your organization later.

**Email address \***  
  
We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

**Password \***  
  
Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

#### You'll love GitHub

**Unlimited** public repositories

**Unlimited** private repositories

✓

 Limitless collaboration

✓

 Frictionless development

✓

 Open source community

## 1. 가입 정보 입력

# Github 회원가입



Completed

Set up your account



Step 2:

Choose your subscription



Step 3:

Personalize your experience

## Choose your subscription

With tools developers love and the world's largest open source community, there's no wrong choice.



Free

The basics of GitHub for every developer

\$0

per month

Includes:

- ∞ Unlimited public and private repositories
- ✓ 3 collaborators for private repositories
- ✓ Issues and bug tracking
- ✓ Project management



Pro

Pro tools for developers with advanced requirements

\$7

per month

[\(view in KRW\)](#)




Includes:

- ∞ Unlimited public and private repositories
- ∞ Unlimited collaborators
- ✓ Issues and bug tracking
- ✓ Project management
- ✓ [Advanced tools and insights](#)

Are you a [student](#)? Get access to the best developer tools for free with the [GitHub Student Developer Pack](#).

## 2. 무료 구독 선택

# Github 회원가입

 <b>Completed</b> Set up a personal account	 <b>Step 2:</b> Choose your subscription	 <b>Step 3:</b> Tailor your experience
---	--	--

What is your level of programming experience?

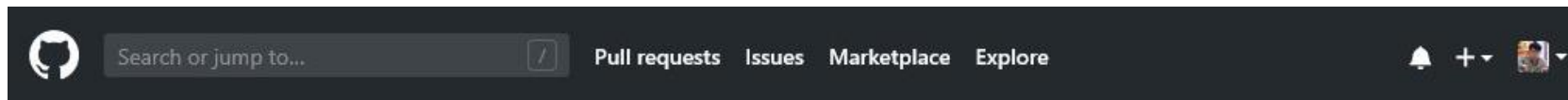
- ☐ None—I don't program at all
- ☐ New to programming
- ☐ Somewhat experienced
- ☐ Very experienced

What do you plan to use GitHub for? (Select up to 3)

- ☐ Learning to code
- ☐ Learning Git and GitHub
- ☐ Host a project (repository)
- ☐ Creating a website with GitHub Pages
- ☐ Collaborating with my team
- ☐ Finding a project to contribute to
- ☐ School work / School-related project
- ☐ The GitHub API
- ☐ I don't know yet
- ☐ Other (please specify)

## 3. 질문 응답

# Github 회원가입



## Please check your email settings

Before you can contribute on GitHub, we need you to verify your email address.

The mailserver for gmail.com is not accepting our messages to . Please check the spelling of your email address and make sure email from GitHub is not rejected by any (spam) filter.

[Send verification email to fsdlksjdfkjk@gmail.com](#) or [change your email settings](#).

### GitHub

#### Subscribe to our newsletter

Get product updates, company news, and more.

[Subscribe](#)

#### Product

[Features](#)

[Security](#)

[Enterprise](#)

[Customer stories](#)

#### Platform

[Developer API](#)

[Partners](#)

[Atom](#)

[Electron](#)

#### Support

[Help](#)

[Community Forum](#)

[Training](#)

[Status](#)

#### Company

[About](#)

[Blog](#)

[Careers](#)

[Press](#)

## 4. 메일 확인하기



# Git 실습 - 준비하기 2



5. [git-tutorial-example.zip](#) 압축풀고  
폴더 열기



6. 편집기 (vscode, atom) 열기

# Git 실습 방법

Git 실습 방법은 ?



하나의 프로그램을 만드는데 Git을 활용하는 **시나리오**



multiply (곱셈 알고리즘 문제)

- 1) 코딩은 Ctrl-c, v로 하고
- 2) Git은 직접 명령어 쳐서 (git-bash)

2588번 - 곱셈

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	128 MB	12721	8249	7622	67.439%

문제

(세 자리 수) × (세 자리 수)는 다음과 같은 과정을 통하여 이루어진다.

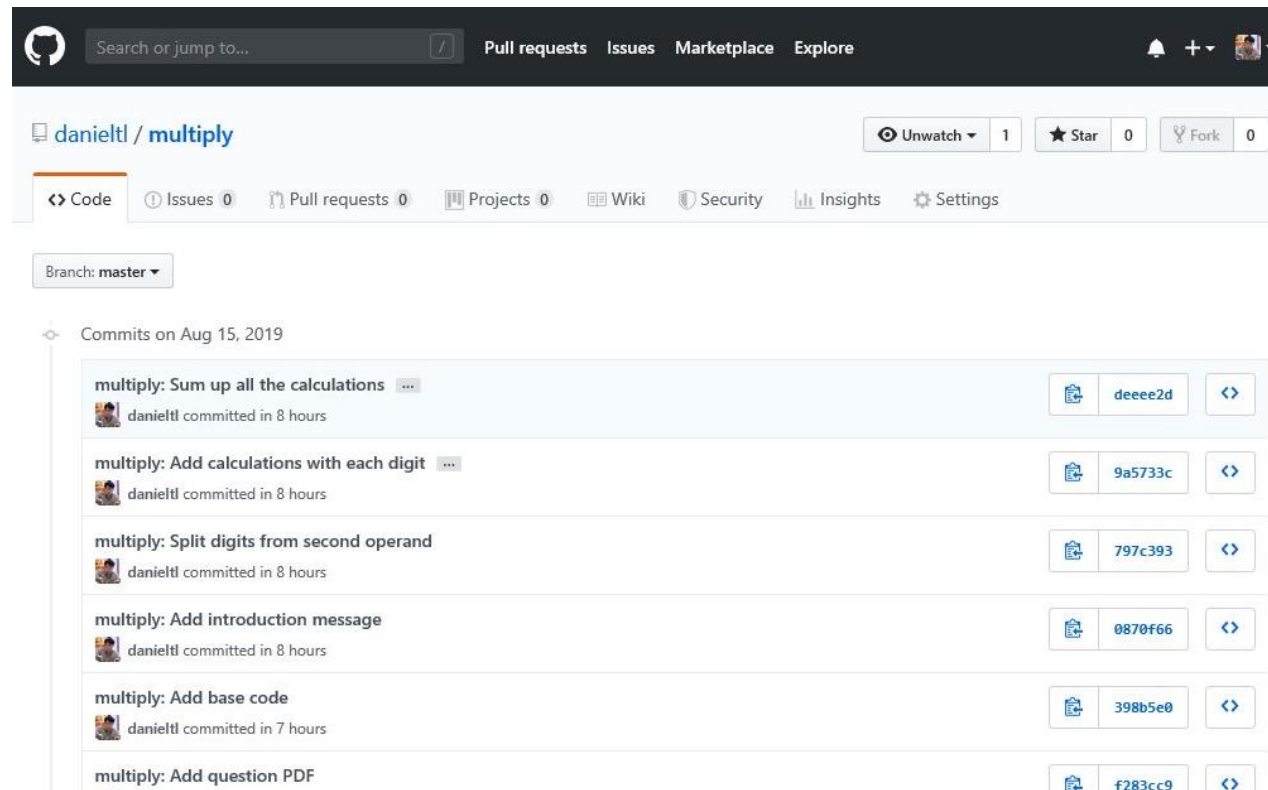
$$\begin{array}{r} 4\ 7\ 2\ \dots\dots\dots (1) \\ \times\ 3\ 8\ 5\ \dots\dots\dots (2) \\ \hline 2\ 3\ 6\ 0\ \dots\dots\dots (3) \\ 3\ 7\ 7\ 6\ \dots\dots\dots (4) \\ 1\ 4\ 1\ 6\ \dots\dots\dots (5) \\ \hline 1\ 8\ 1\ 7\ 2\ 0\ \dots\dots\dots (6) \end{array}$$

(1)과 (2)위치에 들어갈 세 자리 자연수가 주어질 때 (3), (4), (5), (6)위치에 들어갈 값을 구하는 프로그램을 작성하시오.

# Git 실습 방법

Stage 해결하기 위해서 수단과 방법을 가리지말자 !

본인의 Github 계정에 본인의 프로젝트에 commit 기록이 다음과 똑같아 지면 된다.



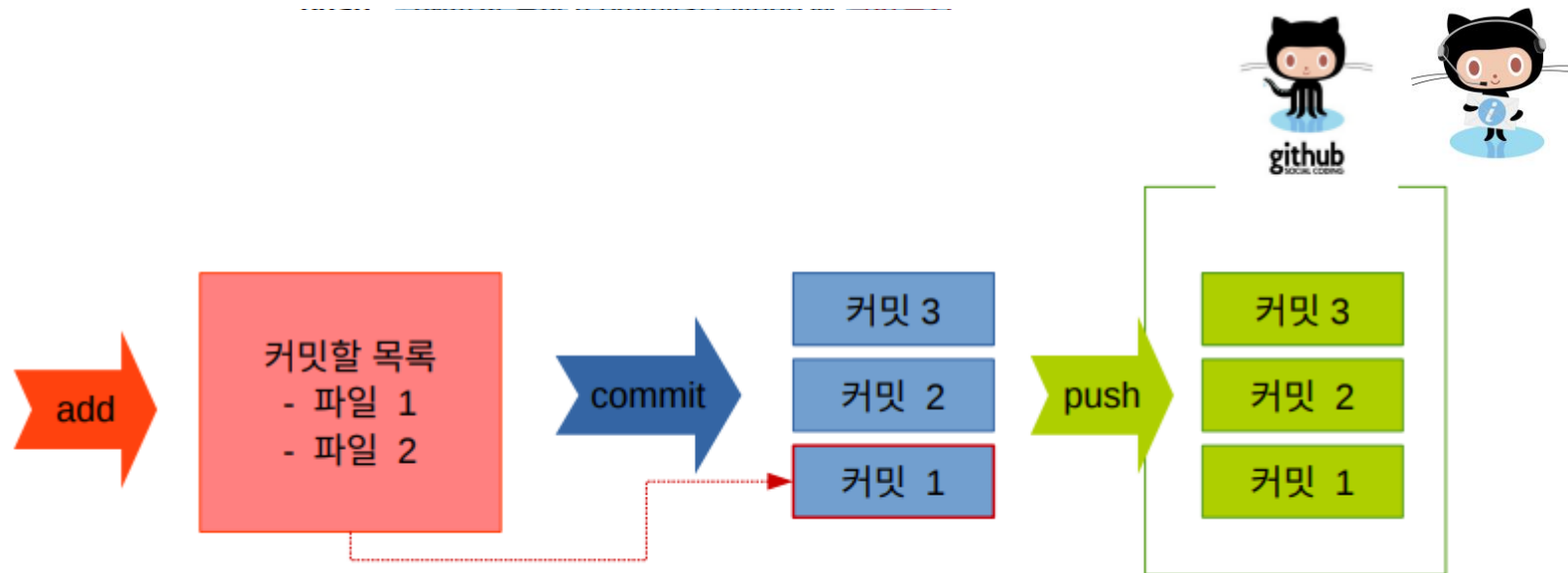
# Git 필수 명령

## <Git 필수 명령>

**add** : 커밋할 목록에 추가

**commit** : commit ( 히스토리의 한단위 ) 만들기

**push** : 현재까지 역사 (commits) Github 에 밀어넣기



# Git 실습 본격적인 실습에 앞서

Prompt, 프롬프트

```
$ git add multiply.pdf
```

Arguments, 인자 (1개)

Git의 Sub-command

```
$ git config --global user.name "Daniel T. Lee"
```

Arguments, 인자 (2개)

--로 시작하면은 보통은 Long Name 옵션

```
$ git commit -s
```

-로 시작하면은 보통은 Short Name 옵션

# Git 실습 기본 설정

- 1) Git-bash 혹은 터미널 실행 후에
- 2) 미리 캐시저장되어 있을지 모를 계정정보 삭제 (처음설치시 생략 가능)

```
$ git config --global --unset credential.helper
```

```
$ git config --system --unset credential.helper
```

- 3) 나의 Github 계정 이메일과 이름 (본인 영문이름, Github 아이디 x)를 적자

```
$ git config --global user.email " 본인메일적으세요 @gmail.com"
```

```
$ git config --global user.name " 본인이름적으세요 Hoyeon Lee"
```

- 4) git-tutorial-example.zip 압축 푼 폴더 열기 (windows 탐색기로)

# Git 실습 Stage 1 초기화 및 첫 commit하기

- 1) git-bash 또는 터미널 실행하고 HOME 경로 (~) 로 이동

```
$ cd ~
```

- 2) multiply 폴더 만들고 window 폴더탐색기로도 열어두기

```
$ mkdir multiply
```

- 3) 경로 이동 ( **pwd** 명령어로 현재경로 확인하기 )

```
$ cd multiply
```

- 4) 해당 폴더를 git 초기화 (**ls -A** 명령어로 생성된 .git 폴더 확인하기 )

```
$ git init
```

- 5) 프로그램 문제 PDF 파일 추가 ( **커밋할 목록에 추가** add) (commit1 폴더내 파일 활용 )

```
$ git add multiply.pdf
```

- 6) 첫 commit 하기 ( 역사 한단위 만들기 )

```
$ git commit -m "multiply: Add question PDF"
```

Git 상태확인 명령어

( 중간중간 치면서 수시로 확인하자 )

```
$ git show
```

```
$ git log
```

```
$ git shortlog
```

```
$ git diff
```

```
$ git status
```

# Git 실습 Stage 1 초기화 및 첫 commit하기

- 7) 소스코드 추가하기 ( 커밋할 목록에 추가 add) (commit2 폴더내 파일 활용 )

```
$ git add multiply.c
```

- 8) commit 하기 ( 역사 한단위 만들기 )

```
$ git commit -m "multiply: Add base code"
```

Git 상태확인 명령어  
( 중간중간 치면서 수시로 확인하자 )

```
$ git show  
$ git log  
$ git shortlog  
$ git diff  
$ git status
```



# Git 실습 Stage 2 diff 사용과 추가 commit하기

1) 상태를 확인한다

```
$ git status
```

2) **commit3** 폴더에 있는 multiply.c 소스 파일로 수정 , 덮어쓰기 후 확인

```
$ git diff
```

3) diff 를 통해서 변화분을 확인했다면 add 진행

```
$ git add multiply.c
```

4) 준비된 소스파일을 commit 한다

```
$ git commit -m "multiply: Add introduction message"
```

5) 지금까지한 3 개의 commit 들을 확인하자

```
$ git log
```

# Git 실습 Stage 2 diff 사용과 추가 commit하기

- 5) **commit4** 폴더에 있는 multiply.c 소스 파일로 수정 , 덮어쓰기 후 확인

```
$ git diff
```

- 6) diff 를 통해서 변화분을 확인했다면 add 진행

```
$ git add multiply.c
```

- 7) 준비된 소스파일을 commit 한다

```
$ git commit -m "multiply: Split digits from second operand"
```

- 8) 지금까지한 **4 개의 commit** 들을 확인하자

```
$ git log
```

# Git 실습 Stage 3 commit에 본인 서명 추가하기

- 1) **commit5** 폴더에 있는 multiply.c 소스 파일로 수정 , 덮어쓰기 후 확인

```
$ git diff
```

- 2) diff 를 통해서 변화분을 확인했다면 add 진행

```
$ git add multiply.c
```

- 3) 서명과함께 commit 한다 . (-s 옵션으로 서명을 포함한다)

```
$ git commit -sm "multiply: Add calculations with each digit"
```

\* 서명의 의미는 본 오픈소스의 라이선스를 제대로 이해하고 작성한 commit 이라는 확인서명  
( 주로 리눅스커널에서 commit 을 공식적으로 만들때 많이 이용된다 . )

# Git 실습 Stage 3 commit에 본인 서명 추가하기

\* 반복해서 추가 commit 만든다

4) **commit6** 폴더에 있는 multiply.c 소스 파일로 수정 , 덮어쓰기 후 확인

```
$ git diff
```

5) diff 를 통해서 변화분을 확인했다면 add 진행

```
$ git add multiply.c
```

6) 서명과함께 commit 한다 . (-s 옵션으로 서명을 포함한다)

```
$ git commit -sm "multiply: Sum up all the calculations"
```

# Git 실습 Stage 4 지금까지의 commit 을 push 하자

- 1) 상태를 확인하고 현재 브랜치명 master 를 확인하자

```
$ git status
```

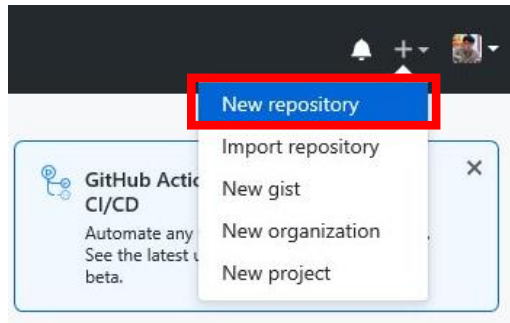
- 2) 지금까지한 commit 들을 확인하자 (6 개 아니면 다시 확인하자)

```
$ git shortlog
```

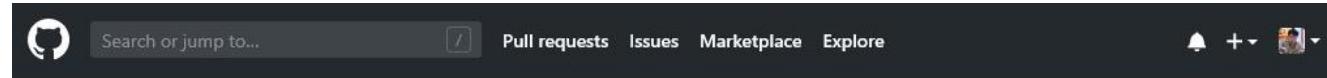
- 3) Github 원격저장소 URL 를 등록하자

( 잠깐 멈추고 <http://github.com> 를 켜고 repository 새로 생성하자 )

# Github 원격 저장소 만들기



새로운 원격저장소 생성  
(프로젝트명 자유)



## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner



Repository name \*

multiply



Great repository names are short and memorable. Need inspiration? How about [crispy-adventure](#)?

Description (optional)

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

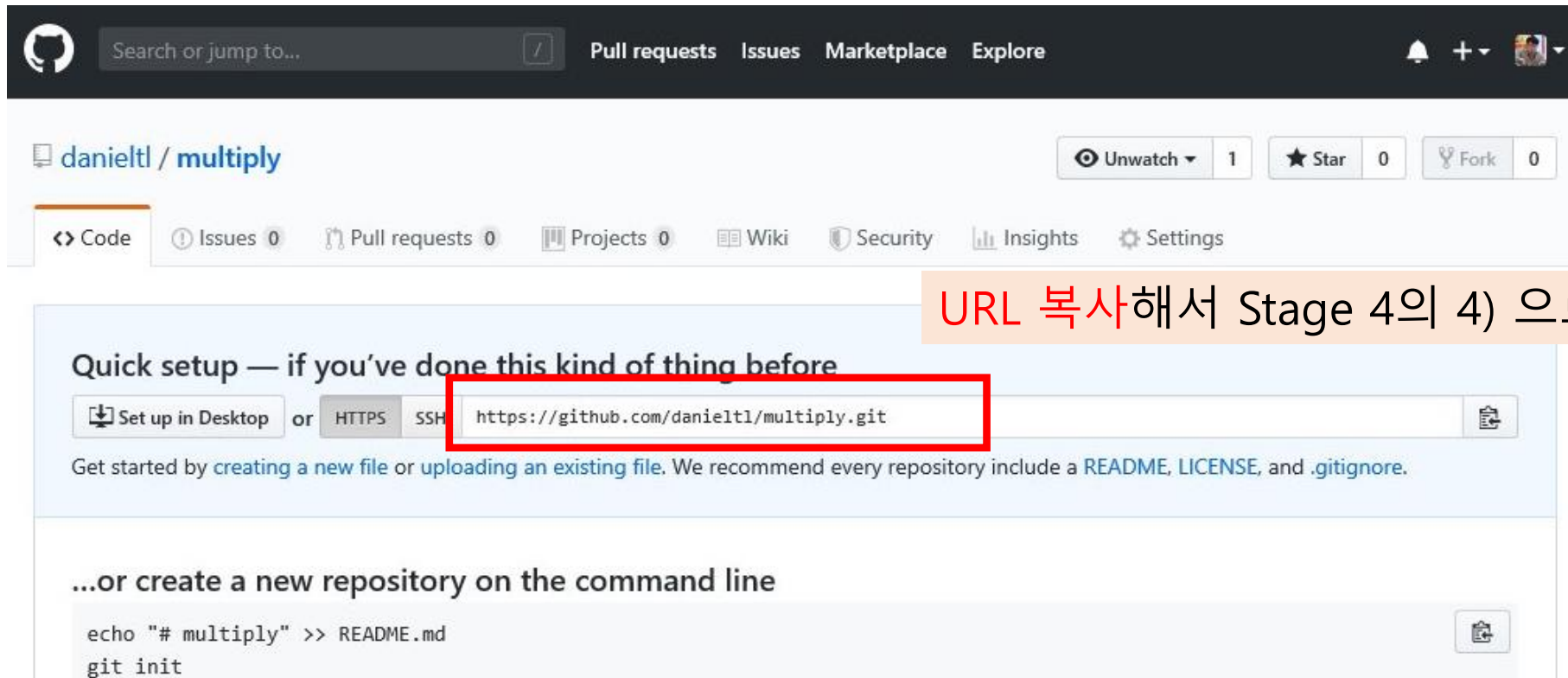
Add .gitignore: **None**

Add a license: **None**



Create repository

# Github 원격 저장소 만들기



URL 복사해서 Stage 4의 4) 으로 이어서 진행

# Git 실습 Stage 4 지금까지의 commit 을 push 하자

- 4) 아까복사한 URL 로 Github 원격저장소 등록하자 ('<' 와 '>' 기호는 제외하고 입력한다 )

```
$ git remote add origin < 아까복사한 URL>
```

- 5) Github 원격저장소 (origin) 에다 밀어 넣자

```
$ git push origin master
```

- 6) Github 웹사이트 열고 확인하자



# Git 실습 Stage 4 지금까지의 commit 을 push 하자

Search or jump to... Pull requests Issues Marketplace Explore

danieltl / multiply Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

No description, website, or topics provided.  
Manage topics

6 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

danieltl multiply: Sum up all the calculations ... Latest commit deeee2d in 9 hours

multiply.c	multiply: Sum up all the calculations	now
multiply.pdf	multiply: Add question PDF	now

Help people interested in this repository understand your project by adding a README. Add a README

# Git 실습 Stage 4 지금까지의 commit 을 push 하자

Search or jump to... Pull requests Issues Marketplace Explore

danieltl / multiply Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Branch: master

Commits on Aug 15, 2019

multiply: Sum up all the calculations danieltl committed in 8 hours	deeee2d	<>
multiply: Add calculations with each digit danieltl committed in 8 hours	9a5733c	<>
multiply: Split digits from second operand danieltl committed in 8 hours	797c393	<>
multiply: Add introduction message danieltl committed in 8 hours	0870f66	<>
multiply: Add base code danieltl committed in 7 hours	398b5e0	<>
multiply: Add question PDF danieltl committed in 7 hours	f283cc9	<>

지금까지 추가한 commit 들이 나오면 성공!

# Git 실습 Stage 5 방금 한 commit 수정하기

- 1) multiply.c 소스내의 Print 메시지를 바꾼다고 하자  
(commit6-1 폴더내 소스 활용)

```
$ git status
```

- 2) diff 를 통해서 변화분을 확인했다면 add 진행

```
$ git diff
```

- 3) 가장 위에 있는 commit 을 수정하자 (vi 에디터등 등록된 에디터 프로그램 켜지면 저장후 닫기)

```
$ git commit --amend
```

\* vi 에디터 또는 지정된 에디터 ( 메모장 등 ) 이 열릴수있다 .  
Commit 메시지 를 수정하거나 수정없이 에디터를 닫으면 완료

\* vi 에디터는 i 또는 a 키를 눌러 수정모드로 변경하여 수정 후  
ESC 키 누르고 :wq 명령어를 입력하여 Enter 눌러 나올 수 있다 .

# Git 실습 Stage 5 방금 한 commit 수정하기

5) 바로 push 해보자 ( 충돌 오류발생 )

```
$ git push origin master
```

6) 강제로 push 해서 수정하자 (--force 또는 -f 옵션 사용가능 )

```
$ git push origin master --force
```

7) 다시 Github 가서 제대로 변경되었는지 확인해보자

\* 4) 의 충돌이유는 Local( 본인 노트북 ,PC) 에 기록된 commit 들과 Github 에 먼저 push 하여 저장된 commit 들의 commit ID 가 일치하지 않는부분이 있기 때문

\* 물론 모든 commit ID 가 일치한상태에서 Local 에만 새로운 추 가 commit 있을때는 push 가능

# Git 실습 Stage 6 add 한거 취소하기

- 1) touch 로 빈파일 생성하고 add 하자 (; 로 명령어들을 연속적 실행가능 )

```
$ touch test; git add test
```

- 2) 현재상태 확인해보고

```
$ git status
```

- 3) reset 으로 add 한거 취소해보자

```
$ git reset
```

- 4) 현재 상태 다시한 번 확인해본다

```
$ git status
```

# Git 실습 Stage 7 commit 한거 없애기

- 1) 아까 test 파일 여전히 존재하는지 확인 ( 지웠으면 다시만들기 )

```
$ git status
```

- 2) 임의로 실수의 commit 을 만들어 낸다 ( ; 로 명령어들을 연속적 실행가능 )

```
$ git add test; git commit -sm "test"
```

- 3) 그리고 push 까지해서 Github 에 있는 tree 까지 실수 commit 을 넣어버린다

```
$ git push origin master
```

- 4) 그리고 가장 최근 commit 을 지운다

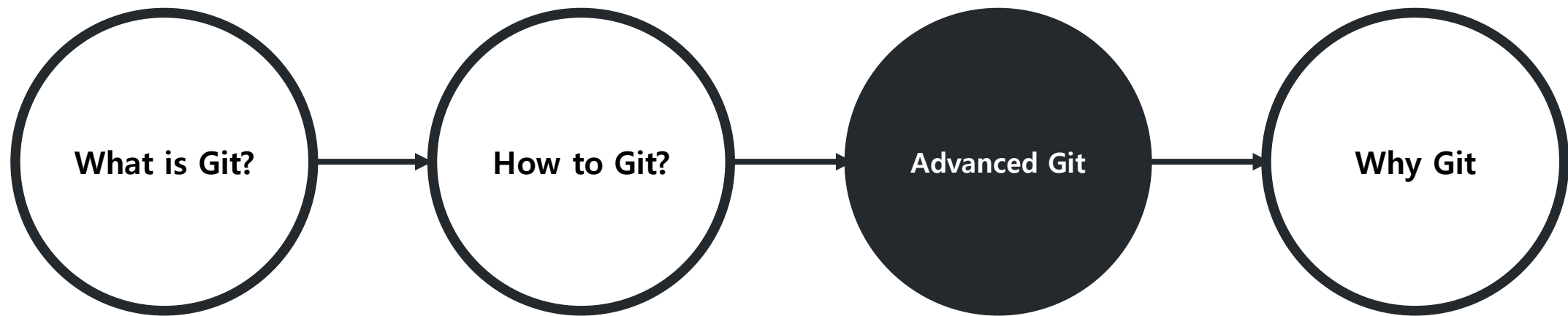
```
$ git reset HEAD~1
```

- 5) 강제로 Github 에 있는 tree 도 밀어넣어서 수정한다

```
$ git push origin master --force
```

\* Github 에 있는 commit 을 수정할 길은 Local 에서 수정후 -f 옵션으로 push 하는방법뿐이다 .

# Contents



# | Git 자유실습

다시 Stage 1 부터 새롭게 반복연습 하거나

조금은 어려울 수 있으나 고급과정을 계속해서 진행 하거나



# Git 실습 Github의 기능 pull-request 실습

Pull-request 한다는 의미

다른 프로젝트에 내가 만든 commit을 제출한다는 의미 (실제 전송단위는 branch)

상대방 프로젝트를 fork(복사) 해서

내 계정에서 관리되는 프로젝트로 새롭게 만들어 두고

그 fork한 github 프로젝트를 토대로 새로운 commit 내용들을 pull-reqeust 제출 할 수 있다

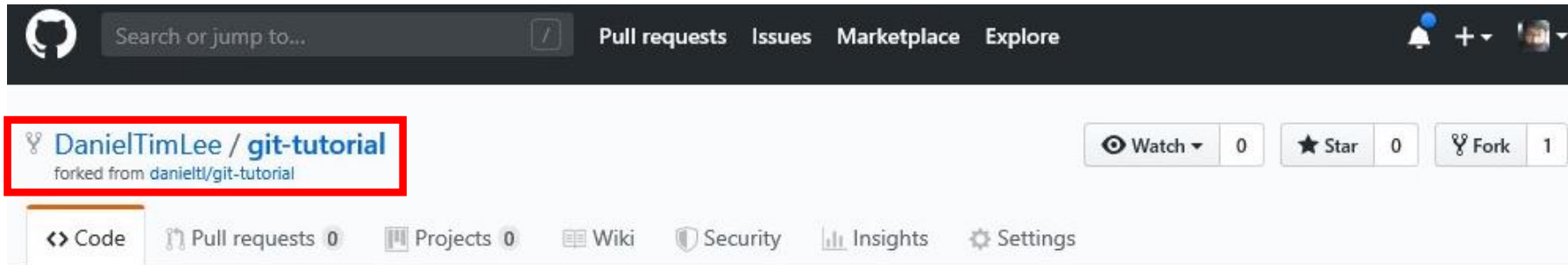
# Git 실습 Github에서 fork하기

\* 주의 fork 는 본인 프로젝트를 대상으로 하는 게 아니다 .

The screenshot shows the GitHub interface for the repository 'danieltl / git-tutorial'. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this, the repository name 'danieltl / git-tutorial' is displayed. To the right of the name are buttons for 'Watch' (1), 'Star' (0), and 'Fork' (0). The 'Fork' button is highlighted with a red rectangle. Below the repository name, there are tabs for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Security', and 'Insights'. A section titled 'Just Do Git!' contains tags for 'git', 'github', and 'tutorial'. Below this, a summary bar shows '11 commits', '1 branch', '0 releases', '1 contributor', and 'MIT' license. At the bottom, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and 'Clone or download'. A commit history section shows a commit by 'danieltl' titled 'Readme: Added demo for each stage' with the latest commit 'd3d31e1' made '4 hours' ago. Below the commit history, a file entry 'pull\_request\_test/danieltl' is shown with the description 'multiply: add algorithm problem pdf' and a timestamp of '6 hours ago'.

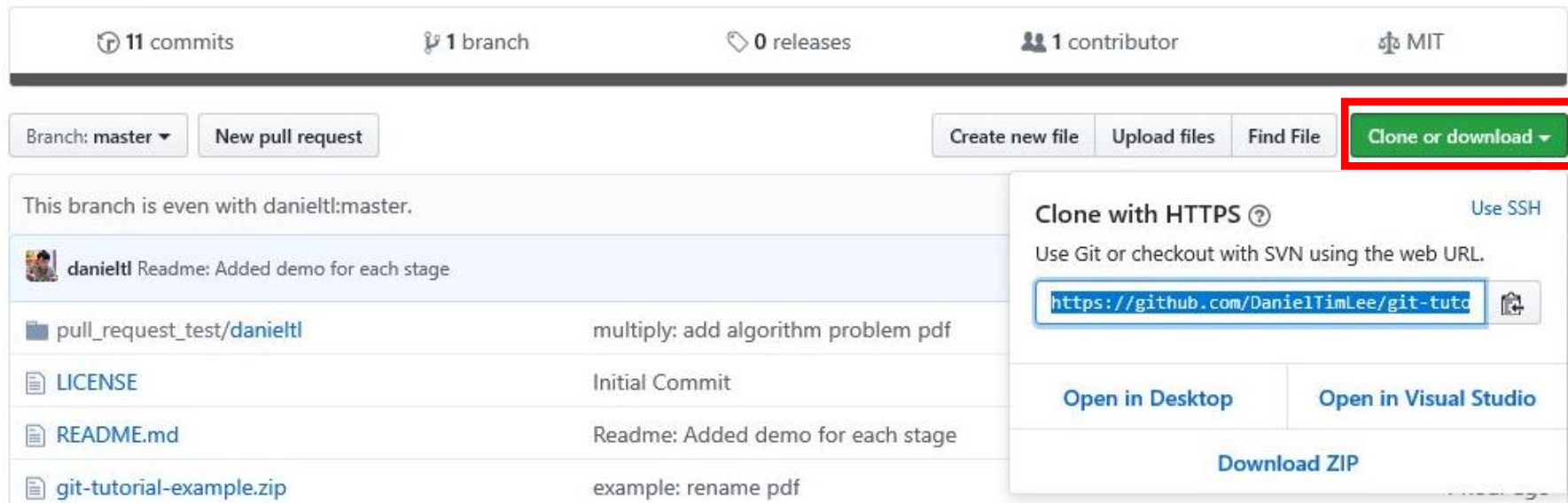
github.com/danieltl/git-tutorial 에서 fork 버튼 클릭!

# Git 실습 Github에서 fork하기



fork가 되면 내 원격 저장소가 추가 된다  
\* forked from danieltl/git-tutorial 확인

Clone or download 클릭  
fork해서 만들어진 본인 repo URL 복사



# Git 실습 Stage 8 Pull-Request 보내기

- 1) (git-bash/ 터미널에서) 최초경로 HOME 경로로 이동하자 (multiply 작업하던 폴더에서 벗어나기)

```
$ cd ~
```

- 2) clone 으로 fork한 repo 받아오기 ('<' 와 '>' 두 기호는 생략하고 적는다)

```
$ git clone < 아까 fork 한 repo 에서 복사한 URL >
```

- 3) clone 한 프로젝트 폴더로 이동하기 ( 만약 프로젝트명이 git-tutorial-1 이면 그 이름으로 이동 )

```
$ cd git-tutorial
```

- 4) pull-request 작업할 브랜치 (develop) 따로 만들기

```
$ git checkout -b develop
```

- 5) pull\_request\_test 폴더로 이동해서

```
$ cd pull_reqeust_test
```

\* 브랜치 생성이란  
간단한 비유로 설명하면

“ 같은 폴더에 또다른 세상열기 ”

# Git 실습 Stage 8 Pull-Request 보내기

6) 내 이름으로된 폴더 만들고

```
$ mkdir daniel; cd daniel
```

7) 내 이름으로된 폴더에 작업하던 multiply.c 소스파일 또는 아무파일 복사해 넣기

8) 추가한 폴더 가 ( 내가 작업한 소스 내용 ) 통째로 add ('<' 와 '>' 은 생략한다)

```
$ git add < 나의 소스작업폴더 >
```

9) 준비된 파일들 commit

```
$ git commit -sm "test pull request"
```

10) 내가 fork 한 repo 의 develop 브랜치로 push ( 주의 : master 아님 )

```
$ git push origin develop
```

# Git 실습 Stage 8 Pull-Request 보내기

\* 나의 프로필에서 fork 해서 만들어진 프로젝트 페이지로 이동

Search or jump to... Pull requests Issues Marketplace Explore

DanielTimLee / git-tutorial  
forked from danieltl/git-tutorial

Watch 0 Star 0 Fork 1

Code Pull requests 0 Projects 0 Wiki Security Insights Settings

Just Do Git!  
Manage topics

11 commits 2 branches 0 releases 1 contributor MIT

Branch: master New pull request Create new file Upload files Find File Clone or download

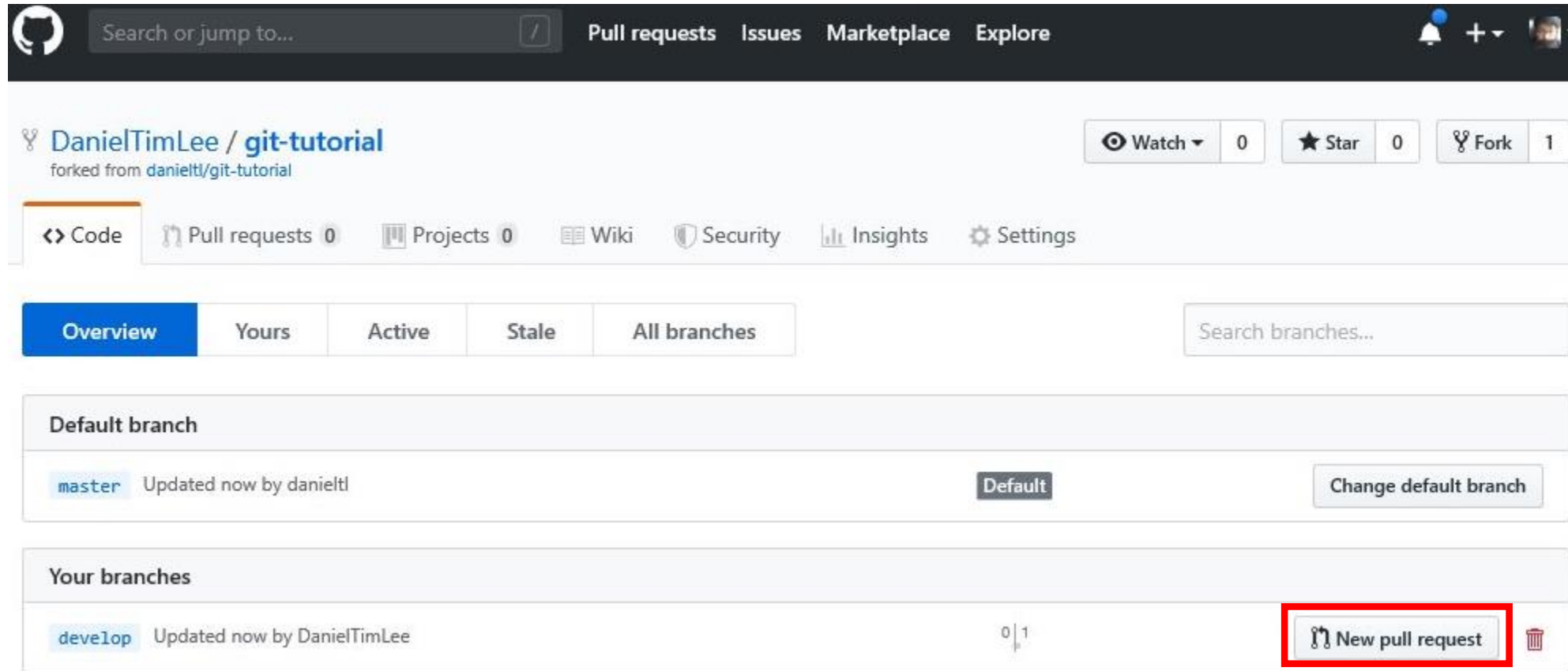
This branch is even with danieltl:master. Pull request Compare

danieltl Readme: Added demo for each stage Latest commit d3d31e1 in 4 hours

pull_request_test/danieltl	multiply: add algorithm problem pdf	7 hours ago
LICENSE	Initial Commit	yesterday

방법 1-1) 방금 push한 브랜치를 확인하기 위해 클릭!

# Git 실습 Stage 8 Pull-Request 보내기



방법 1-2) pull-request 하려는 브랜치에서 New pull-request 버튼 클릭

# Git 실습 Stage 8 Pull-Request 보내기

The screenshot shows the GitHub interface for a repository named 'git-tutorial' by 'DanielTimLee', which is a fork of 'danieltl/git-tutorial'. The repository has 0 stars, 0 forks, and 1 watch. The 'Pull requests' tab is selected, showing 0 pull requests. Below this, a bar indicates 11 commits, 2 branches, 0 releases, 1 contributor, and the MIT license. The 'Your recently pushed branches' section shows the 'develop' branch, pushed 3 minutes ago. A green button labeled 'Compare & pull request' is highlighted with a red box. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and 'Clone or download'. The bottom section shows a commit by 'danieltl' titled 'Readme: Added demo for each stage' with the latest commit hash 'd3d31e1' from 4 hours ago. Below that, a file named 'pull\_request\_test/danieltl' is shown, with the description 'multiply: add algorithm problem pdf' and a timestamp of '7 hours ago'.

DanielTimLee / **git-tutorial**  
forked from danieltl/git-tutorial

Watch 0 Star 0 Fork 1

Code Pull requests 0 Projects 0 Wiki Security Insights Settings

Just Do Git!  
Manage topics

11 commits 2 branches 0 releases 1 contributor MIT

Your recently pushed branches:

develop (3 minutes ago) **Compare & pull request**

Branch: master New pull request Create new file Upload files Find File Clone or download

This branch is even with danieltl:master. Pull request Compare

danieltl Readme: Added demo for each stage Latest commit d3d31e1 in 4 hours

pull\_request\_test/danieltl multiply: add algorithm problem pdf 7 hours ago

방법 2) fork한 저장소에서 **pull-request** 클릭



# Git 실습 Stage 8 Pull-Request 보내기

danieltl / git-tutorial

Watch 1 Star 0 Fork 1

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Security

Insights

## 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 repository: danieltl/git-tutorial

base: master

head repository: DanielTimLee/git-tutorial

compare: develop

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



test pull request

Write

Preview

AA B i “ <> ↺ ⋮ ≡ ↻ @ 🚩 ↶

Signed-off-by: Daniel T. Lee <danieltimlee@gmail.com>

Attach files by dragging & dropping, selecting them.

☒ Allow edits from maintainers. [Learn more](#)

Create pull request

pull-request 보내기

# Git 실습 Stage 8 Pull-Request 보내기

The screenshot shows a GitHub Pull Request interface. At the top, the repository name 'danieltl / git-tutorial' is highlighted with a red box. Below it, the 'Pull requests' tab is also highlighted with a red box. The pull request title is 'test pull request #1'. The description states 'DanielTimLee wants to merge 1 commit into danieltl:master from DanielTimLee:develop'. The status bar shows 'Conversation 0', 'Commits 1', 'Checks 0', and 'Files changed 1' with a green bar indicating '+27 -0' changes. A comment from DanielTimLee is visible, signed off by Daniel T. Lee. A green box with a checkmark indicates 'This branch has no conflicts with the base branch'. The right sidebar shows sections for Reviewers, Assignees, Labels, Projects, Milestone, and Notifications.

danieltl / git-tutorial

Watch 1 Star 0 Fork 1

Code Issues 0 Pull requests 1 Projects 0

test pull request #1

Open DanielTimLee wants to merge 1 commit into danieltl:master from DanielTimLee:develop

Conversation 0 Commits 1 Checks 0 Files changed 1 +27 -0

DanielTimLee commented in 9 hours

Signed-off-by: Daniel T. Lee danieltimlee@gmail.com

test pull request 1a19022

Add more commits by pushing to the **develop** branch on DanielTimLee/git-tutorial.

This branch has no conflicts with the base branch  
Only those with [write access](#) to this repository can merge pull requests.

Write Preview AA B i “ <> @

Leave a comment

Reviewers  
No reviews

Assignees  
No one assigned

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Notifications Customize

원본 프로젝트 에서 만들어진 pull-request 확인하기  
\* 내 프로필에 fork된 프로젝트가 아님!

# Git 실습 Stage 9 merge로 2개 브랜치 합치기

- 1) 방금작업한 develop 브랜치 현재 브랜치인지 확인하자 (status로도 확인가능 )

```
$ git branch
```

- 2) 추가 브랜치 만들어보자

```
$ git checkout -b test
```

- 3) touch 로 빈파일하나 만들어서 commit 만들어보자 (; 로 명령어들을 연속적 실행가능 )

```
$ touch test; git add test; git commit -sm "test"
```

- 4) 현재브랜치 (develop) 을 기준으로 추가브랜치 (test) 을 합치자

```
$ git checkout develop; git status; git merge test
```

# Git 실습 Stage 10 rebase 하기

Rebase 사용하는 시나리오 !!

commit 을 역사의 한단위 '블럭' 이라 하고 블럭들의 모임을 'tree' 라 할때

내가 쌓은 블럭을 잠시 빼고

( 뺀 나머지 ) 기준이 되는 tree 를 최신 업데이트 한 후 에

그 위에 다시 내 블럭을 쌓아 올릴때 쓸수있다 .

# Git 실습 Stage 10 rebase 하기

- 1) Stage 8. 에서 fork 후 clone 했던 프로젝트경로로 이동하여 upstream 을 추가하자  
\* 주의 ) 본인프로젝트 URL 이 아니다

```
$ git remote add upstream https://github.com/daniel-tl/git-tutorial.git
```

- 2) upstream 의 dev 브랜치를 가져오자

```
$ git fetch upstream dev
```

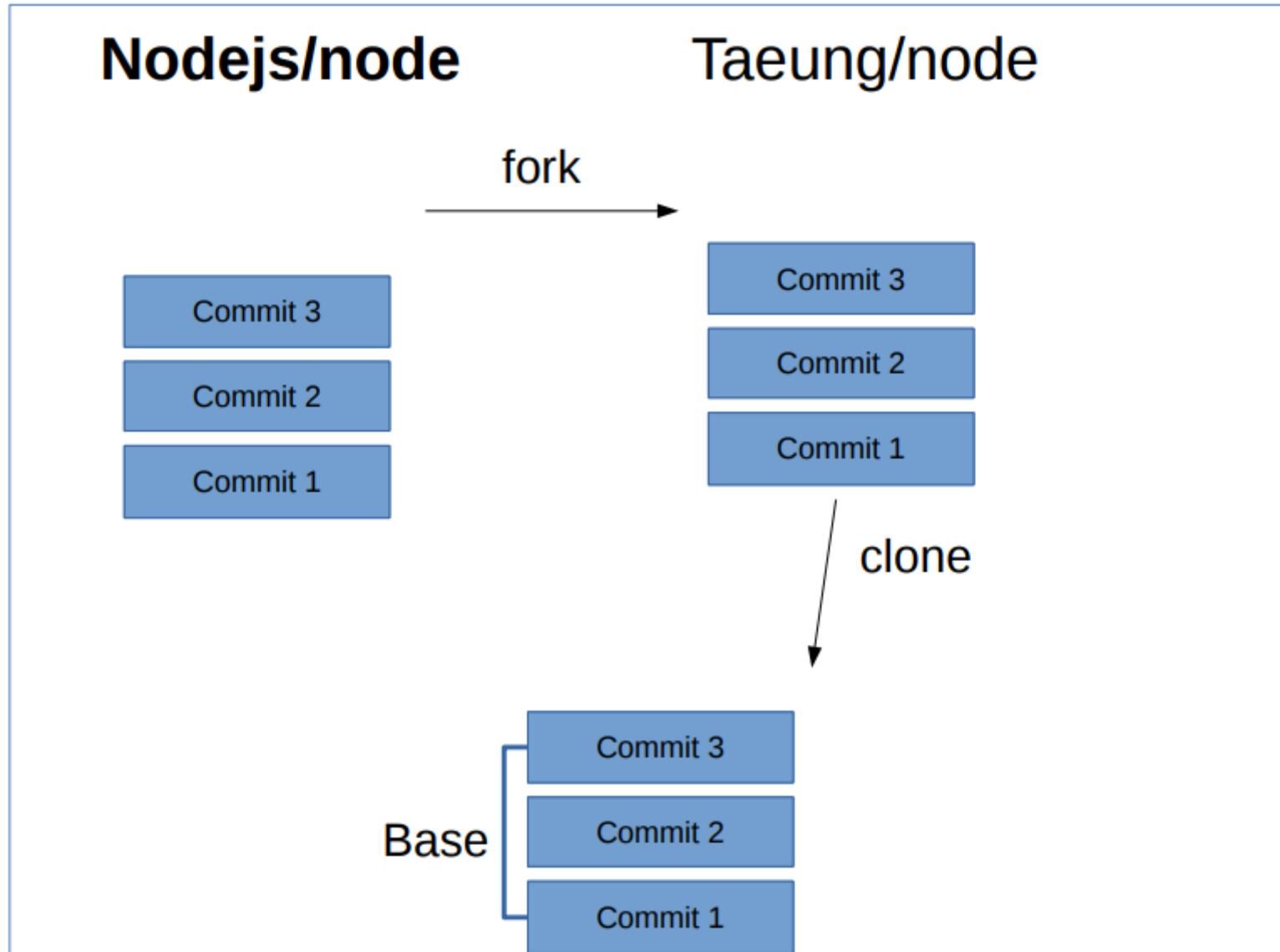
- 3) 현재 내 브랜치 가 develop 인지 확인하자

```
$ git status
```

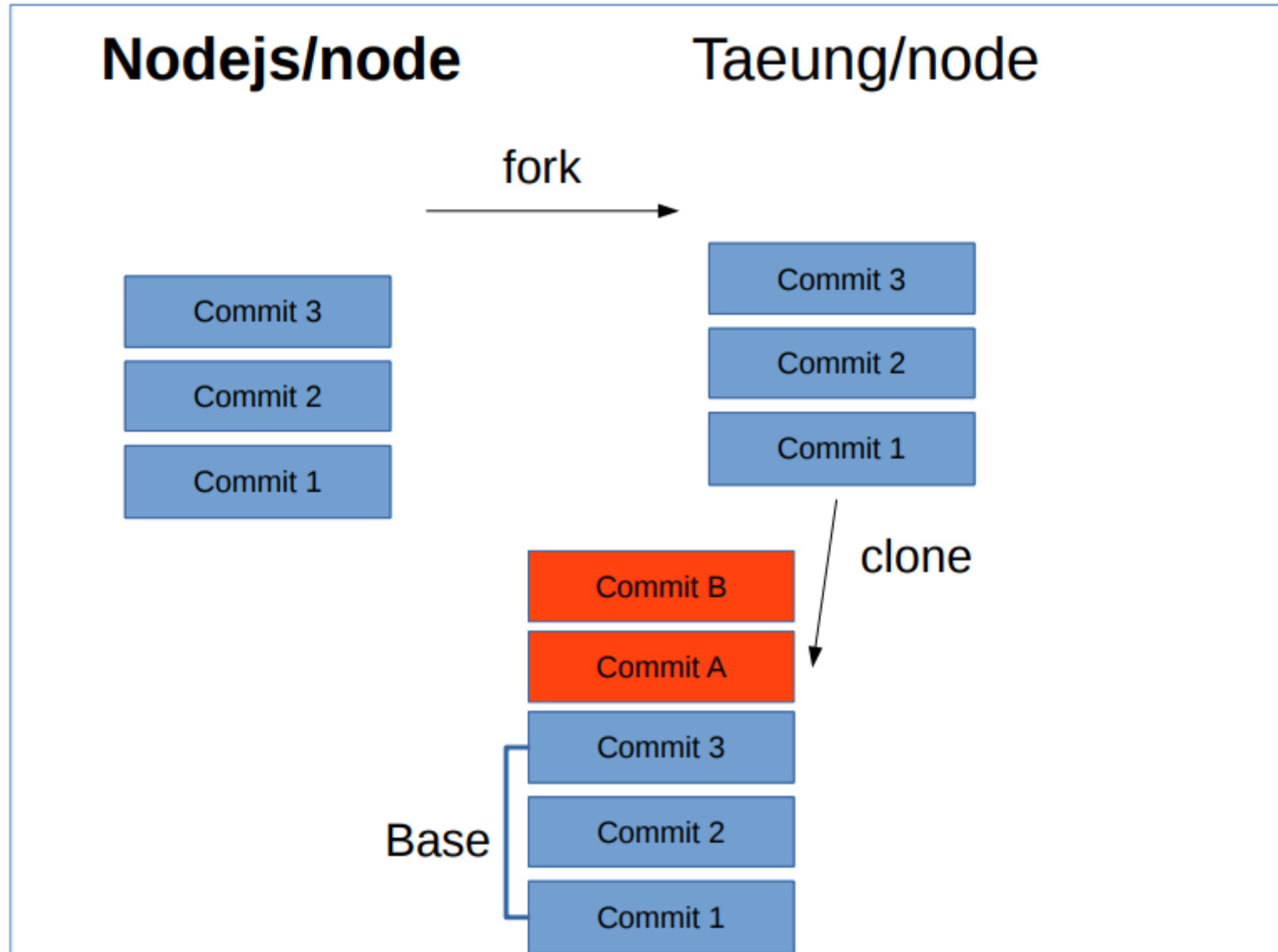
- 4) rebase 하자

```
$ git rebase upstream/dev
```

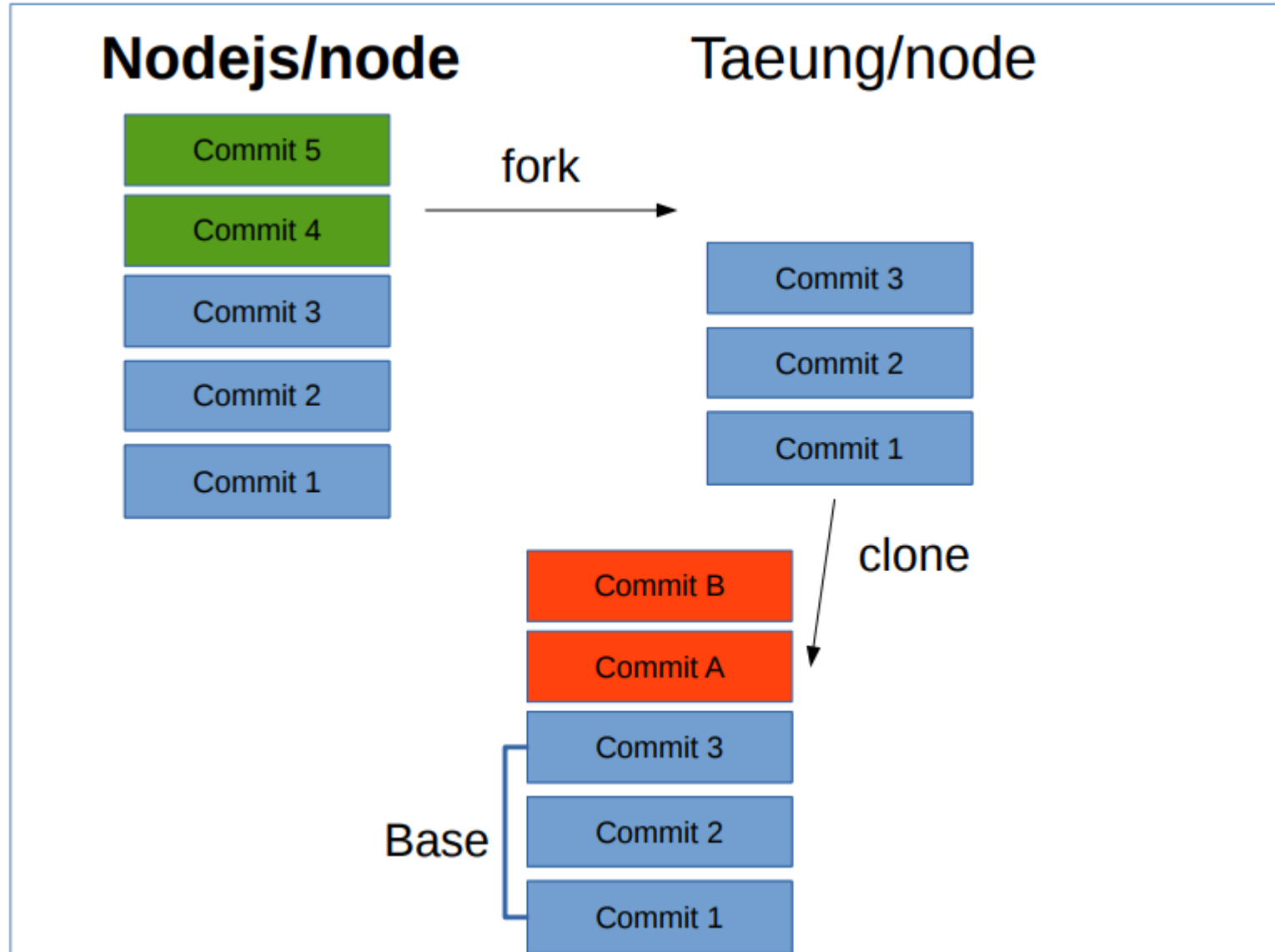
# Git 실습 Stage 10 rebase 하기



# Git 실습 Stage 10 rebase 하기

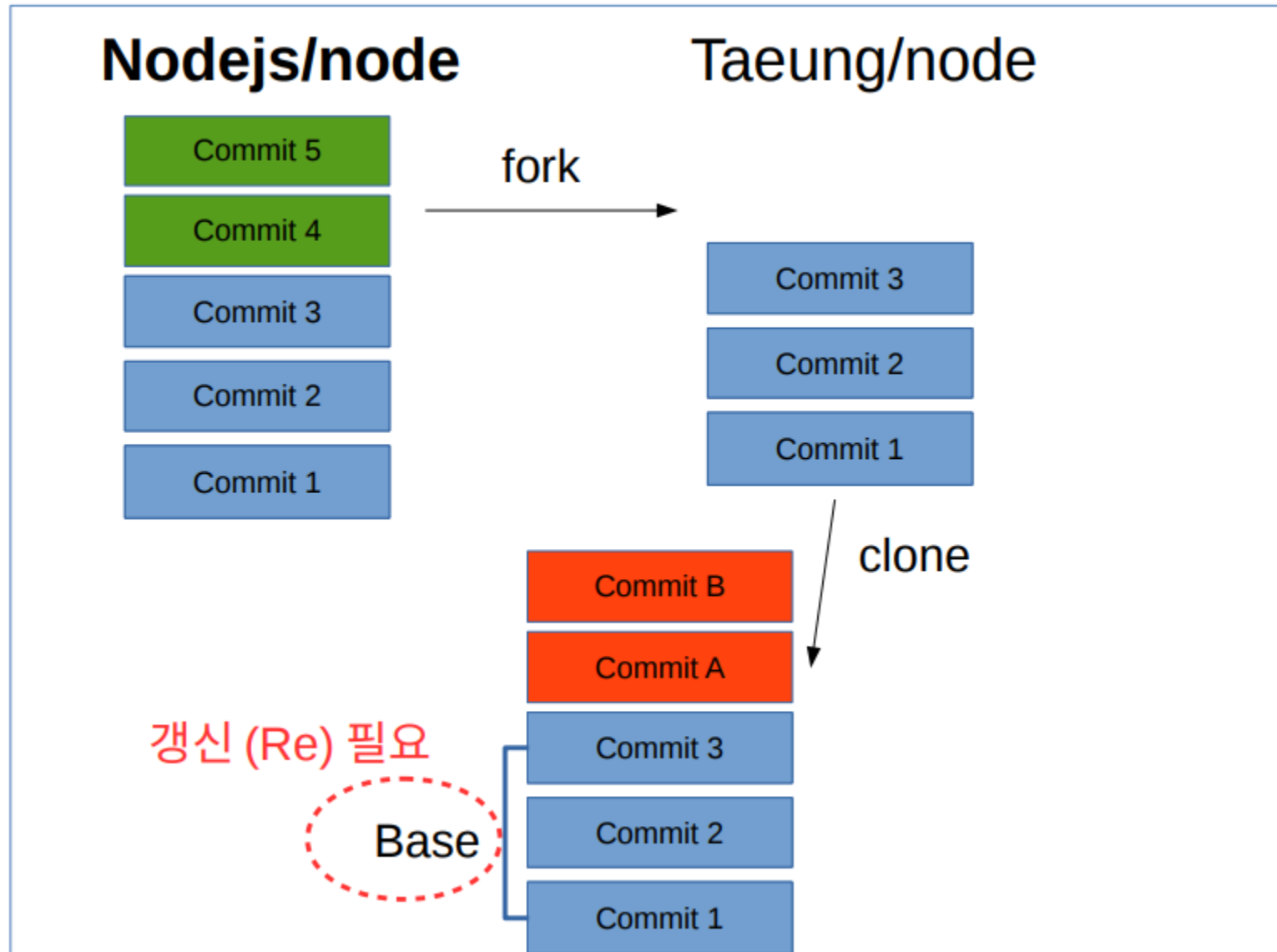


# Git 실습 Stage 10 rebase 하기

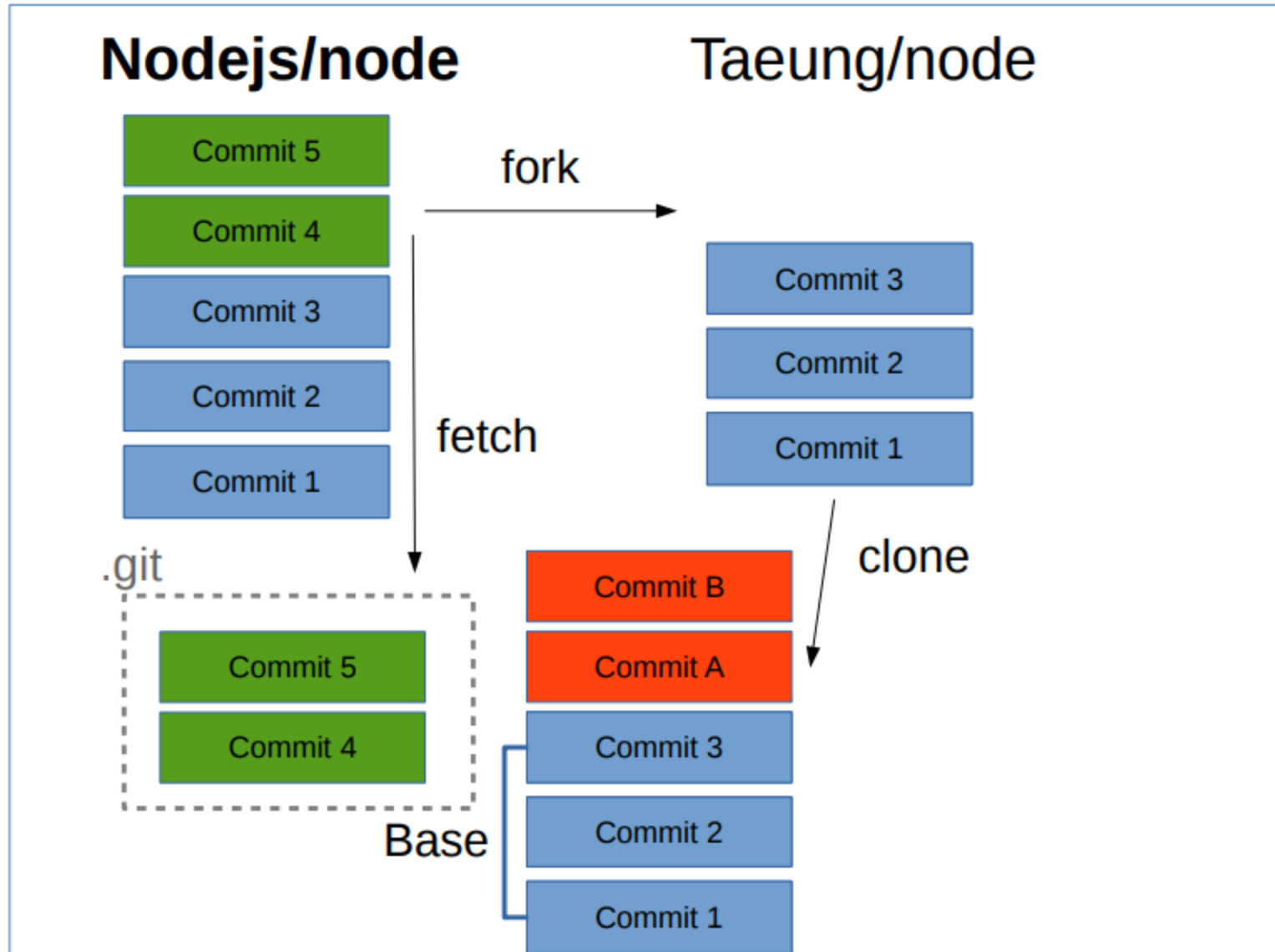




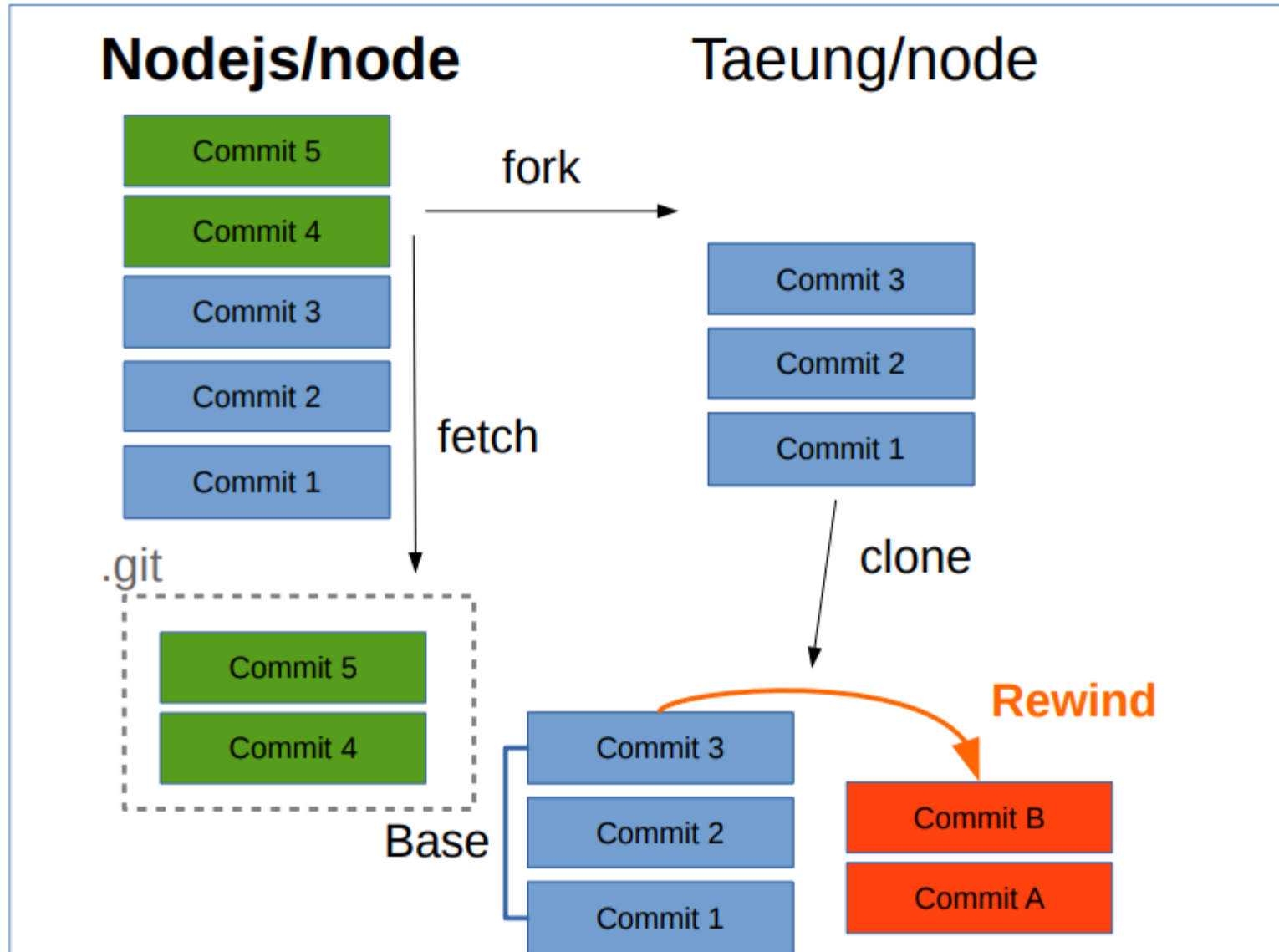
# Git 실습 Stage 10 rebase 하기



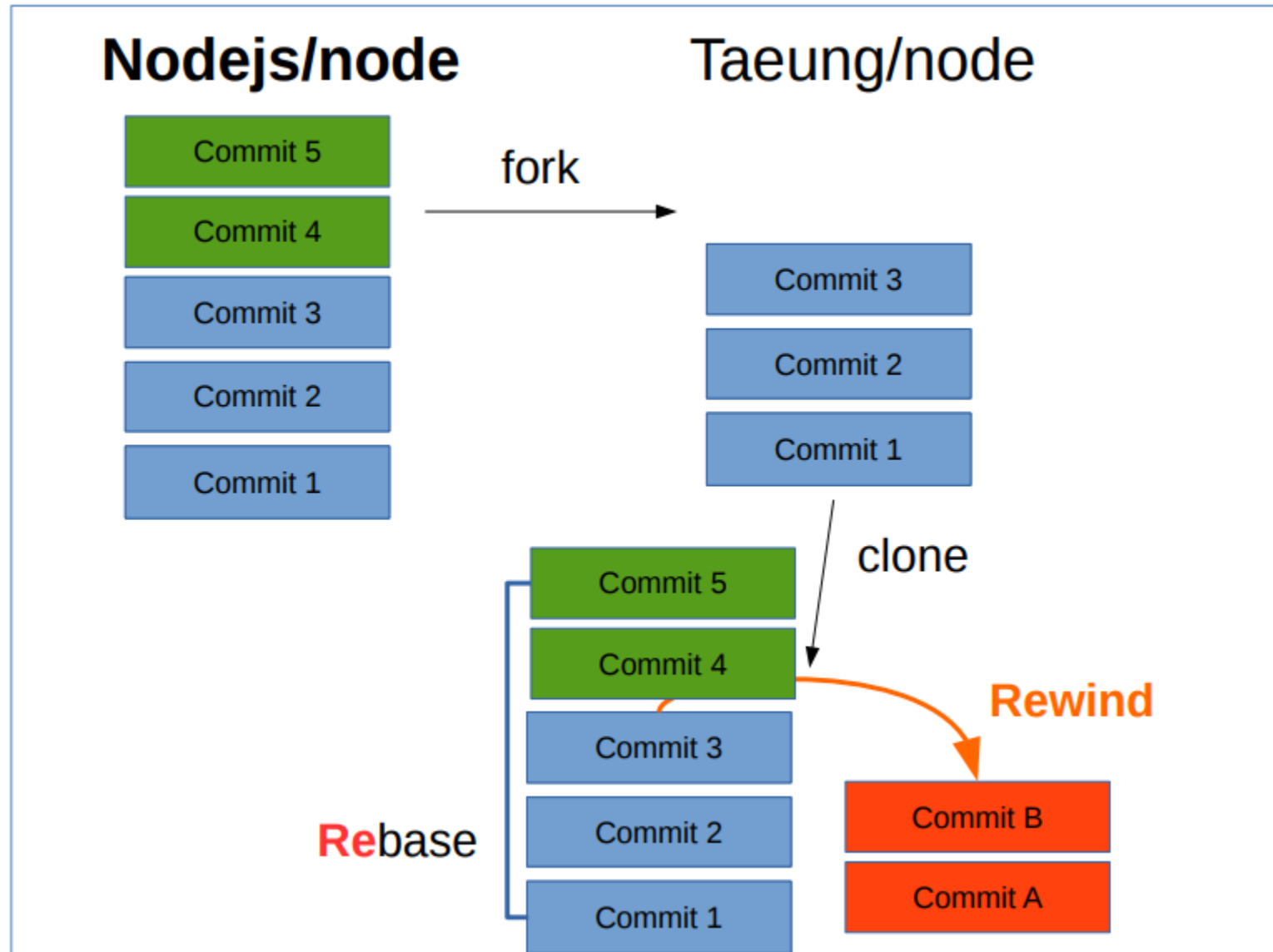
# Git 실습 Stage 10 rebase 하기



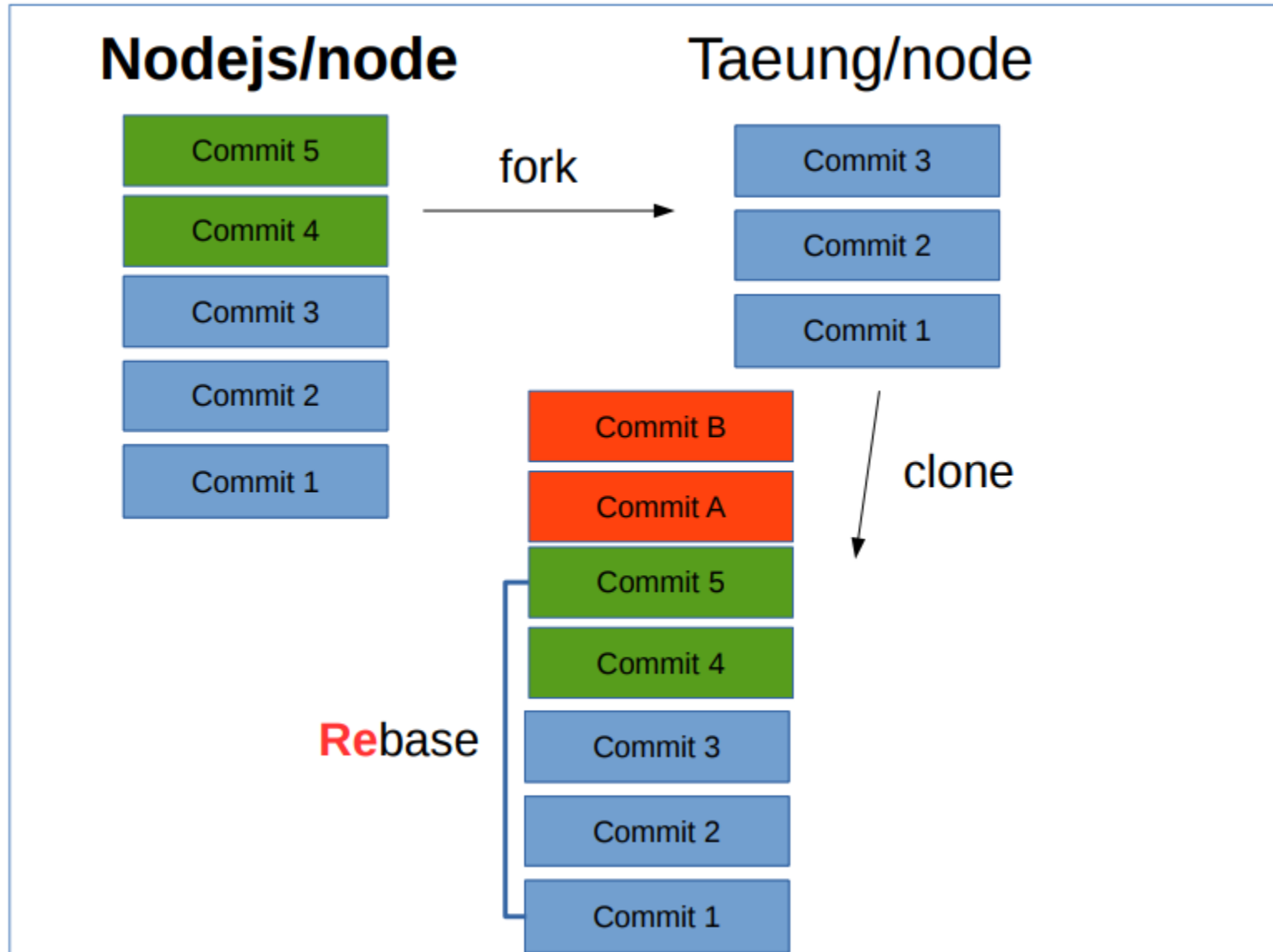
# Git 실습 Stage 10 rebase 하기



# Git 실습 Stage 10 rebase 하기



# Git 실습 Stage 10 rebase 하기



# Git 실습 Stage 11 중간에 낀 commit 수정하기

- 1) commit 최초기록 부터 2 번째 commit 을 수정해보자

```
$ git rebase -i --root
```

- 2) vi 에디터 열리면 수정하려는 commit(2 번째가) 맨앞에  
"pick" 을 지우고 대신에 "edit" 을 적고 저장하고 에디터를 끄자

\* --root 를 사용한 이유는 최초의 commit 까지 수정할수있도록 하기위함

- 3) 상태확인해서 rebase 진행 정상적인지 보고

```
$ git status
```

- 4) commit 정보 수정 ("baekjoon" commit 메시지에 추가) 하고 --continue 로 마무리

```
$ git commit --amend -sm "multiply: add baekjoon algorithm problem pdf"
```

```
$ git rebase --continue
```

# Git 실습 Bonus Stage blame 으로 추적하기

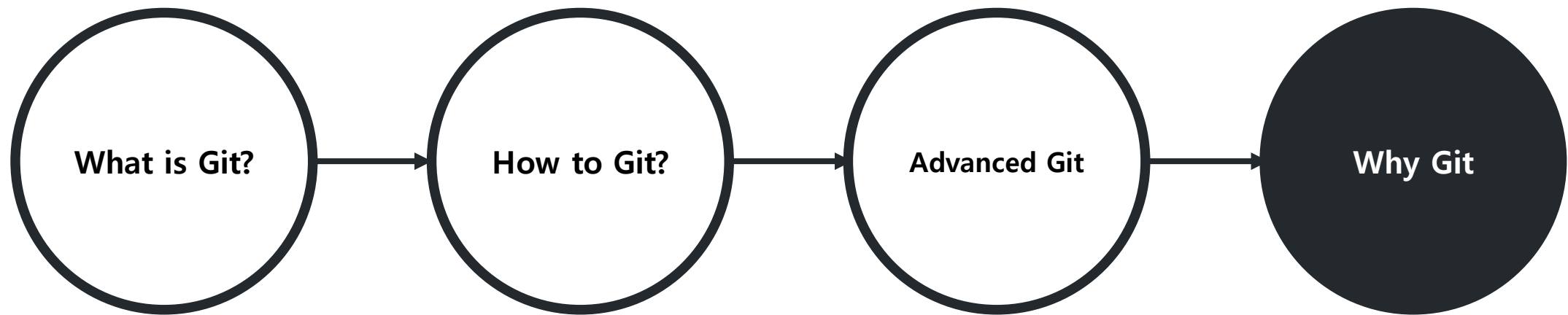
- 1) 어떤 파일이든 누가 어느라인을 수정했는지 파악해보자

```
$ git blame multiply.c
```

- 2) 해당 commit ID를 이용하여 그 당시 commit 정보확인

```
$ git show <Commit ID>
```

# Contents





# Git 을 쓰는 진짜이유

협업 때문에 Git 을 쓴다. ( 집단지성의 극대화 )

현대적인, 선진화된 소스코드 개발과정의 필수도구로 Git 을 쓴다 (Needs)



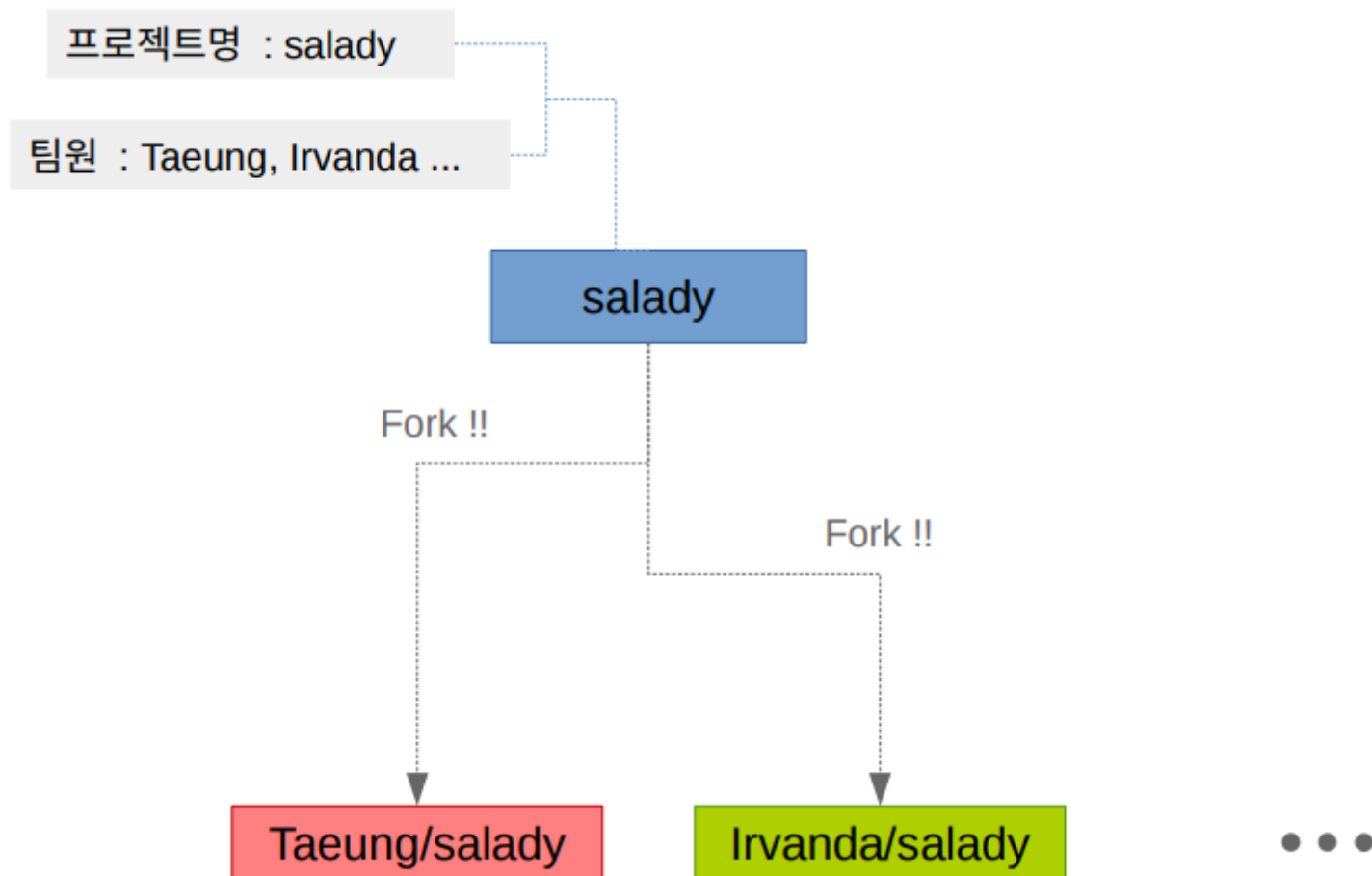
# Commit 단위개발 Idea

Commit 단위로 코딩하고 리뷰하고 토론하고 적용한다 . ( 집단지성의 극대화 )

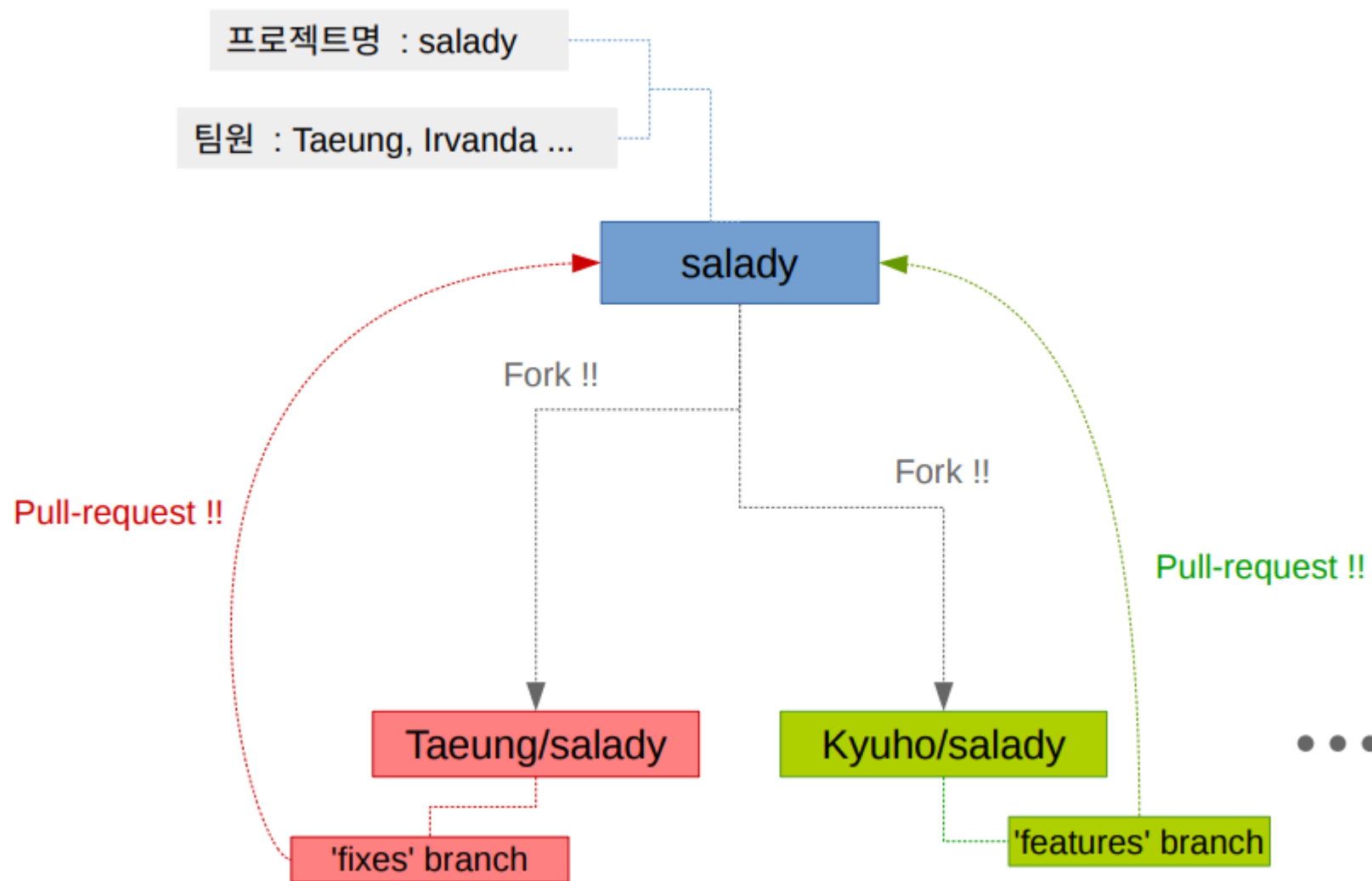
혼자가면 빨리가지만 함께가면 멀리간다



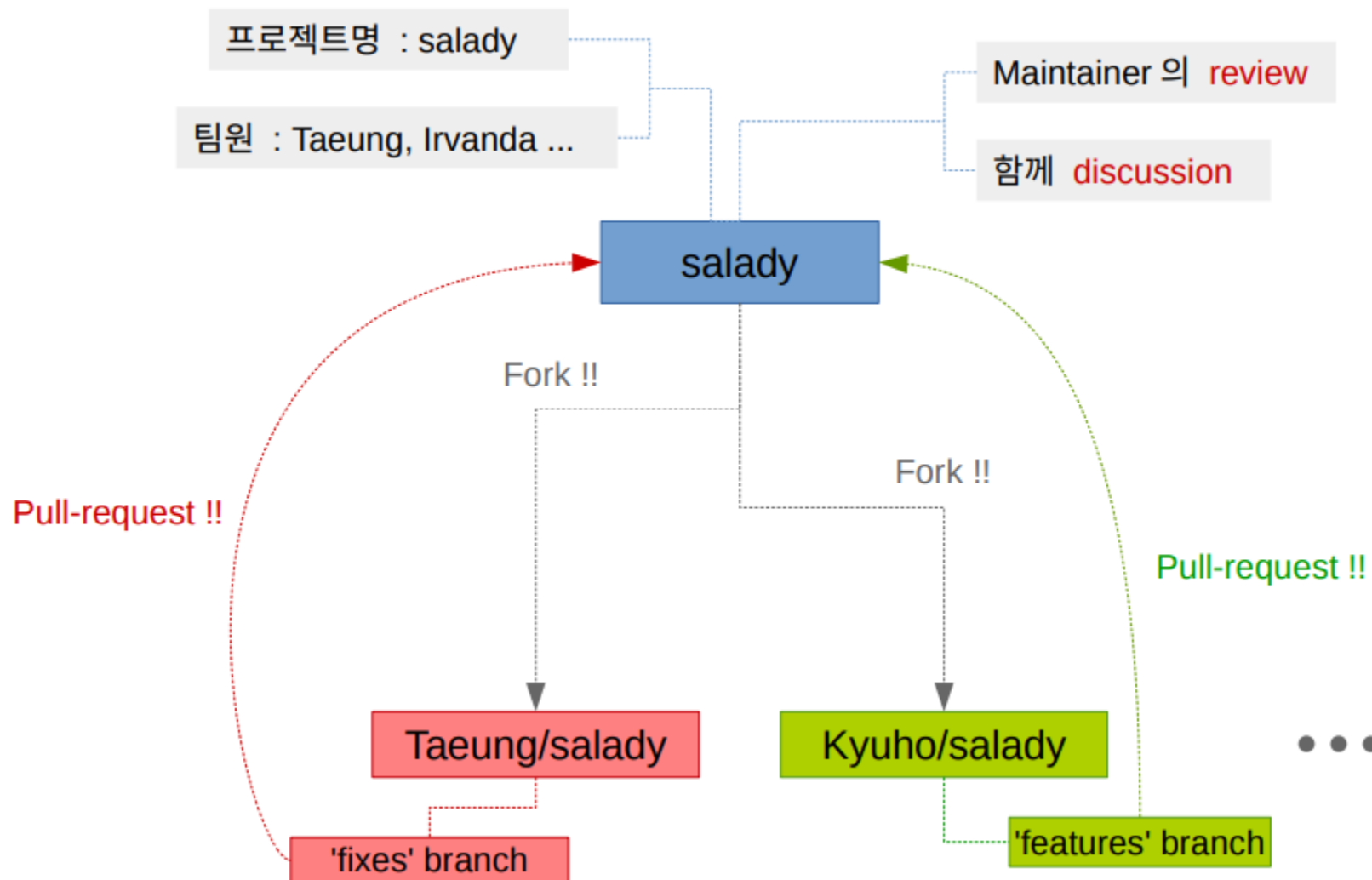
# 프로젝트 관리 방식 Git 운용 전략



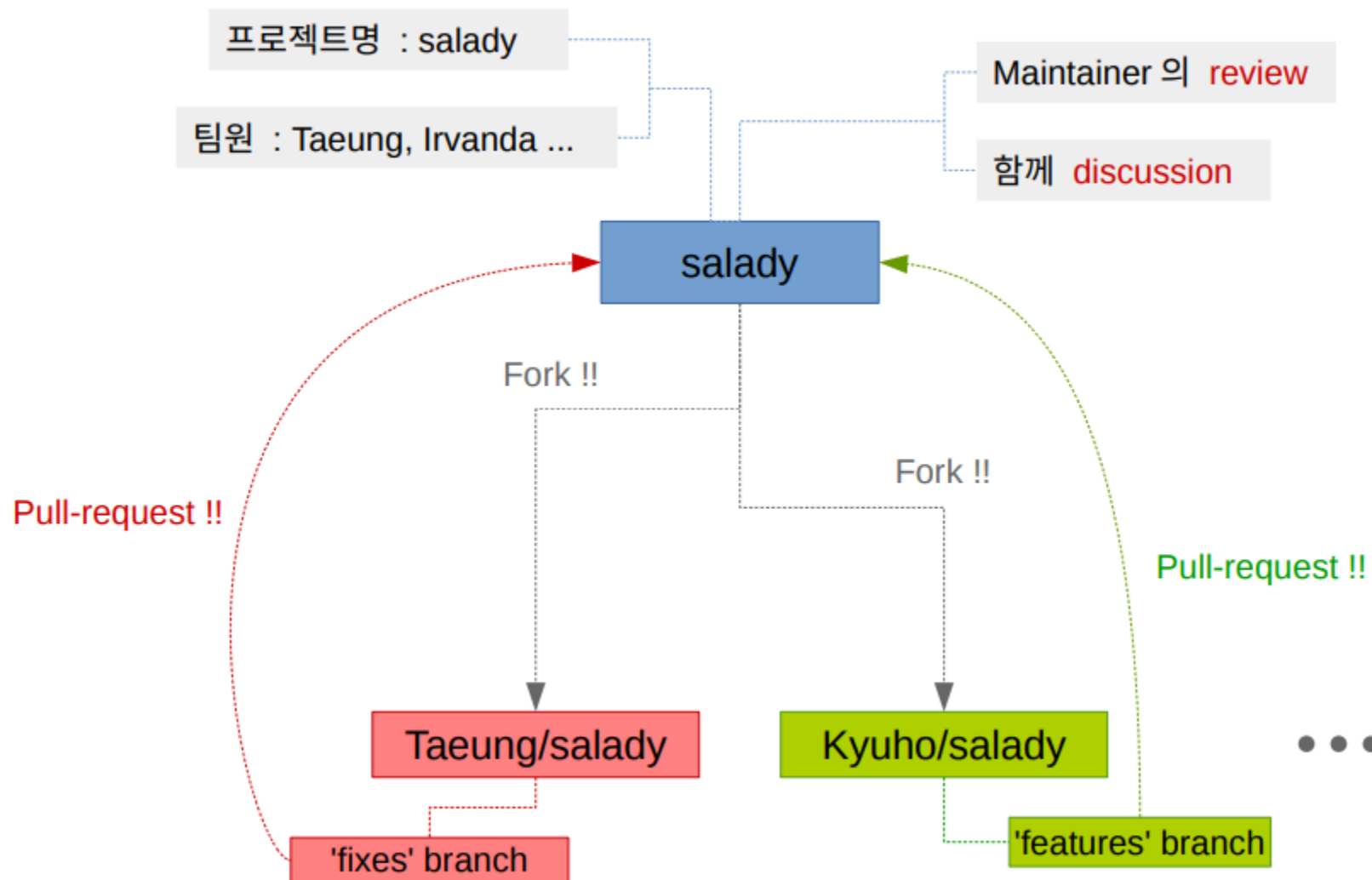
# 프로젝트 관리 방식 Git 운용 전략



# 프로젝트 관리 방식 Git 운용 전략



# 프로젝트 관리 방식 Git 운용 전략



# Q&A - 자주 묻는 질문들과 간단한 대답

## 1) Git 과 Github 의 차이는 ?

Git 은 각 컴퓨터 (**local**) 에 설치되어 소스코드관리가 가능한 프로그램이고  
Github 는 **remote** 저장소가 있는 외부서버를 지칭한다 .

## 2) Commit 과 Push 의 차이는 ?

commit 은 **local** 작업폴더에 history 를 쌓는것이어서 외부망 (internet) 을 안쓰고  
Push 는 **remote** 저장소 (Github 등 ) 에 history 를 쌓는것이어서 외부망 (internet) 이 필요하다 .

## 3) Fetch 와 Pull 의 차이는 ?

Remote 저장소 (Github 등 ) 로 부터 최신 commit 정보들을 가져오는것은 매한가지이나  
Fetch 는 가져와서 **임시폴더** (.git) 에 **저장**하고  
Pull 은 바로 현재 branch 에 **merge** 작업을 **동반**한다 .

## 4) Rebase 와 Merge 의 차이는 ?

둘다 두 branch 의 차이점 (commits) 를 합치는것은 매한가지나  
Rebase 는 합치기 전에 **되감기** (rewinding) 를 하고  
Merge 는 **안하고** 합친다 .

# Thank You



71cf8aa