

git 설치와 기본 설정



About..

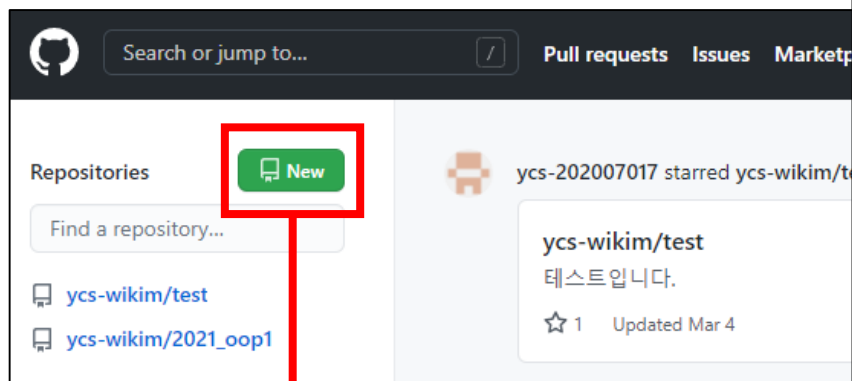
컴퓨터소프트웨어공학과
김 원 일



나의 저장소 관리하기 - 1




• 나의 메인 페이지



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 *

 ycs-wikim ▼ /

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

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

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

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)




나의 저장소 관리하기 - 2



- 저장소 이름

- 저장소는 항상 본인 계정 명 뒤에 입력한 이름으로 생성


Owner * Repository name *


 ycs-wikim / 2021_oop2 ✓

Great repository names are ... 2021_oop2 is available. ... Need inspiration? How about [bookish-bassoon](#)?

Description (optional)

- 저장소의 권한 설정

☒  **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.



나의 저장소 관리하기 - 3



• 저장소 초기 설정

- Add a README file
 - 저장소 설명 파일 추가 여부 설정
- Add .gitignore
 - 저장소가 관리하지 않을 파일 정보 설정
- Choose a license
 - 저장소의 라이선스 설정

Initialize this repository with:

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

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

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

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




Create repository



나의 저장소 관리하기 - 4






• 저장소 정보 확인


 Search or jump to... / [Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)  + 




[ycs-wikim / 2021_oop2](#) Unwatch 1 Star 0 Fork 0


[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

 main  1 branch  0 tags

[Go to file](#) [Add file](#) [Code](#)

 ycs-wikim Initial commit 213bfba 3 days ago 1 commit

 .gitignore	Initial commit	3 days ago
 LICENSE	Initial commit	3 days ago
 README.md	Initial commit	3 days ago

README.md 

2021_oop2

About 

No description, website, or topics provided.

 Readme

 GPL-3.0 License

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

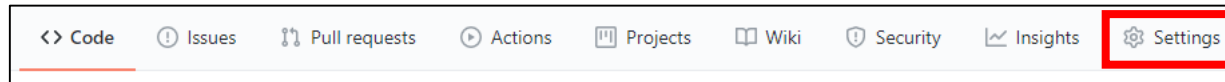


나의 저장소 관리하기 - 5



• 저장소 삭제하기

- 저장소 상단의 "Settings" 메뉴 클릭
- 클릭한 화면 가장 아래 "Danger Zone"에서 삭제 가능



Danger Zone

Change repository visibility

This repository is currently public.

[Change visibility](#)

Transfer ownership

Transfer this repository to another user or to an organization where you have the ability to create repositories.

[Transfer](#)

Archive this repository

Mark this repository as archived and read-only.

[Archive this repository](#)

Delete this repository

Once you delete a repository, there is no going back. Please be certain.

[Delete this repository](#)



저장소 로컬로 가져오기



- 탐색기에서 “git bash here” 실행

- 계정명 : ycs-wikim, 저장소명 : 2021_oop2 이라면

- 실행된 CUI 에서 아래 명령 실행

- **git clone https://github.com/ycs-wikim/2021_cpp2.git**

- git clone https://github.com/계정명/저장소명.git 형식
- clone은 복제해서 가져오라는 명령

```
MINGW64:/d/___수업자료/2021/객체지향언어

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어
$ git clone https://github.com/ycs-wikim/2021_oop2.git
Cloning into '2021_oop2'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 12.78 KiB | 12.78 MiB/s, done.

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어
$
```



git 명령어 동작 가능 위치



- git bash의 상태 정보로 확인 가능
 - 현재 디렉터리 정보 뒤에 (브랜치이름)이 보이는 경우만
 - 아래 그림에서 디렉터리 이동 전에는 (main) 표시가 없음
 - (main)이 나타난 경우에만 git 명령어가 정상적으로 동작

```
MINGW64:/d/___수업자료/2021/객체지향언어/2021_oop2
YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어
$ cd 2021_oop2/
YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ |
```




저장소의 상태 정보 확인 - 1



- git status
 - 현재 저장소의 상태 정보를 출력
 - 변경이 없는 경우는 아래와 같이 표시
 - nothing to commit, working tree clean
 - 변경 사항이 없으며, 추가로 처리할 사항이 없음을 나타냄

```
MINGW64:/d/___수업자료/2021/객체지향언어/2021_oop2
YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$
```



저장소의 상태 정보 확인 - 2



• 저장소 정보 변경

- touch 명령으로 저장소에 파일을 생성
- 상태 정보에서 관리되지 않는 파일명을 출력하여 알림
 - 붉은 색으로 파일 명을 출력
 - git에 의해 관리되지 않는 파일을 나타냄

```
MINGW64:/d/___수업자료/2021/객체지향언어/2021_oop2
YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ touch report.txt

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
       report.txt

nothing added to commit but untracked files present (use "git add" to track)
YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ |
```



저장소의 상태 정보 수정 - 1



- git add
 - 수정이 발생한 파일을 관리 대상에 포함
 - 관리 대상에 포함한 다음 상태 확인
 - 추가된 새로운 파일이 녹색으로 표시

```
MINGW64:/d/___수업자료/2021/객체지향언어/2021_oop2
YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        report.txt

nothing added to commit but untracked files present (use "git add" to track)

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git add report.txt

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   report.txt

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$
```



저장소의 상태 정보 수정 - 2



- git commit
 - 현재 수정 상태를 저장소에 반영
 - git commit만 입력할 경우 vim 에디터가 실행
 - git commit -m "message" vim 없이 commit 수행

```
MINGW64:/d/___수업자료/2021/객체지향언어/2021_oop2
(use "git add <file>..." to include in what will be committed)
report.txt

nothing added to commit but untracked files present (use "git add" to track)
YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git add report.txt

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   report.txt

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git commit -m "add report file"
[main de1645a] add report file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 report.txt

YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$
```



저장소의 상태 정보 수정 - 3



- git push
 - commit 된 정보를 github 서버로 전달
 - 인증 창이 나타날 경우, 인증 토큰을 생성해야 함
 - ID/PWD 방식에서 Personal Access Token으로 인증 방법이 변경되었음

The screenshot shows a Windows terminal window with the title bar 'MINGW64:/d/___수업자료/2021/객체지향언어/2021_oop2'. The terminal output for the command 'git push' is as follows:

```
YUHAN@YP12624115 MINGW64 /d/___수업자료/2021/객체지향언어/2021_oop2 (main)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 506 bytes | 506.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/yys-wikim/2021_oop2.git
   eea5b1d..125e0e4  main -> main
```

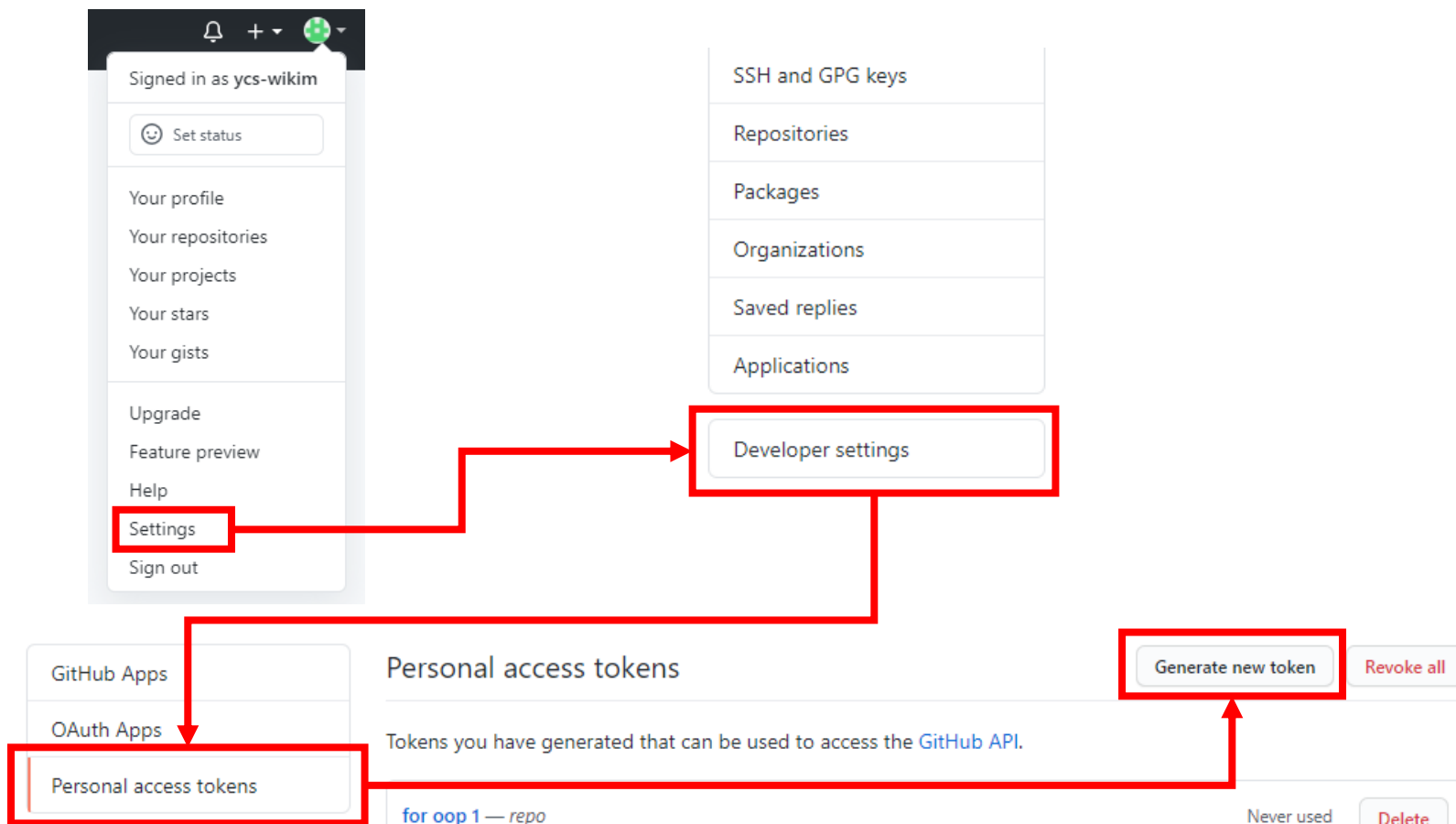
Overlaid on the terminal is a 'Connect to GitHub' dialog box. It features the GitHub logo and the text 'Sign in'. There are two main options: 'Sign in with your browser' (with an external link icon) and a 'Personal Access Token' input field. Below the input field is a 'Sign in' button. At the bottom, there is a link for 'Don't have an account? Sign up'.



github 인증 토큰 생성 - 1

- github 우측 상단의 아이콘 클릭

- “Settings” → “Developer settings” → “Personal access tokens”
→ “Generate new token” 순으로 선택





github 인증 토큰 생성 - 2



- “Note”에 간단한 설명 입력
 - 가장 위 “repo”만 선택하고, 토큰을 생성

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

new token

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo:deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events

<input type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input type="checkbox"/> admin:gpg_key	Full control of public user GPG keys (Developer Preview)
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys

[Cancel](#)



github 인증 토큰 사용

- 생성된 토큰 옆의 아이콘을 클릭
 - 클립보드로 정확한 토큰을 입력
 - 복사해서 붙일 경우, 잘못된 입력이 발생할 수 있음

The image shows two overlapping windows. The background window is titled 'Personal access tokens' and contains a list of tokens. One token, '59f951b32d85ff7dc47c7a5ca576c29d785d1228', is highlighted in green and has a blue clipboard icon to its right, which is enclosed in a red square. A red arrow points from this icon to the foreground window. The foreground window is titled 'Connect to GitHub' and shows the 'GitHub Sign in' page. It has a 'Sign in with your browser' button, an 'or' separator, and a text input field. The input field contains a series of dots, indicating that the token has been pasted. Below the input field is a 'Sign in' button. At the bottom, there is a link for 'Don't have an account? Sign up'.



github 인증 토큰 삭제 - 1



- 다중 사용자 환경에서 github 인증서 변경 필요 시
 - "제어판"에서 "자격 증명"을 검색
 - "자격 증명 관리자"를 실행

The screenshot shows the Windows Settings application. In the search bar, '자격 증명' (Credential) is entered. The search results show '자격 증명 관리자' (Credential Manager) as the top result, which is highlighted with a red box. A red arrow points from this box to the 'Credential Manager' window shown in the foreground. The 'Credential Manager' window displays the path: 제어판 > 모든 제어판 항목 > 자격 증명 관리자. It shows two categories: '웹 자격 증명' (Web Credentials) and 'Windows 자격 증명' (Windows Credentials). The '웹 자격 증명' section shows a list of credentials with a '삭제' (Delete) button next to each one.



github 인증 토큰 삭제 - 2

• 자격 증명 토큰 삭제

- “Windows 자격 증명”에서 github 자격 증명을 삭제
- github에서 서버 연결 시 인증 토큰 입력 창이 다시 나타남
- 새로운 토큰 입력 가능





협업을 위한 git 사용



• 시스템 및 소스 파일 설계 우선

- 시스템에 대한 분석과 기능 구현을 먼저 **설계**하고 개발을 시작
- 소스 코드 개발에 대한 **분업**을 먼저 진행하는 것이 좋음
- 수정할 파일과 수정하지 않을 파일을 구분하여 작업
- 동일 파일을 생성하거나 수정할 경우 문제가 발생 함
- **지정된 파일 이외에는 수정하지 않는 것이 기본**

• 브랜치(branch) 활용

- 브랜치란 **원래 코드**와 관계 없이 독립 개발을 지원하는 논리적인 개념
- 특정 버전 상태에서 논리적으로 분리된 파일들을 별도로 사용
- A 브랜치에서 브랜치를 생성한 시점의 A 파일들을 별도로 사용 가능
- 병합 후 업로드할 경우, 브랜치는 서버에 업로드하지 않아도 문제 없음
- 필요한 경우에는 서버에 브랜치를 업로드해야 함

◆ 브렌치의 활용 - 1

• 브렌치는 일종의 게임 세이브와 동일

- 게임 세이브로 **현재 상태를 저장**할 수 있음
- 저장 상태 후 플레이 도중 이전 상태로 되돌리고 싶을 경우 세이브를 로드
- 이때, 현재 상태 저장은 사용자의 선택 등을 저장하지 않음
- 원하는 시점을 로드하여 확인이나 수정 등이 가능



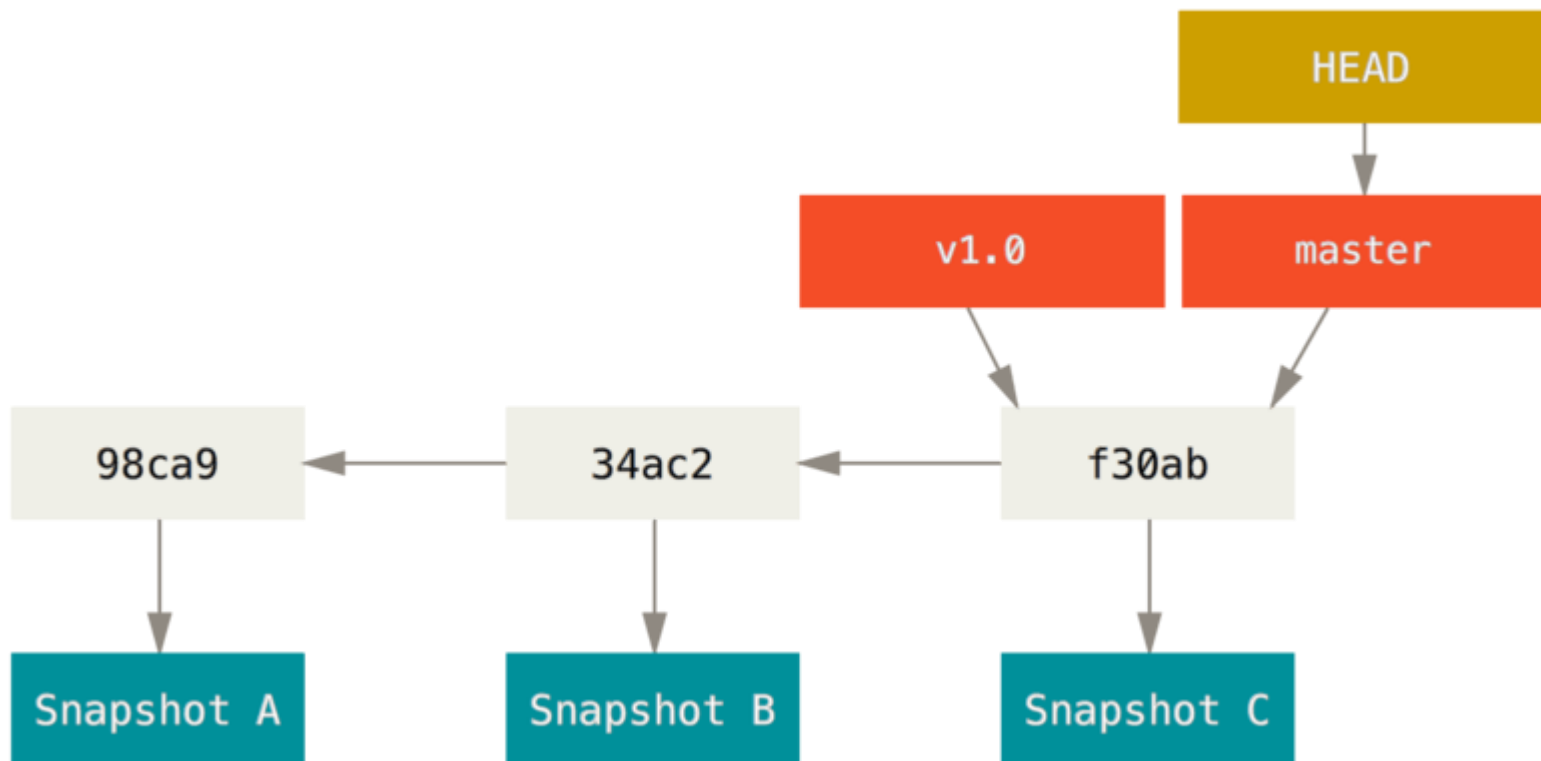


브랜치의 활용 - 2



- 브랜치를 통한 파일 별도 보관

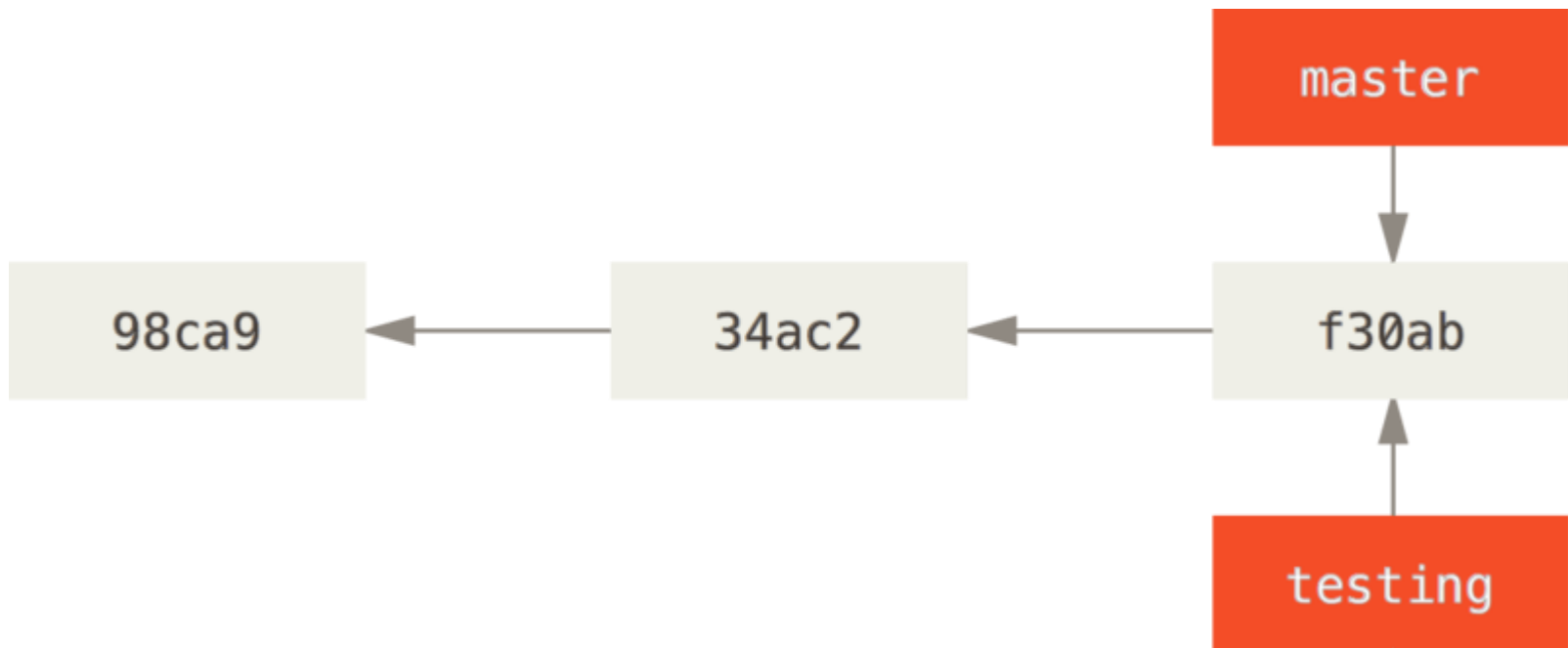
- 저장소의 기본 브랜치 : **master** 또는 **main**
- 현재 저장소의 상태





• 브랜치의 생성과 이동

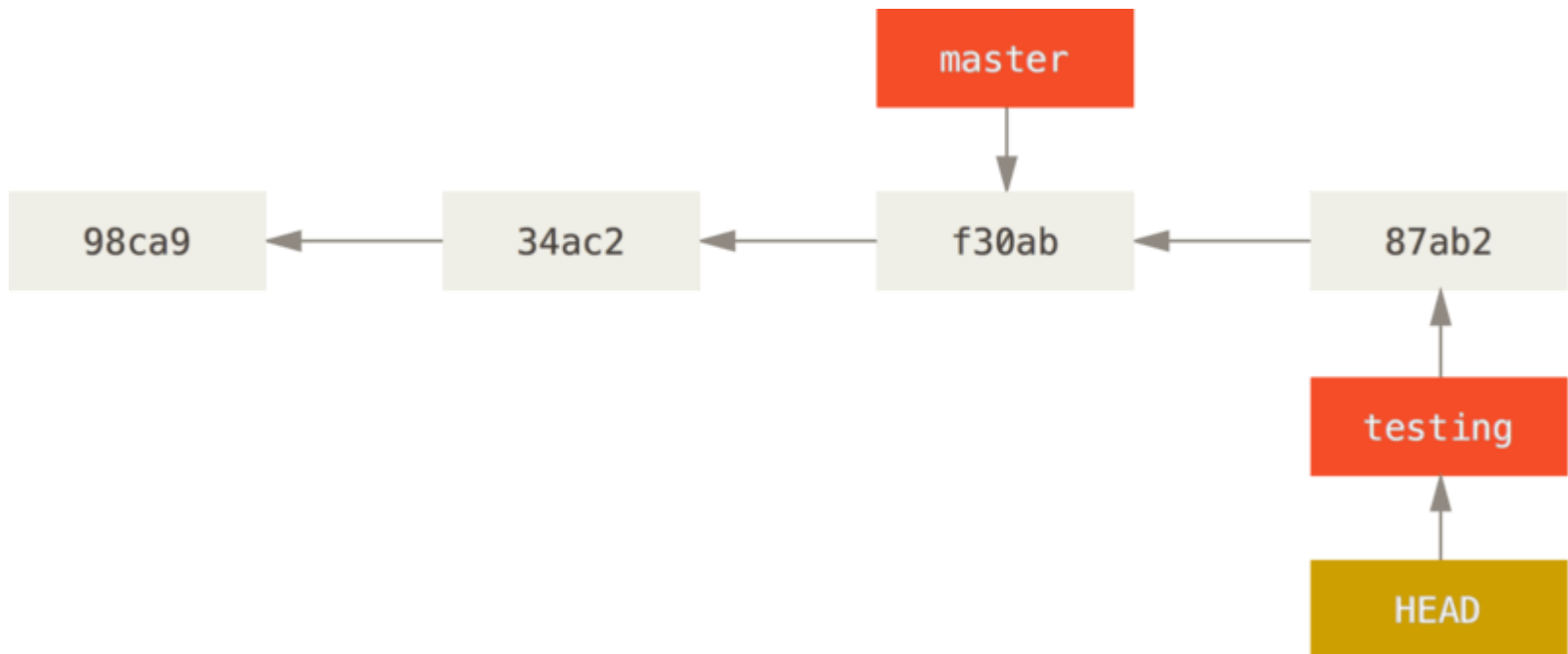
- 저장소의 브랜치 생성: **git branch** "브랜치이름"
- 현재 상태에서 논리적인 개별 상태를 생성
- "git branch testing" 명령을 수행하면 "testing" 브랜치가 생성됨
- "**git checkout** testing" 명령으로 해당 브랜치로 이동
- "git branch -b testing" 명령으로 한번에 수행 가능





• 브랜치에서 파일 수정

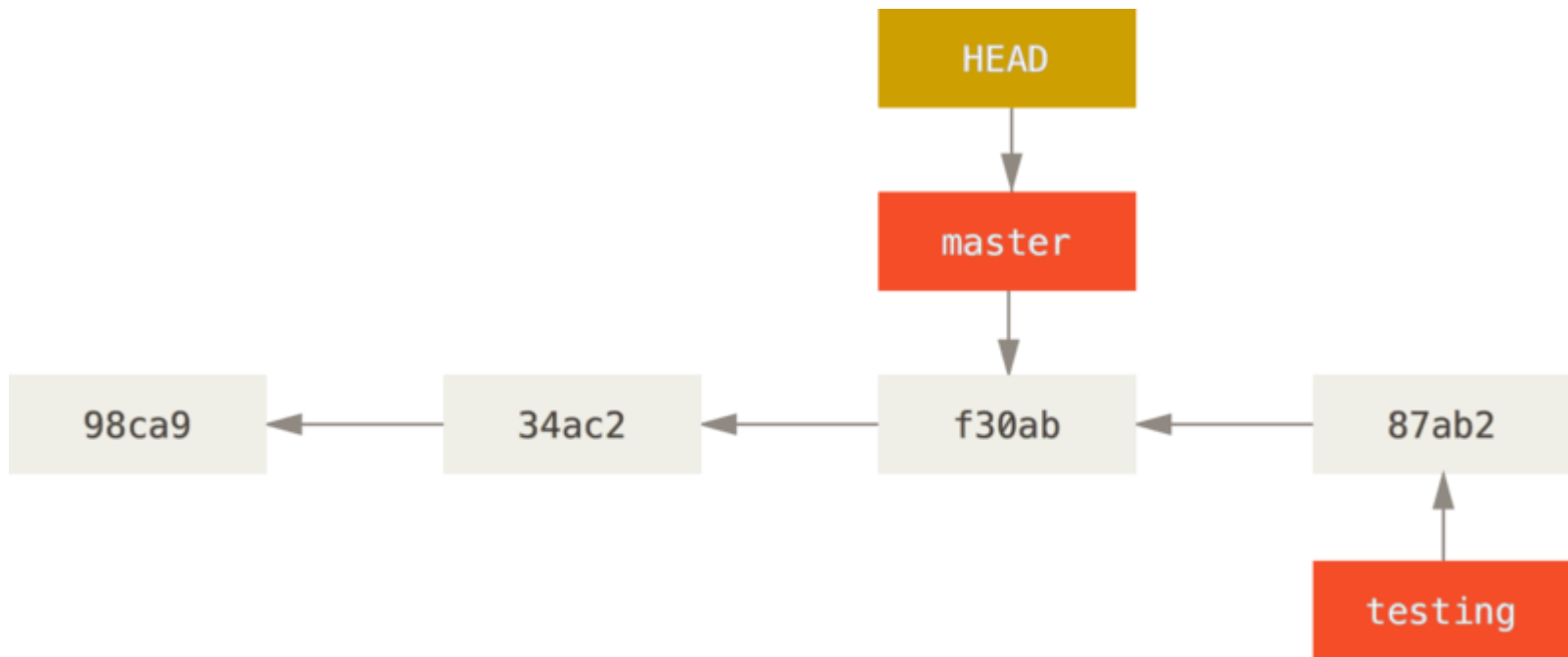
- 생성된 브랜치에서 파일을 수정하여 commit
- 해당 브랜치에 commit된 내용이 추가로 저장
- 현재 작업 위치에 HEAD가 항상 위치





• 브랜치 이동

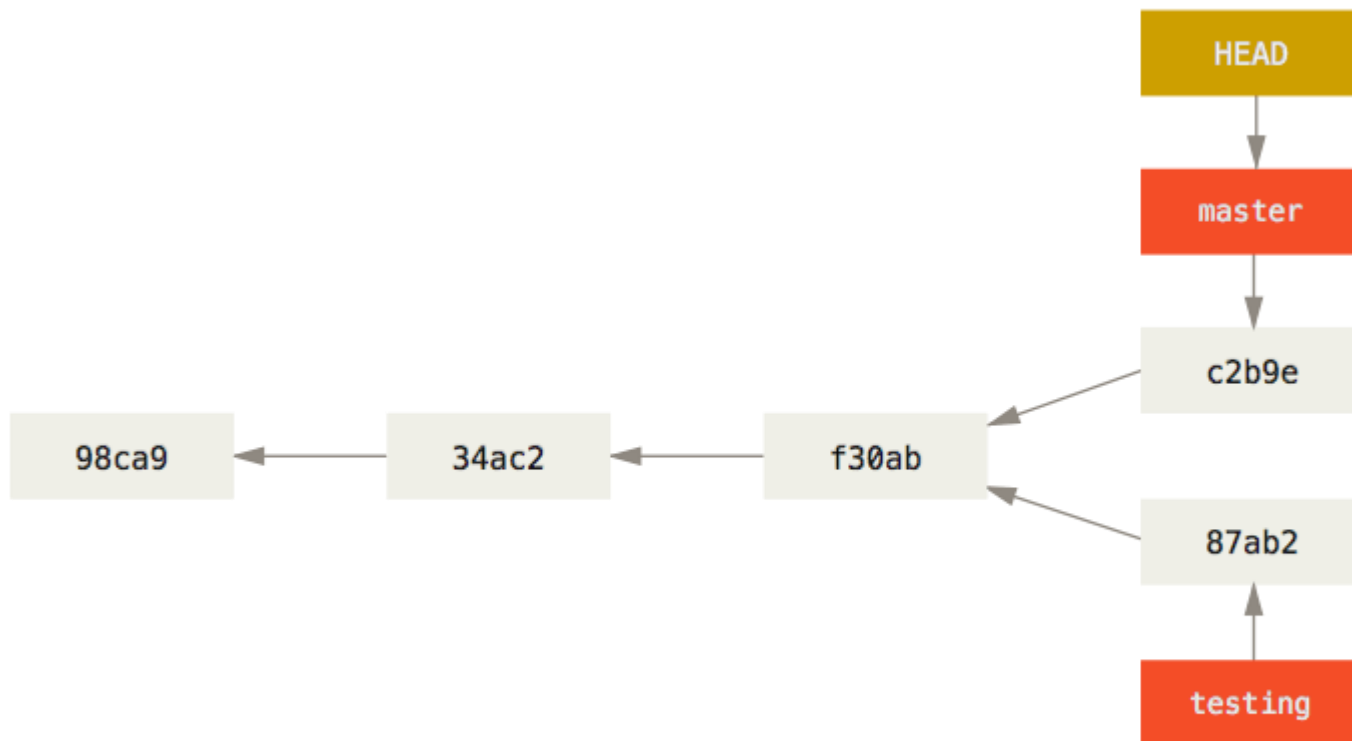
- 현재 브랜치에 모든 내용이 commit/staging된 상태에서만 이동 가능
- untracked 또는 modified 파일이 있는 경우는 이동 불가능
- "git branch 브랜치이름" 명령으로 브랜치 이동 가능
- "git checkout master"로 master 브랜치로 이동 가능
- "**git branch**" 명령은 현재 로컬의 브랜치 목록을 출력





• 브랜치의 파일 commit

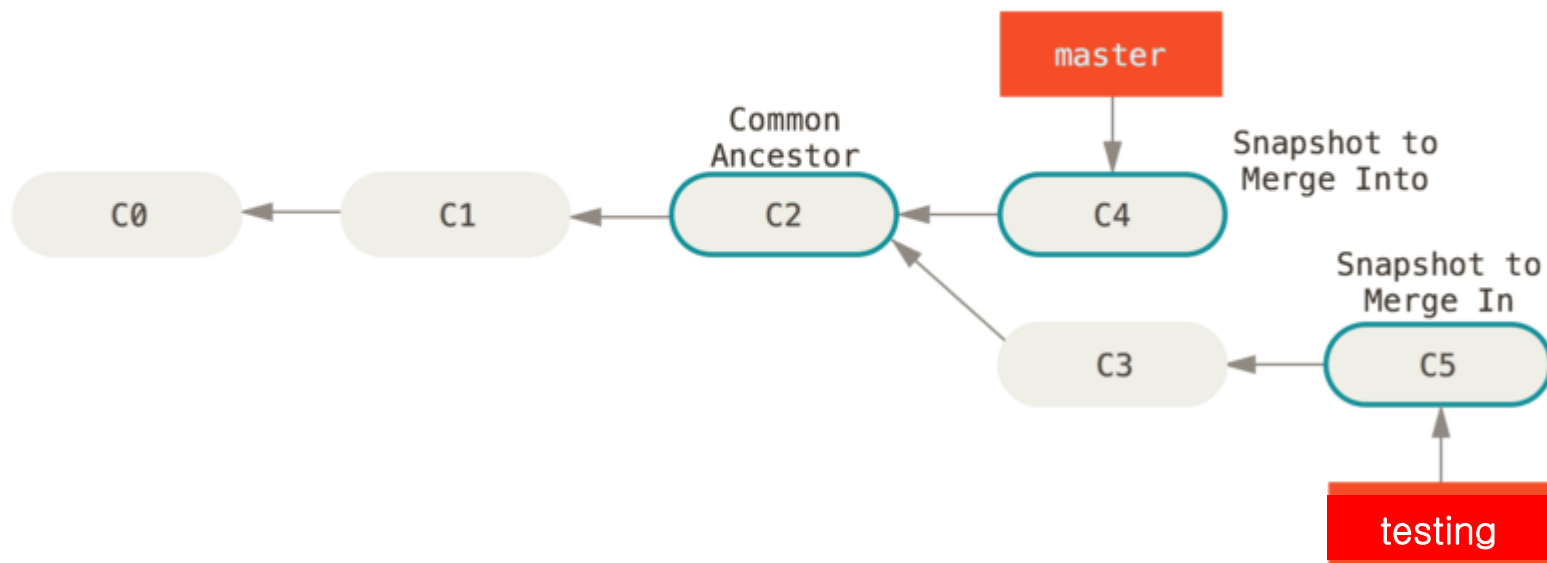
- 이동한 master 브랜치에서 파일을 수정하고 commit
- master 브랜치 만의 파일 버전이 별도로 생성되어 업데이트
- 브랜치 이동 시, 각 브랜치 만의 수정 내역을 확인할 수 있음





• 브랜치 병합 - 1

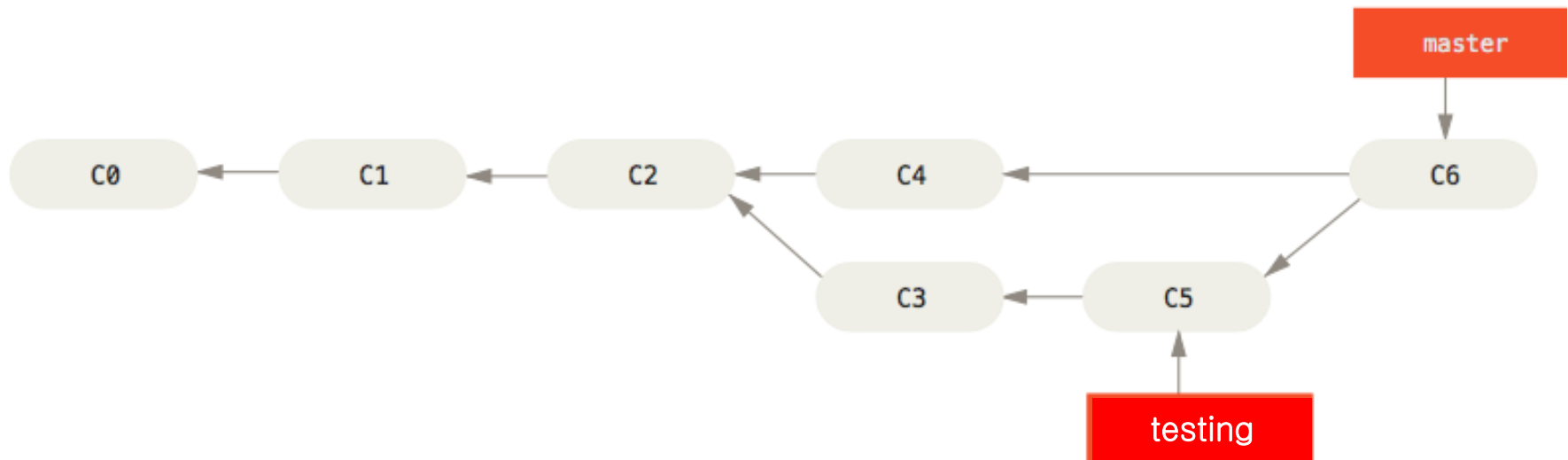
- 브랜치를 생성한 목표를 달성한 경우 병합을 수행
- 병합은 2개의 다른 파일 흐름을 합치는 과정
- "**git merge** 병합할브랜치명" 명령으로 병합
- 현재 브랜치에 다른 브랜치를 합치는 형태로 동작
- **master에 testing 브랜치를 병합하려면 master 브랜치에서 수행해야 함**





• 브랜치 병합 - 2

- "git checkout master"로 master 브랜치로 이동
- "git merge testing"으로 브랜치를 병합
- 병합되면 testing 브랜치 삭제가 가능해짐
- master 브랜치의 **C6** 스냅샷에는 testing에서 수행된 모든 작업이 포함

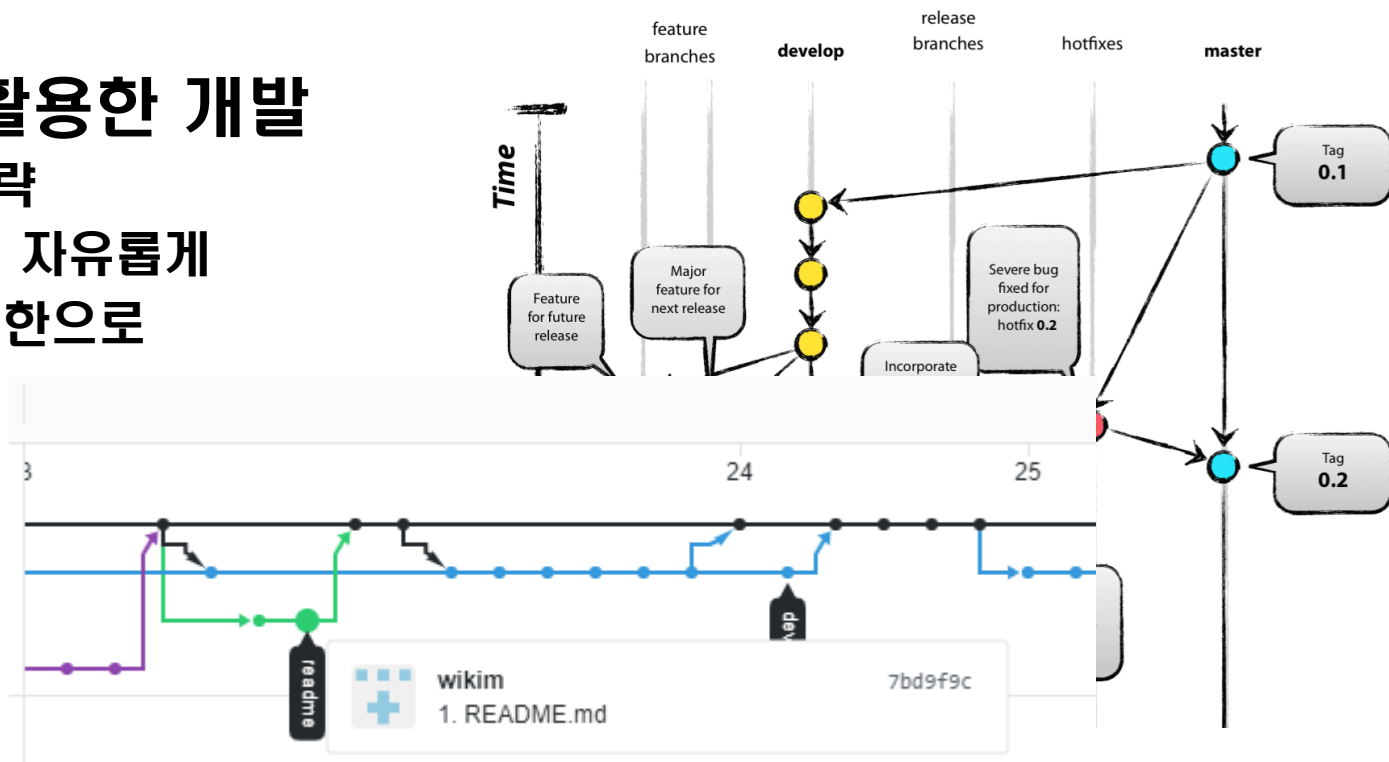




브랜치의 활용 - 9

● 브랜치를 활용한 개발

- git flow 전략
- 버전 관리는 자유롭게
- 충돌을 최소한으로



Pull Request 할때, 하나의 클래스를 몇명에서 수정하게되면 겹치게될텐데 그런경우엔 어떻게 하시나요?

좋아요 · 답글 달기 · 3년

저희는 되도록 코드 충돌이 발생하지 않도록 작업을 나누어서 진행을 합니다.

그래서 하나의 클래스를 여러 명에서 건들지 않도록 하고 있습니다.

작업을 나눌 때 같은 코드를 여러 명이 건드려야 한다면 작업자들끼리 이야기를 한 후 한 명이 먼저 작업을 처리 합니다.

그렇게 하더라도 간혹 코드 충돌이 발생하게 되는데요. 대부분 이 사실은 오전 티타임을 할 때나 코드리뷰를 할 때 알게됩니다.

코드 충돌 해결은 전적으로 뒤에 코드 병합을 하는 사람이 책임을 지게됩니다.

코드 충돌이 작으면 스스로 처리하지만, 코드 충돌 범위가 크면 작업 코드가 겹친 개발자와 함께 충돌 해결을 하고 있습니다.

좋아요 · 답글 달기 · 8 · 3년