소스코드의 특성 VCS 왜 쓰나요?

없어지면 안됩니다

코드를 수시로 더하고 빼고 바꾸고

여럿이 같이 작업

언제 누가 (어디서) 무엇을 어떻게 왜 바꿨는가?

어떻게 바꿨는가? 버전간 차이 비교

버전 관리 시스템이 뭔가요?

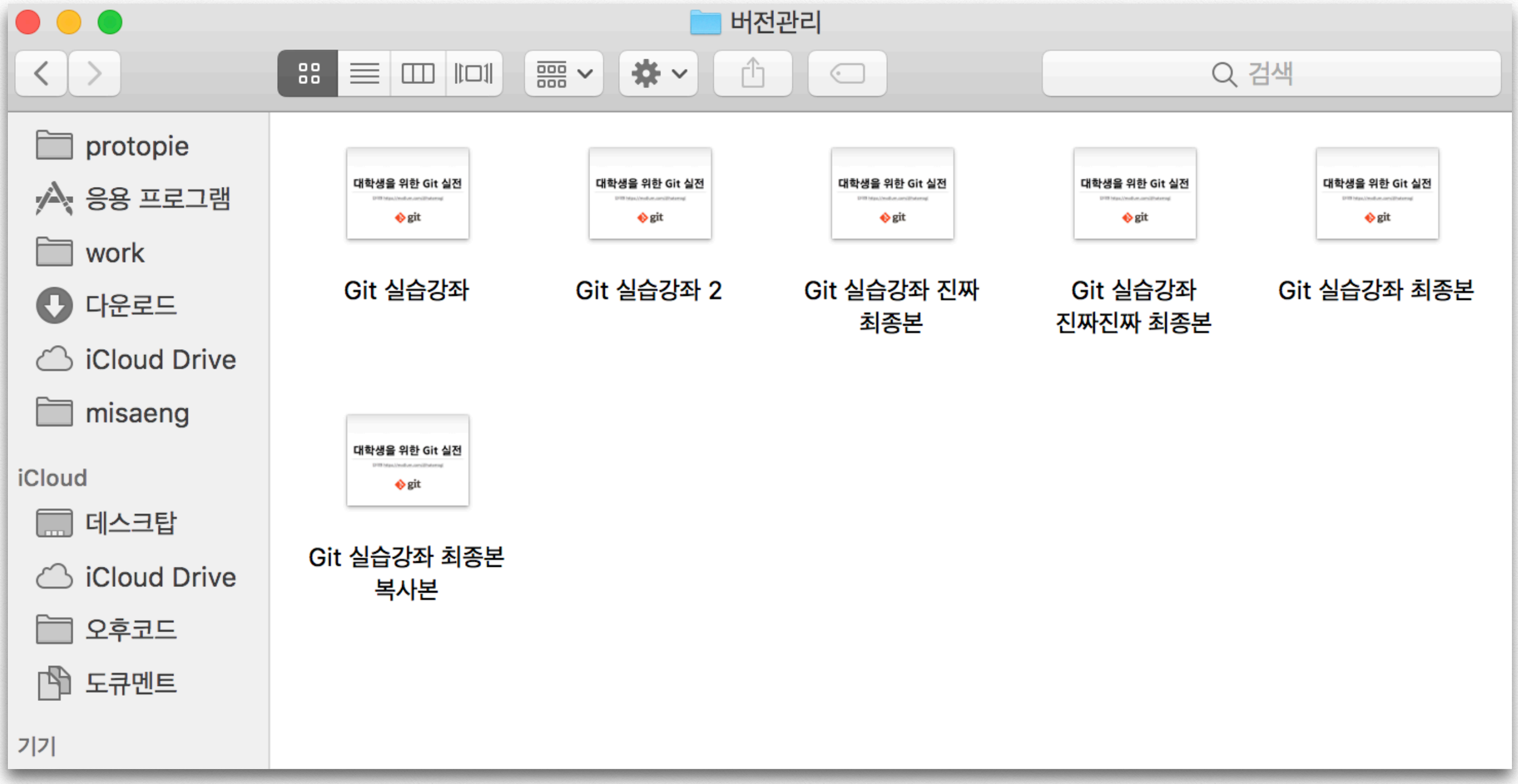
(주로 텍스트) 파일들을 누가 어떻게 바꿨는지 기록하고

기록한 내용을 쉽게 조회하고

특정 버전으로 되돌리고 복구하고

여럿이 함께 개발하고 공유하기 편리한 시스템

가장 간단한(?) 버전 관리


버전관리
검색
protopie
응용 프로그램
work
다운로드
iCloud Drive
misaeng
iCloud
데스크탑
iCloud Drive
오후코드
도큐멘트
기기
대학생을 위한 Git 실전
git
Git 실습강좌
Git 실습강좌 2
Git 실습강좌 진짜 최종본
Git 실습강좌 진짜진짜 최종본
Git 실습강좌 최종본
Git 실습강좌 최종본 복사본

대학생이 알면 뭐가 좋나요?

몰라도 됩니다

기술 면접 우위

준비된 협업 개발자, 공동 개발에 필수!

혼자 개발할 때도 유용함!

지원하는 회사의 신기술 민감도 추측

버전 관리 시스템

여러 VCS 中 Git의 특징

사실상 표준

“분산” 버전 관리 시스템

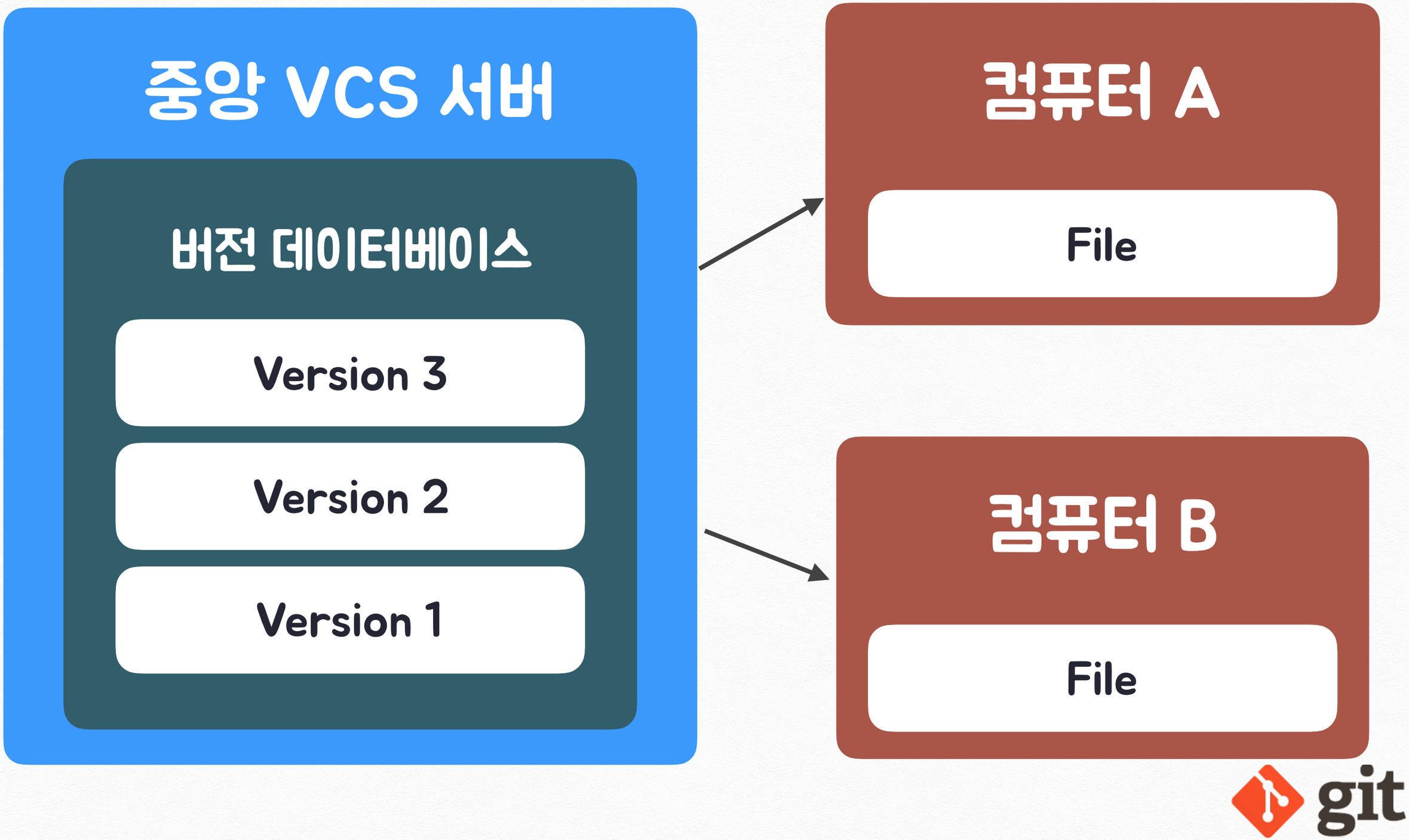
가볍고 빠르고 효과적

버전간 차이가 아니라 (똑똑하게) 전체 사본snapshot**을 보관**

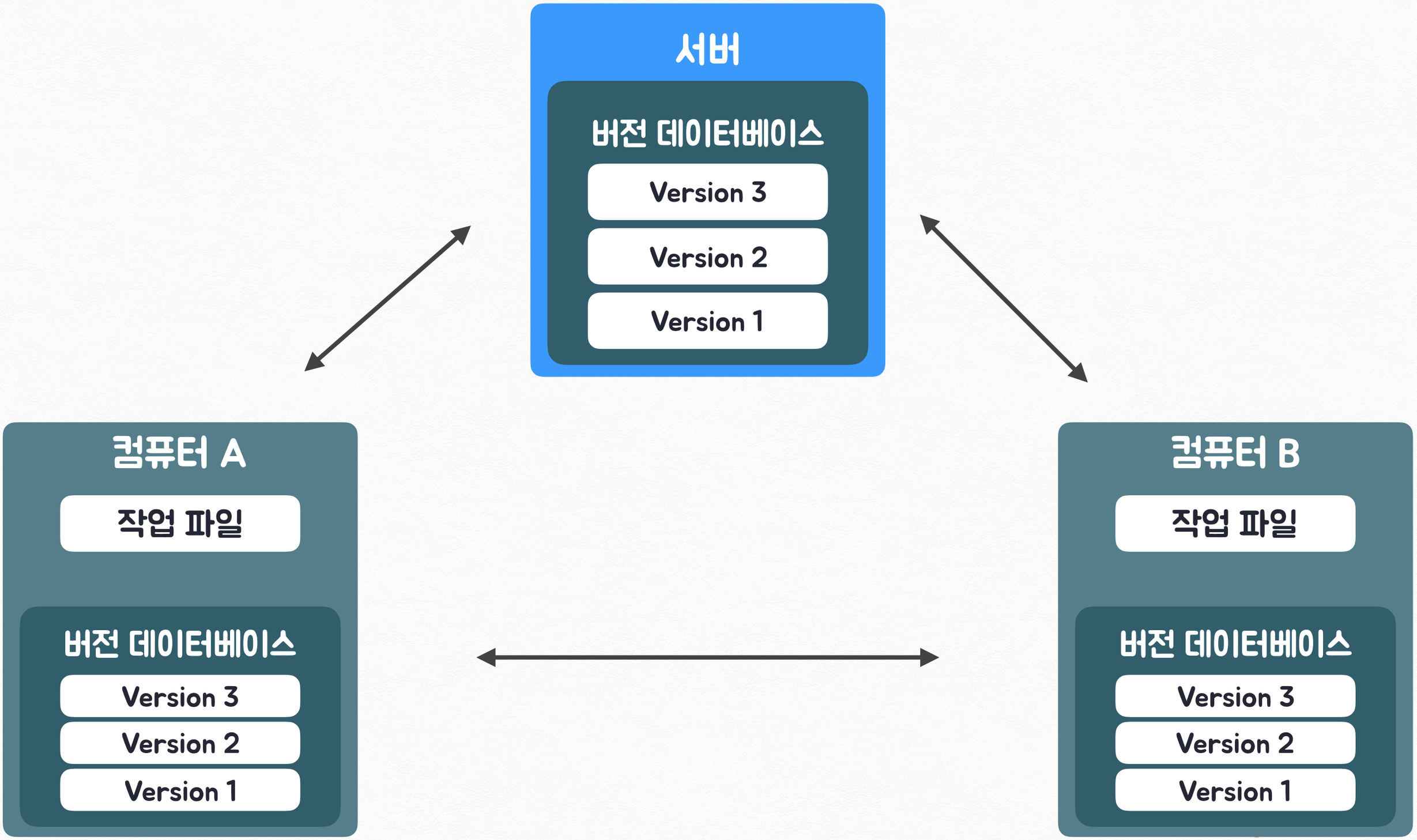
거의 모든 작업을 로컬에서 할 수 있음

강력한 브랜치 기능

중앙집중식 버전 관리 시스템



분산 버전 관리 시스템


서버
버전 데이터베이스
Version 3
Version 2
Version 1
컴퓨터 A
작업 파일
버전 데이터베이스
Version 3
Version 2
Version 1
컴퓨터 B
작업 파일
버전 데이터베이스
Version 3
Version 2
Version 1

DVCS 장점

네트워크 단절 OK

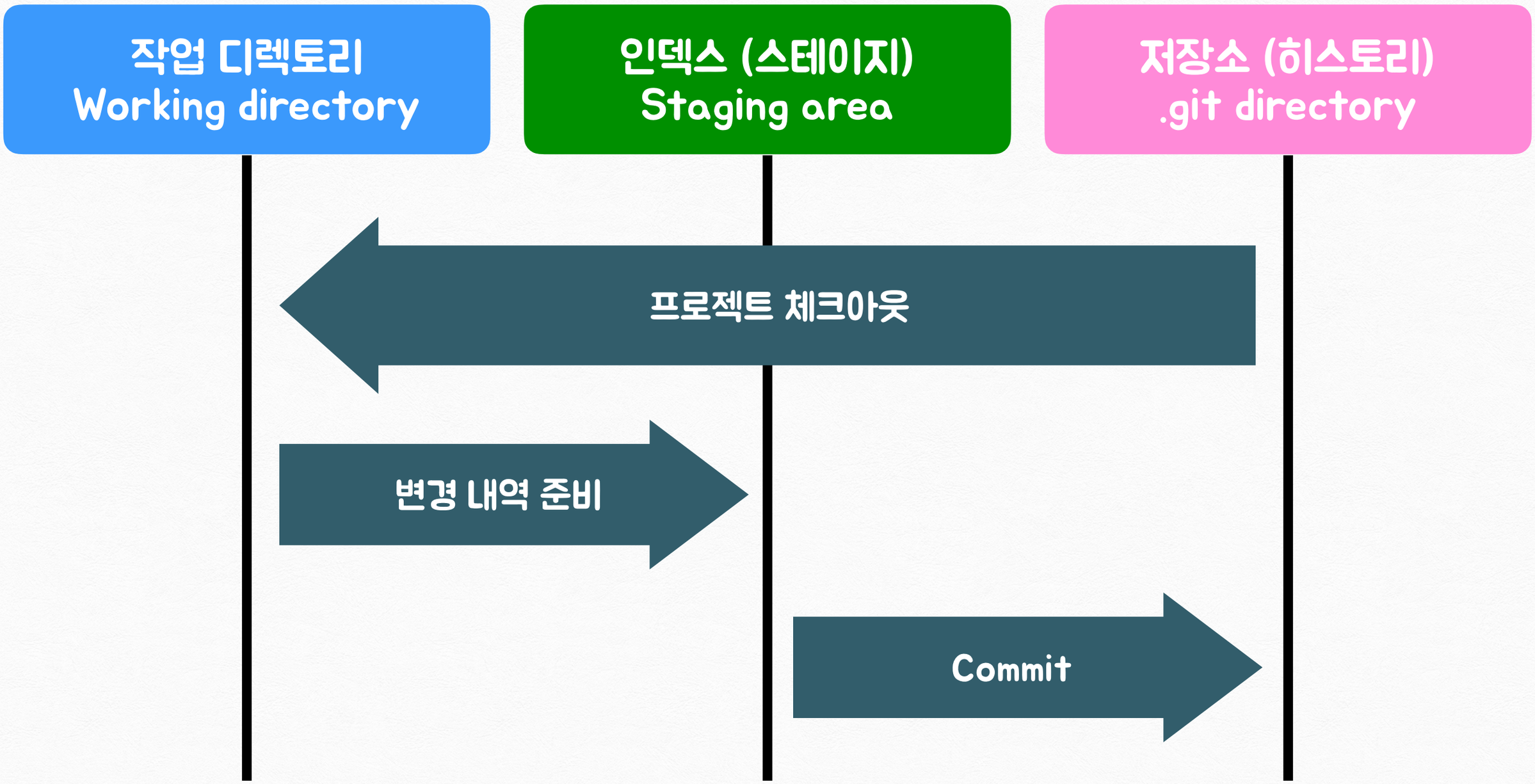
로컬에 빠르고 가볍게 작업

혼자만의 실험 브랜치 진행

다른 사람과 공유전에 커밋을 정리정돈

세 영역

working directory, staging area, repository



git

터미널 명령 (CLI)로 실습 진행

다양한 GUI 툴이 있음. BUT!

CLI툴은 여러 OS 공통이자 오리지널

Git의 모든 기능을 쓸 수 있음

기본 CLI 툴로 익히면, 여러 다른 GUI 툴도 쉽게 적응

그 반대는 잘 안됨

실습 A - 버전 확인

실행

```
$ git version
git version 2.15.1
$
```

실습 B - 기본 설정

실행

```
$ git config --global user.name “홍길동”
$ git config --global user.email “hong@gil.dong”
$ git config --global color.ui true
```

각자 이름과 이메일 주소를 적습니다

-global : 지금 로그온한 계정에 전체 설정

-local: 현재 저장소 로컬 설정

실습 디렉토리 준비

GitHub, Inc. github.com/git-kickstart/hello/blob/c2/hello.html

This repository Search Pull requests Issues Marketplace Explore

git-kickstart / hello

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Tag: c2 hello / hello.html

mkdir hello
cd hello

홍길동 add hello.html

0 contributors

17 lines (16 sloc) 432 Bytes Raw Blame History

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>Git 실습</title>
6      <link rel="stylesheet" type="text/css"
7        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
8        integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb"
9        crossorigin="anonymous">
10   </head>
11   <body>
12     <main>
13     <h1>안녕하세요</h1>
```

실습 C1 - git 저장소 초기화

실행

```
$ git init
```

현재 디렉토리 기준, Git 저장소 초기화

실습 C1 - git 저장소 초기화

결과

```
$ git init
Initialized empty Git repository in /Users/hatemogi/
work/kickstart/.git/
$
```

실습 C1 - 초기화 후 상태 확인

실행

```
$ git status
```

현재 작업영역 상태 확인

실습 C1 - 초기화 후 상태 확인

결과

```
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
$
```

실습 C2 - 새 파일 추가 hello.html

hello.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Git 실습</title>
    <link rel="stylesheet" type="text/css"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
      integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb"
      crossorigin="anonymous">
  </head>
  <body>
    <main>
    <h1>안녕하세요</h1>
    </main>
  </body>
</html>

```

https://github.com/git-kickstart/hello/blob/c2/hello.html

실습 C2 - 다시 상태 확인

작업 디렉토리에 조금전 hello.html을 추가하고,

실행

```
$ git status
```

현재 작업영역 상태 다시 확인

실습 C2 - 다시 상태 확인한 결과

결과

```
$ git status
On branch master

No commits yet

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

    hello.html

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

실습 C2 - 스테이지 영역에 추가

실행

```
$ git add hello.html
$
```

스테이지 영역에 hello.html 추가

실행

```
$ git status
```

작업디렉토리 상태 또 확인 (2)

실습 C2 - 스테이지 된 상태

결과

```
$ git status
```

실습 C2 - 스테이지 된 상태

결과

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

   new file:   hello.html

$
```

실습 C2 - 스테이지에 올라간 hello.html

작업 디렉토리
Working directory

인덱스 (스테이지)
Staging area

저장소 (히스토리)
.git directory

실습 C2 - 스테이지에 올라간 hello.html

작업 디렉토리
Working directory

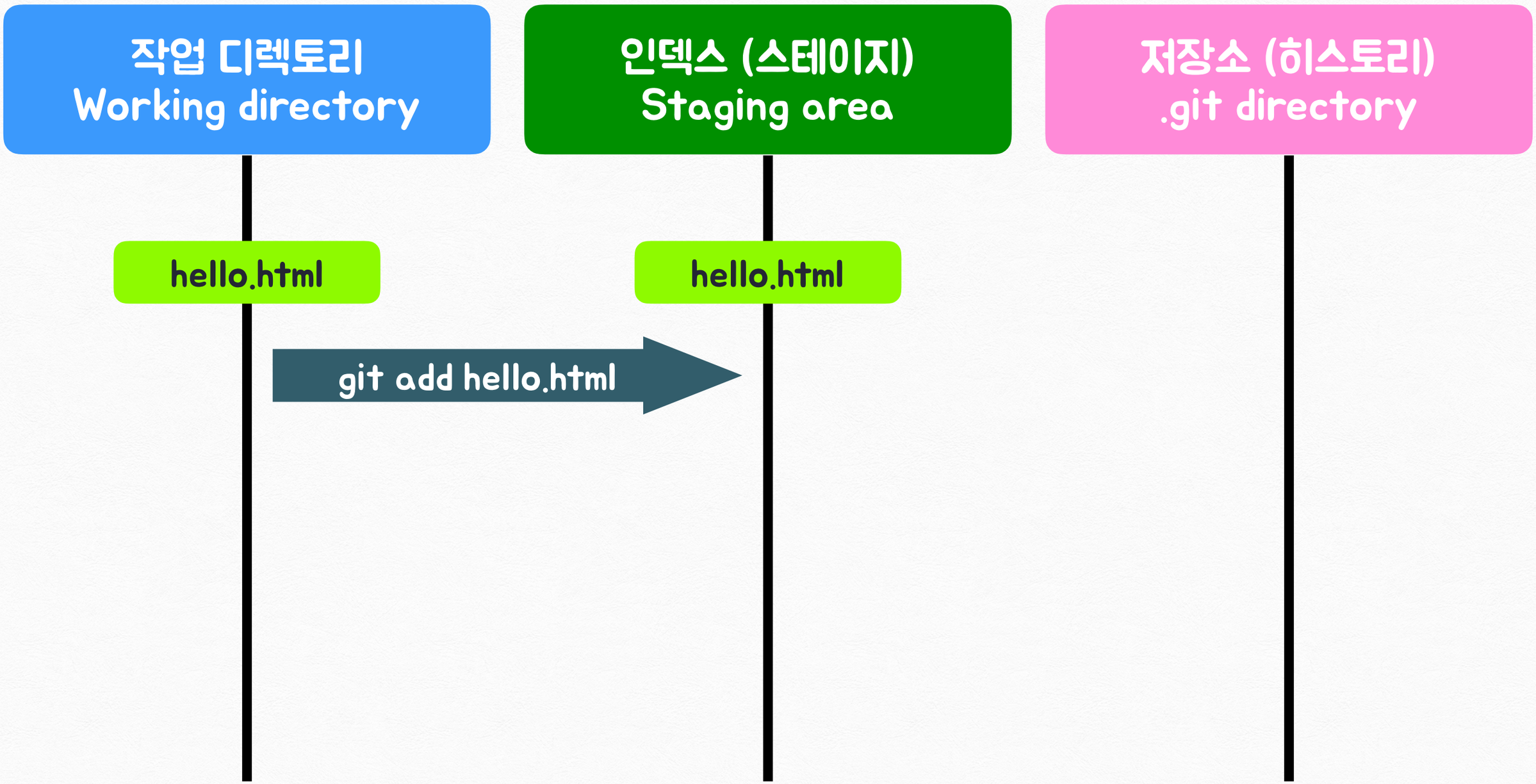
hello.html

인덱스 (스테이지)
Staging area

저장소 (히스토리)
.git directory

실습 C2 - 스테이지에 올라간 hello.html

실습 C2 - 스테이지에 올라간 hello.html



실습 C2 - 첫 커밋!

실행

```
$ git commit -m “add hello.html”
```

커밋하면서 설명을 남긴다

실습 C2 - 첫 커밋!

결과

```
$ git commit -m "add hello.html"
[master (root-commit) 87bad9b] add hello.html
 1 file changed, 16 insertions(+)
 create mode 100644 hello.html
$
```

실습 C2 - 첫 커밋 후 상태 확인

결과

```
$ git status
```

실습 C2 - 첫 커밋 후 상태 확인

결과

```
$ git status
On branch master
nothing to commit, working tree clean
$
```

실습 C2 - 커밋 로그 확인

실행

```
$ git log
```

커밋 로그 보기

실습 C2 - 커밋 로그 확인

결과

```
$ git log
commit 87bad9b0fa781e5596080a3c7dd83f67dd1a0cad (HEAD -> master)
Author: 홍길동 <hong@gil.dong>
Date:   Sat Dec 16 20:36:34 2017 +0900

    add hello.html
$
```

실습 C2 - 87bad9b0f…?

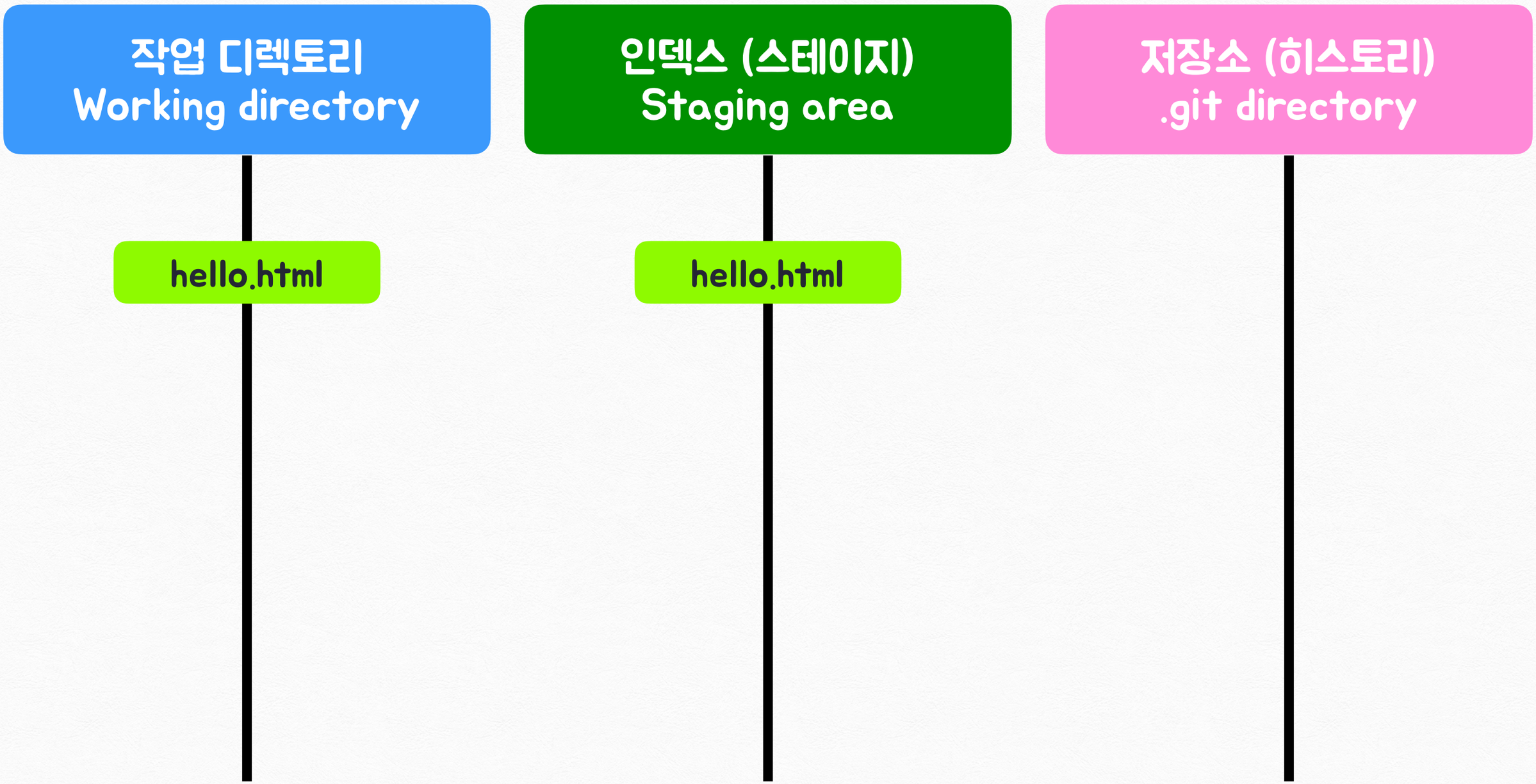
고유 SHA1 값

160bits = 20bytes = 40 hexadecimals

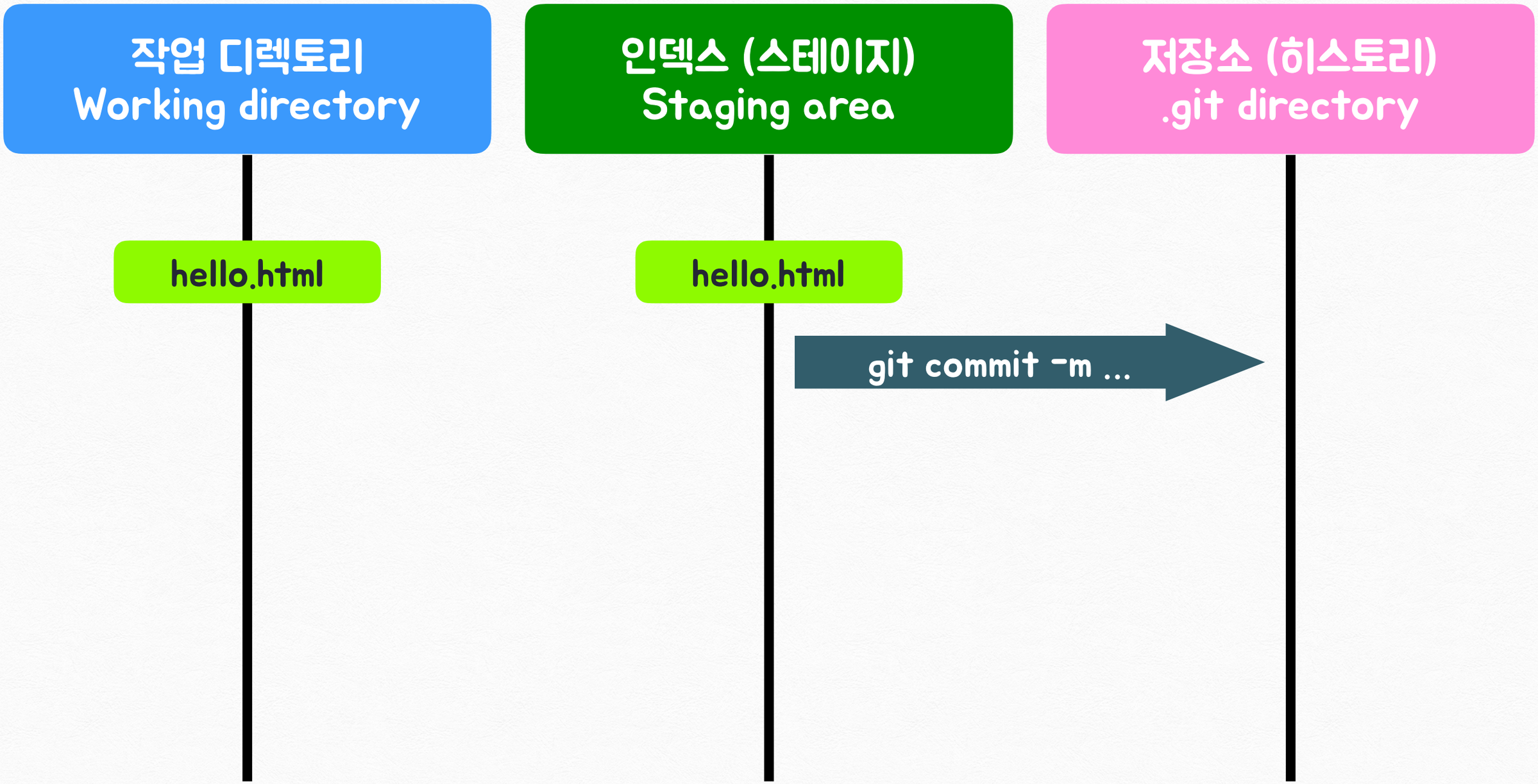
Git 전반에 ID로 쓰임

앞 몇 자만 따서 쓰는 편 (e.g. 앞 7자리)

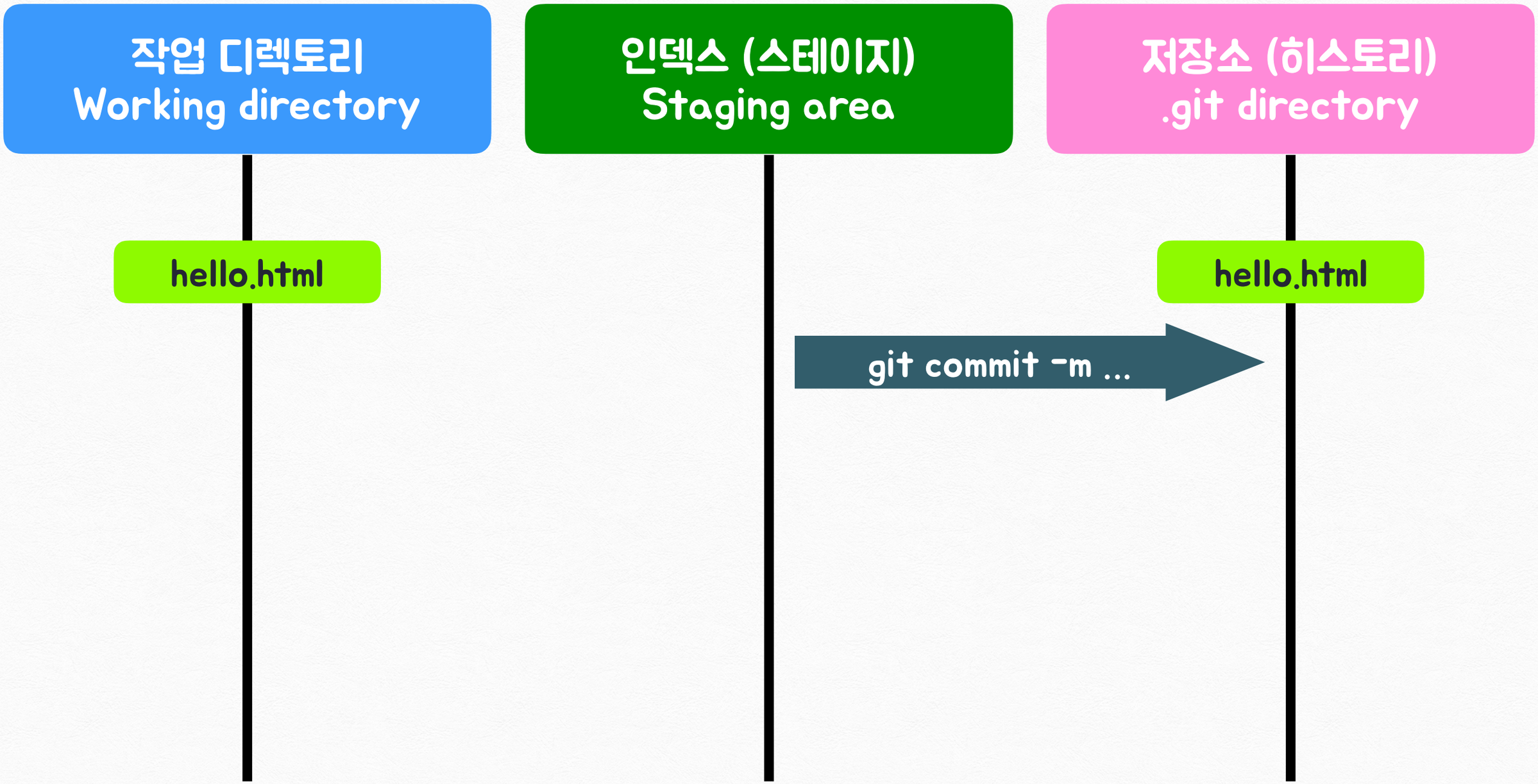
실습 C2 - 안전하게 커밋된 hello.html



실습 C2 - 안전하게 커밋된 hello.html



실습 C2 - 안전하게 커밋된 hello.html



실습 C3 - main.css 추가

main.css — ~/work/kickstart

hello.html | main.css | sub.css

```
1  main {
2    background-color: #ddd;
3    margin: 1rem;
4    padding: 1rem;
5    border-radius: 3px;
6  }
7
```

실습 C3 - hello.html에 main.css 추가

hello.html ● | main.css | sub.css

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="utf-8">
5       <title>Git 실습</title>
6       <link rel="stylesheet" type="text/css"
7         href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
8         integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb"
9         crossorigin="anonymous">
10      <link rel="stylesheet" type="text/css" href="main.css">
11    </head>
12    <body>
13      <main>
14      <h1>안녕하세요</h1>
15      </main>
16    </body>
17  </html>
18
```

실습 C3 - 상태 확인 (1)

결과

```
$ git status
```

실습 C3 - 상태 확인 (1)

결과

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
directory)

    modified:   hello.html

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

    main.css

no changes added to commit (use "git add" and/or "git commit -a”)
$
```

실습 C3 - main.css 스테이지로

실행

```
$ git add '*.css'
```

확장자 .css인 파일 모두 인덱스

실습 C3 - 상태 확인 (2)

결과

```
$ git status
```

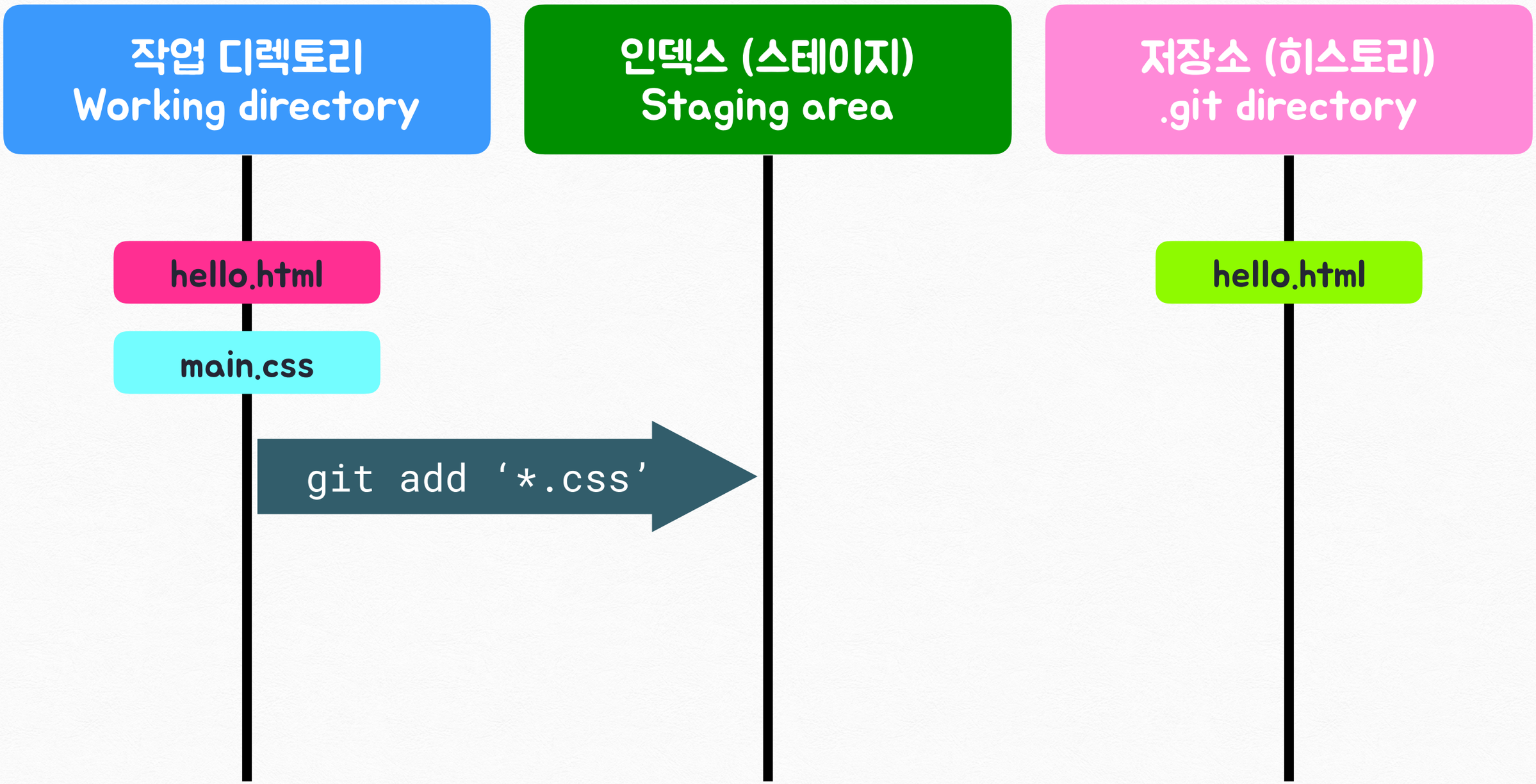
실습 C3 - 상태 확인 (2)

결과

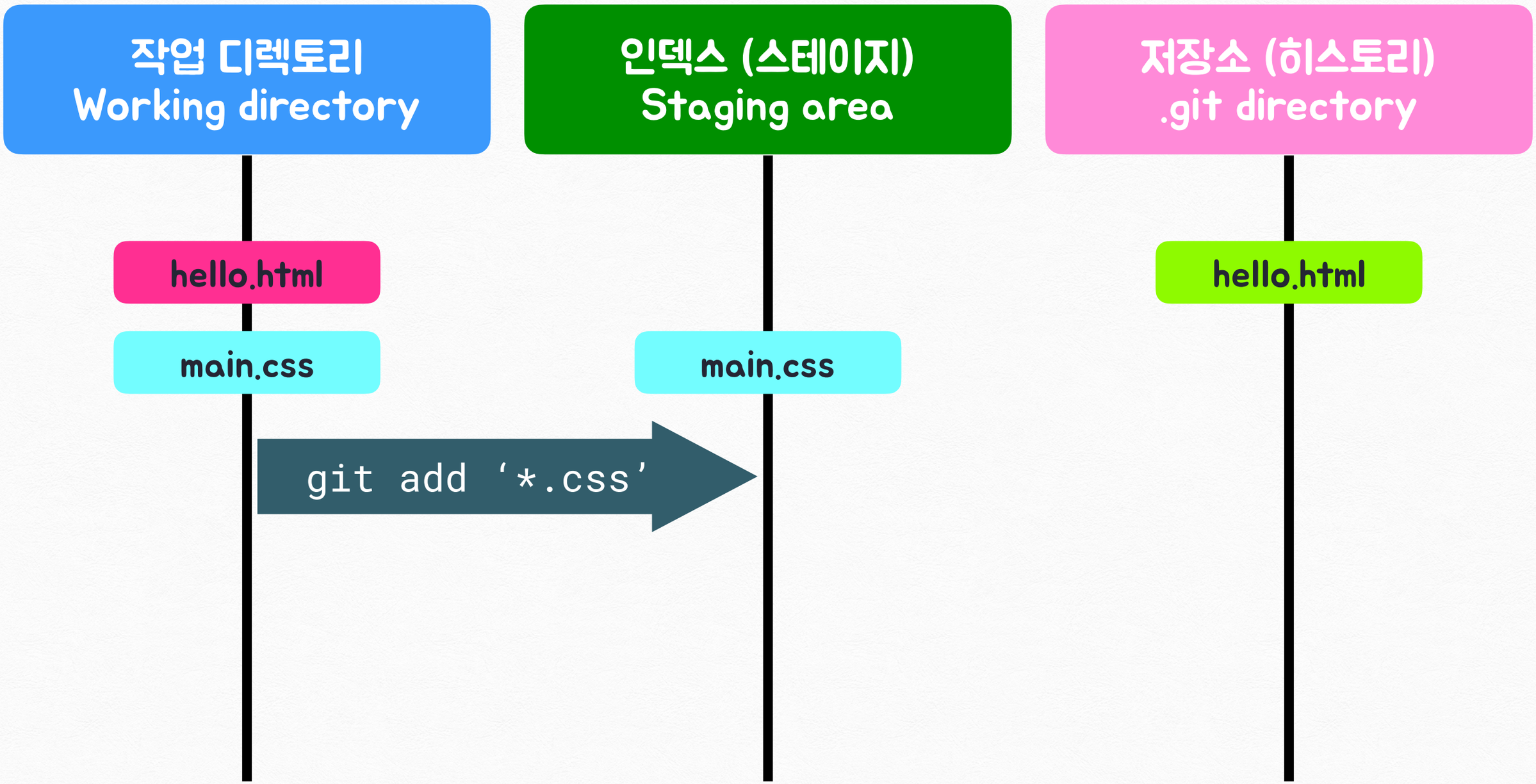
```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   main.css

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
directory)

    modified:   hello.html

$
```

실습 C3 - main.css를 스테이지로

실습 C3 - main.css를 스테이지로



실습 C3 - main.css를 스테이지로



실습 C3 - 이 상황에 커밋하면?

main.css가 저장소에 잘 기록됩니다

hello.html의 변경분이 반영되지 않습니다

실습 C3 - hello.html도 인덱스

실행

```
$ git add hello.html
$
```

스테이지 영역에 hello.html 또 추가

실행

```
$ git status
```

작업디렉토리 상태 또 확인 (2)

실습 C3 - 상태 확인 (3)

결과

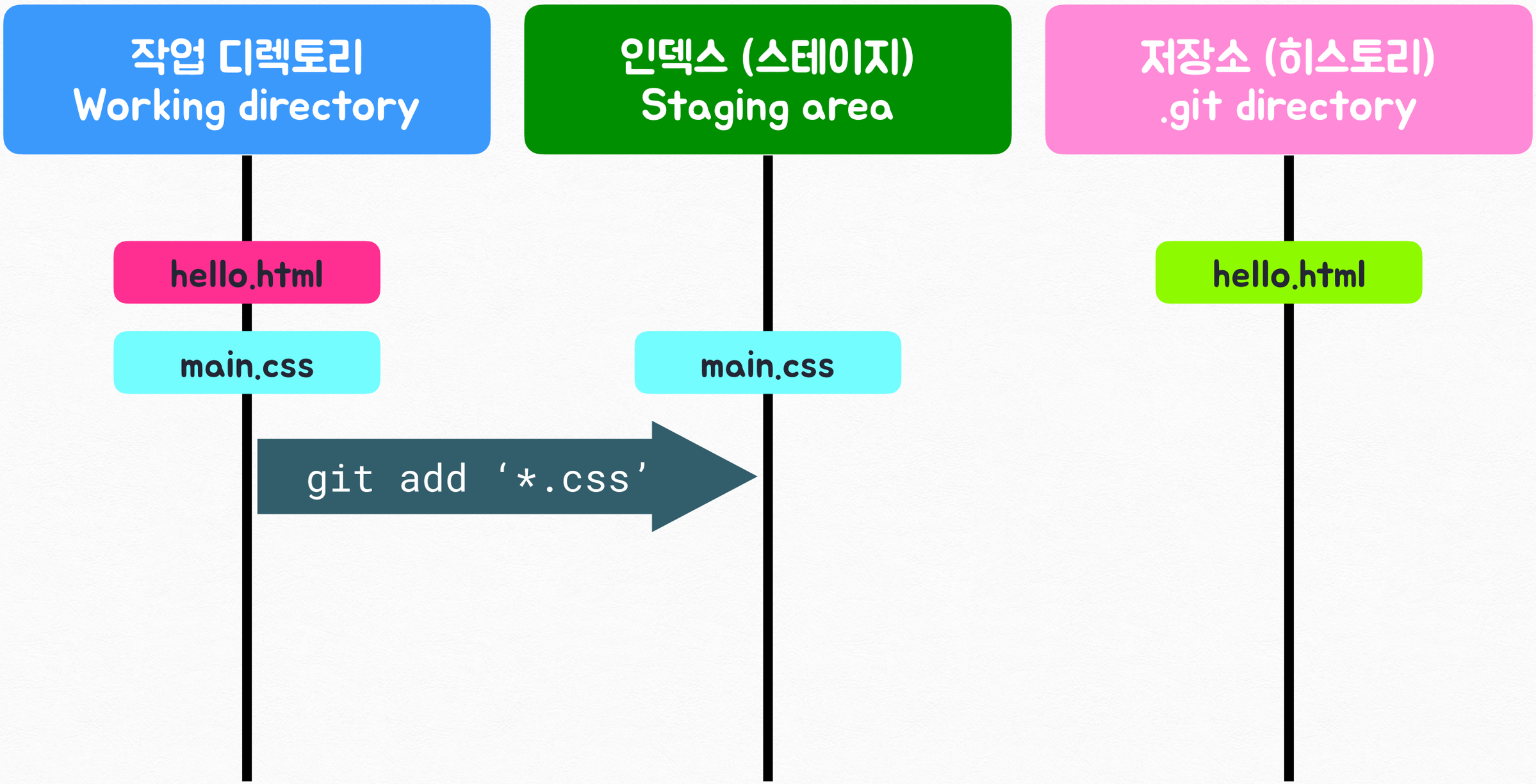
```
$ git status
```

실습 C3 - 상태 확인 (3)

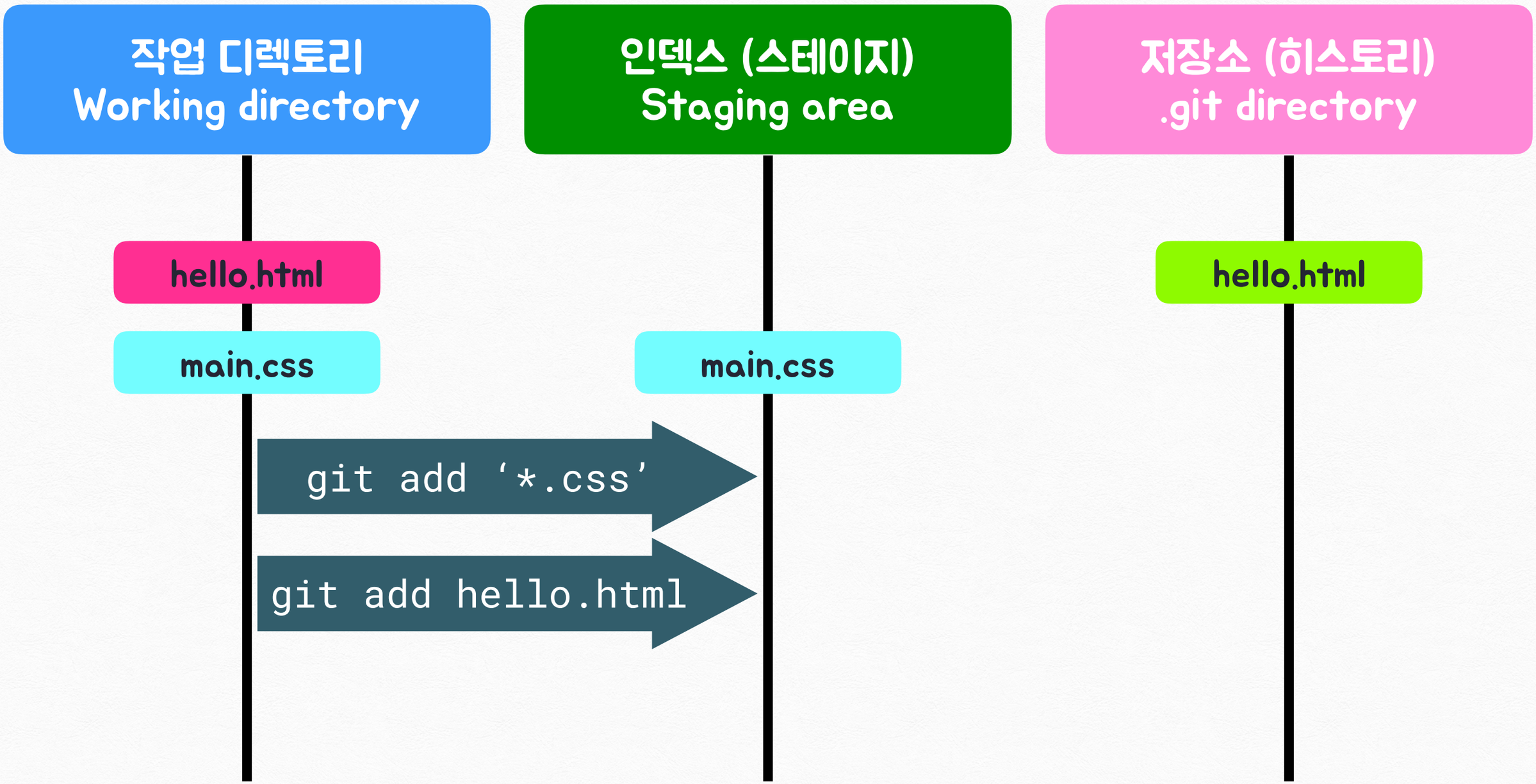
결과

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.html
    new file:   main.css

$
```

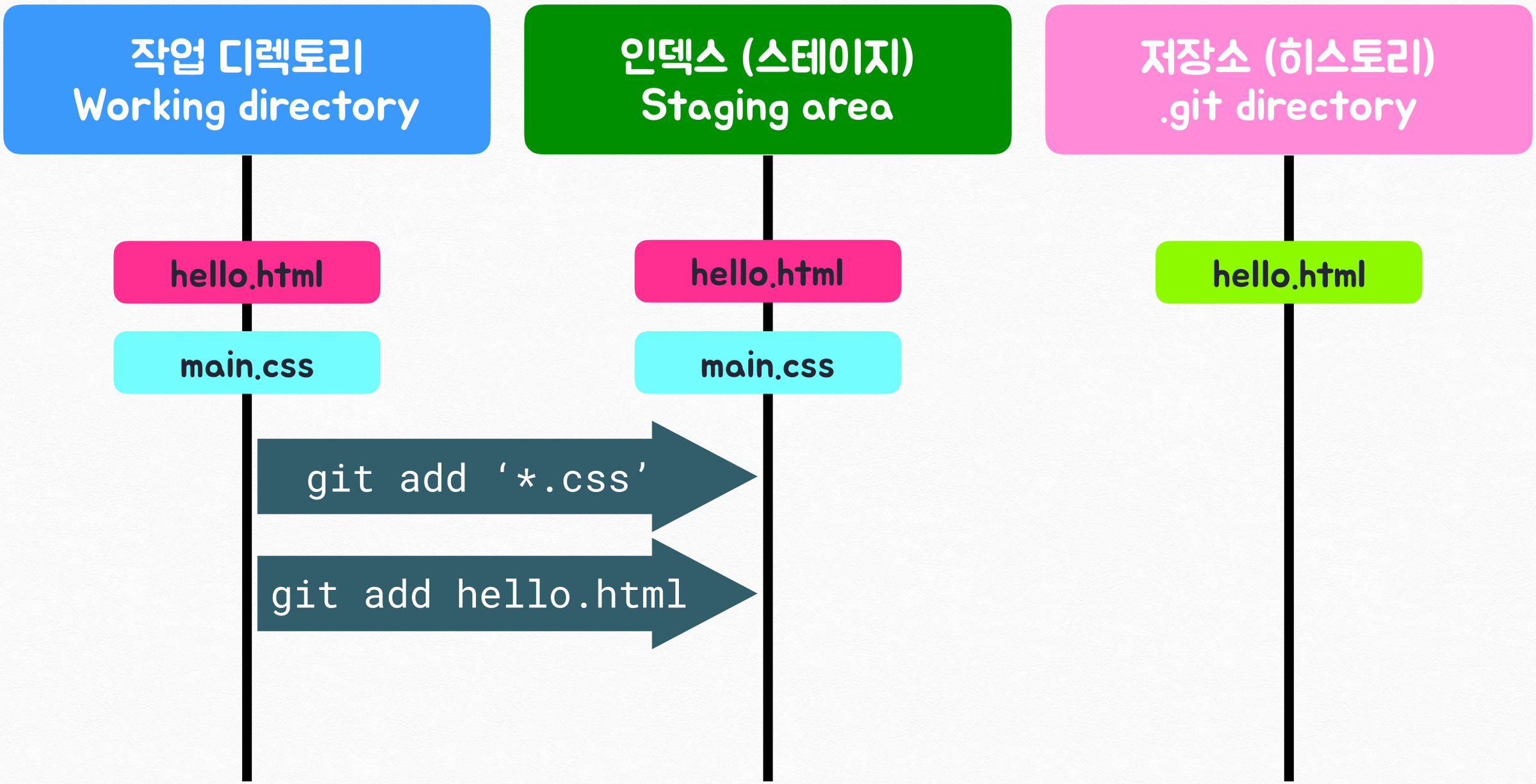
실습 C3 - 커밋 준비 완료


작업 디렉토리
Working directory
인덱스 (스테이지)
Staging area
저장소 (히스토리)
.git directory
hello.html
main.css
main.css
hello.html
git add '*.css'

실습 C3 - 커밋 준비 완료


작업 디렉토리
Working directory
인덱스 (스테이지)
Staging area
저장소 (히스토리)
.git directory
hello.html
main.css
main.css
hello.html
git add '*.css'
git add hello.html

실습 C3 - 커밋 준비 완료


작업 디렉토리
Working directory
인덱스 (스테이지)
Staging area
저장소 (히스토리)
.git directory
hello.html
main.css
hello.html
main.css
hello.html
git add '*.css'
git add hello.html


git

실습 C3 - 두번째 커밋

결과

```
$ git commit -m "add styling"
```

실습 C3 - 두번째 커밋

결과

```
$ git commit -m "add styling"
[master 2942a15] add styling
 2 files changed, 8 insertions(+)
 create mode 100644 main.css
$
```

실습 C3 - 두번째 커밋 후 로그

결과

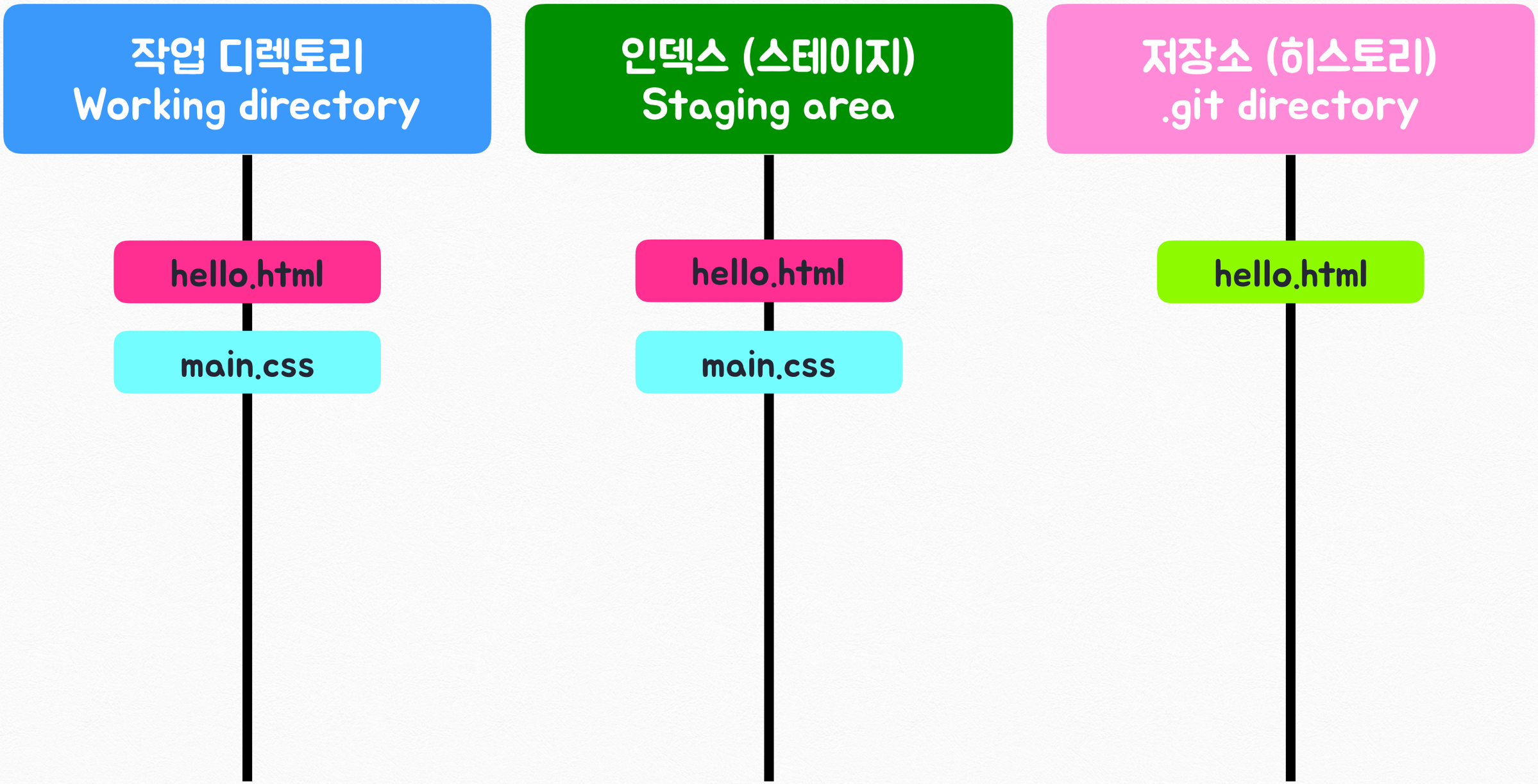
```
$ git log
```

실습 C3 - 두번째 커밋 후 로그

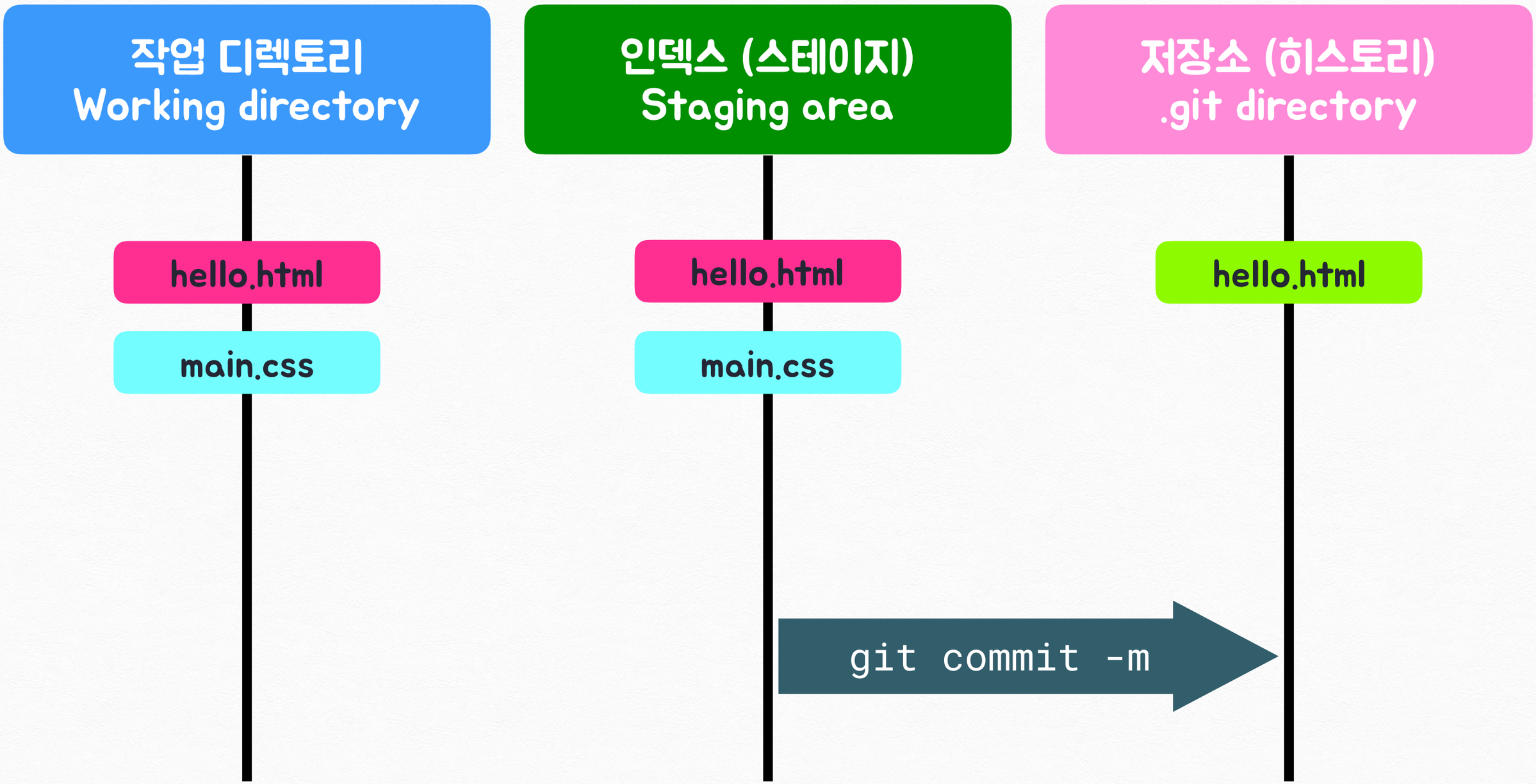
결과

```
$ git log
commit 2942a15c4d64df6a65d7f8d3bcc69561d05a2804 (HEAD -> master)
Author: 홍길동 <hong@gil.dong>
Date:   Sun Dec 17 13:50:55 2017 +0900

    add styling

commit 87bad9b0fa781e5596080a3c7dd83f67dd1a0cad
Author: 홍길동 <hong@gil.dong>
Date:   Sat Dec 16 20:36:34 2017 +0900

    add hello.html
$
```

실습 C3 - 두번 째 커밋


작업 디렉토리
Working directory
인덱스 (스테이지)
Staging area
저장소 (히스토리)
.git directory
hello.html
main.css
hello.html
main.css
hello.html

실습 C3 - 두번 째 커밋


작업 디렉토리
Working directory
인덱스 (스테이지)
Staging area
저장소 (히스토리)
.git directory
hello.html
main.css
hello.html
main.css
hello.html
git commit -m

실습 C3 - 두번 째 커밋


작업 디렉토리
Working directory
인덱스 (스테이지)
Staging area
저장소 (히스토리)
.git directory
hello.html
main.css
hello.html
hello.html
main.css
git commit -m

과제 #1 - sub.css 추가

hello.html ● | main.css | sub.css

```css
main h1 {
  text-decoration: underline;
}
```

1. sub.css 파일 추가
2. hello.html에 sub.css 링크 추가

3. sub.css 인덱스에 추가
4. hello.html도 인덱스에 추가

5. 세번째 커밋!
6. 로그 확인

hello.html | main.css

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Git 실습</title>
    <link rel="stylesheet" type="text/css"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
      integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb"
      crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="main.css">
    <link rel="stylesheet" type="text/css" href="sub.css">
  </head>
  <body>
    <main>
```

과제 #1 결과 로그

결과

```
$ git log
```

과제 #1 결과 로그

결과

```
$ git log
commit d8d202f0d22db5f1b4c53f5793630139c8ef48d5 (HEAD -> master)
Author: 홍길동 <hong@gil.dong>
Date:   Sun Dec 17 14:01:54 2017 +0900

    add sub styling

commit 2942a15c4d64df6a65d7f8d3bcc69561d05a2804
Author: 홍길동 <hong@gil.dong>
Date:   Sun Dec 17 13:50:55 2017 +0900

    add styling

commit 87bad9b0fa781e5596080a3c7dd83f67dd1a0cad
Author: 홍길동 <hong@gil.dong>
Date:   Sat Dec 16 20:36:34 2017 +0900

    add hello.html
$
```

첫 시간 요약

Git 버전 관리 시스템 기초와 유용성

첫 저장소 초기화 & 커밋 세 건

Git의 세 가지 작업 영역 파악

브랜치

대졸 신입 A 이야기 (3)

각 고객사 별로 조금씩 다른 소스코드

한진해운, 대한해운, 흥아해운, 현대상선

브랜치를 썼으면 좋았을 텐데...

다음 시간에 계속...

브랜치 - 친절한 경고!

브랜치는 어렵습니다

익숙해지는데 노력이 듭니다

하지만 완전 매력적인 기능

브랜치 branch

작업 흐름 가지. 커밋 그래프. 커밋 줄기.

각각의 작업 흐름 병행

때때로 병합merge하거나 버리거나 삭제

브랜치branch 장점

상호독립 다수 로컬 브랜치

아주 쉽게 문맥 전환

역할 구분에 활용하기 좋다 (배포/개발/테스트)

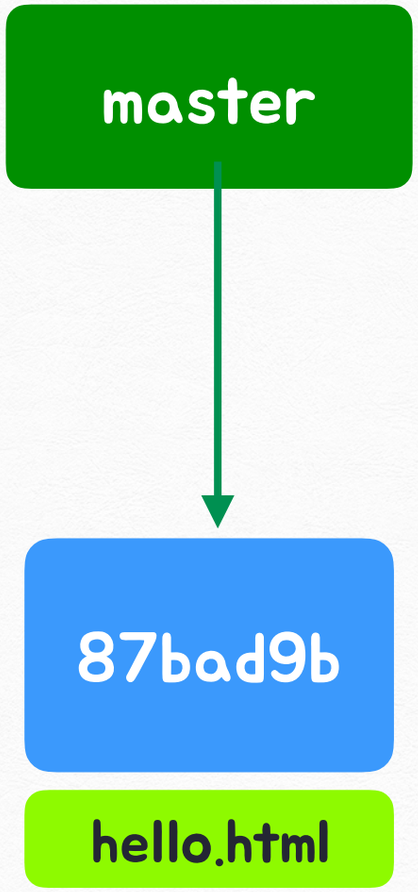
구현하려는 기능 단위 브랜치도 Good!

master 브랜치

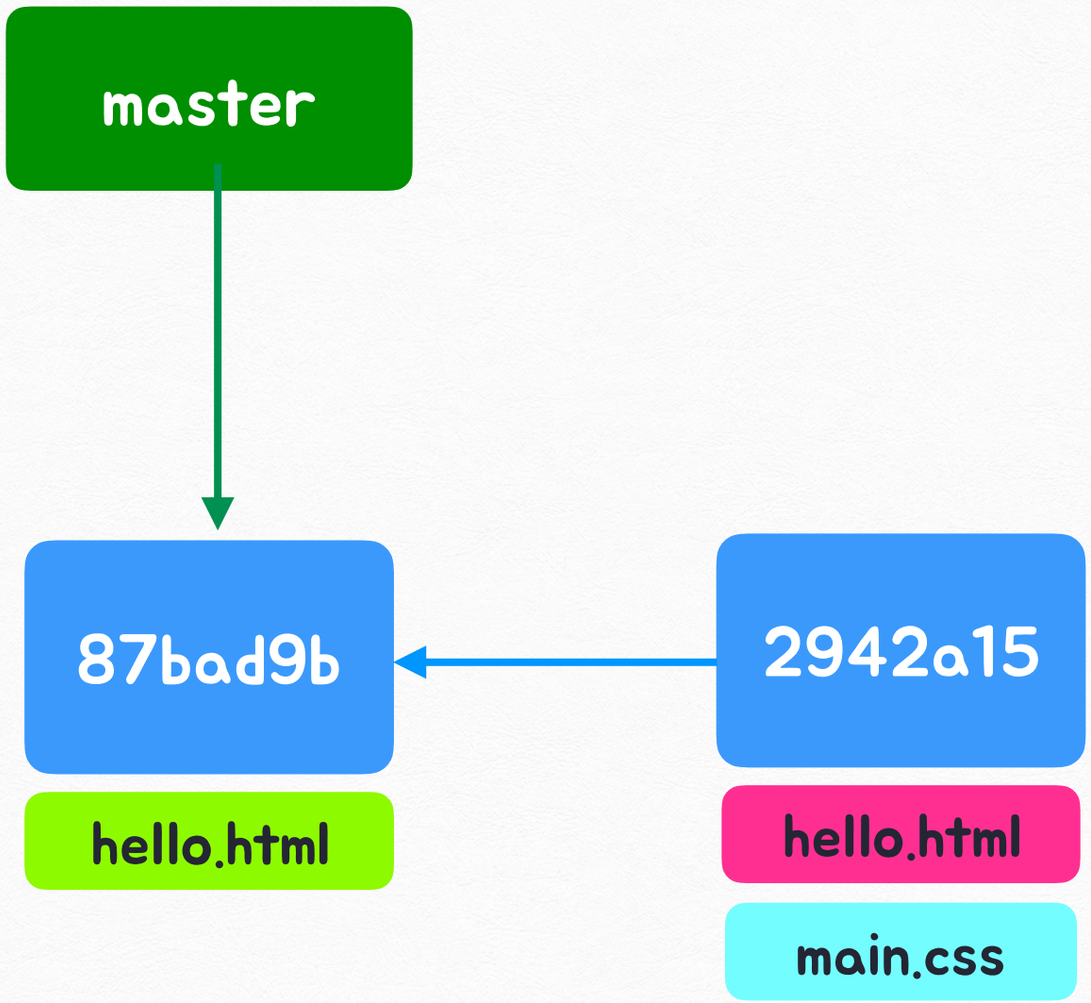
지금까지 작업한 단일 브랜치

주main 작업 영역

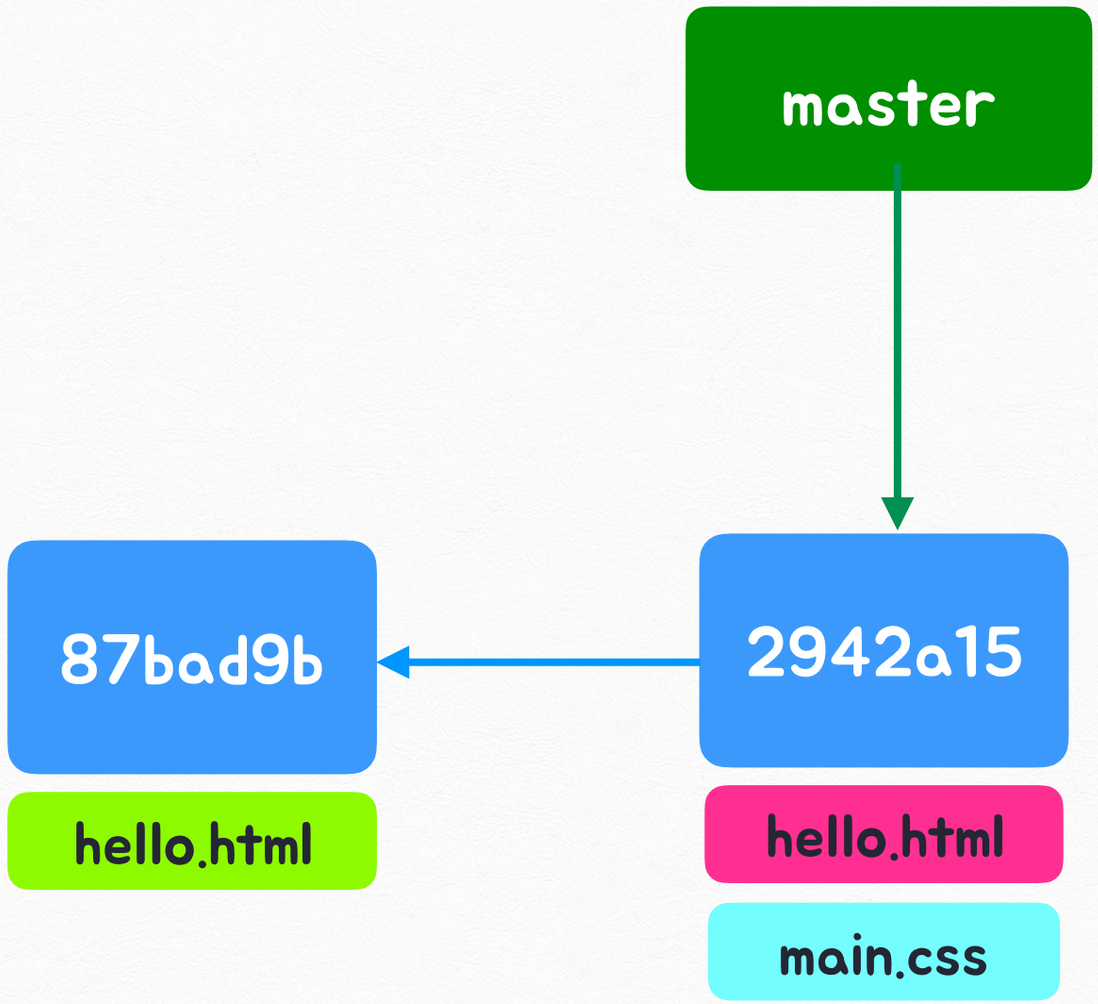
지금껏 master 브랜치 상황



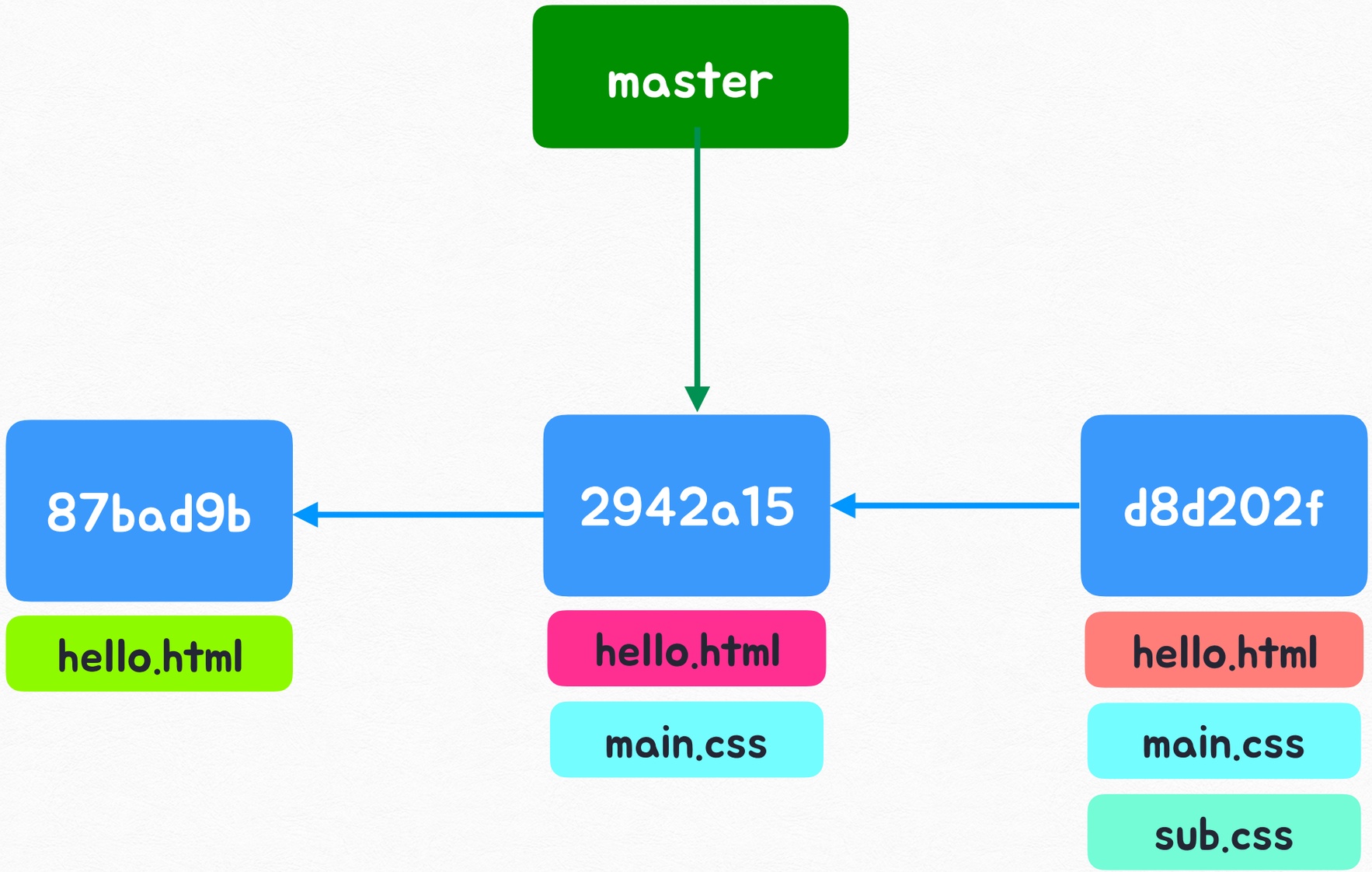
지금껏 master 브랜치 상황



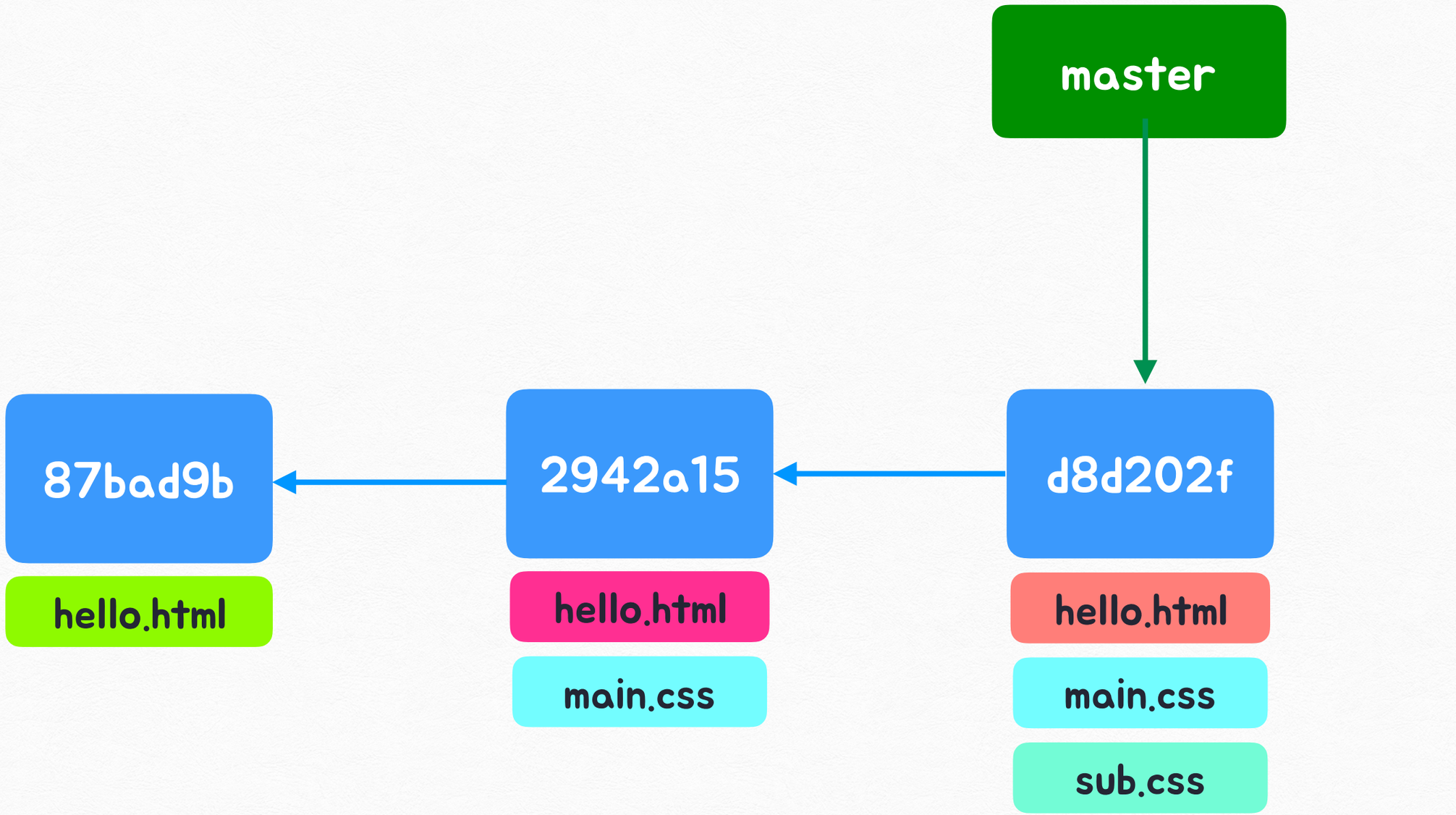
지금껏 master 브랜치 상황



지금껏 master 브랜치 상황



지금껏 master 브랜치 상황



브랜치

어렵지만 매우 유용

팀 개발 작업에 필수

실습 D1 - 현재 브랜치 확인

결과

```
$ git branch
```

실습 D1 - 현재 브랜치 확인

결과

```
$ git branch
* master
$
```

실습 D1 - 현재 브랜치 확인

결과

```
$ git branch
* master
$ git branch -v
* master d8d202f add sub styling
$
```

실습 D2 - 새 브랜치 만들기

결과

```
$ git branch testing
```

실습 D2 - 새 브랜치 만들기

결과

```
$ git branch testing

$
```

실습 D2 - 새 브랜치 만들기

결과

```
$ git branch testing

$ git branch
```

실습 D2 - 새 브랜치 만들기

결과

```
$ git branch testing

$ git branch
* master
  testing
$
```

실습 D2 - 새 브랜치 만들기

결과

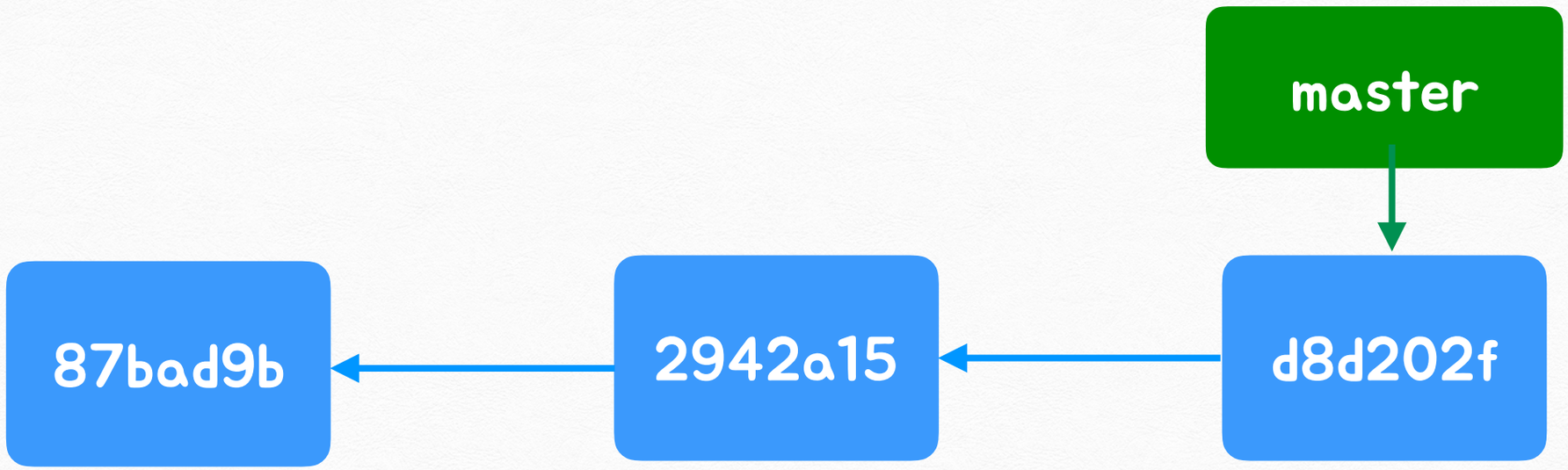
```
$ git branch testing

$ git branch
* master
  testing
$ git branch -v
```

실습 D2 - 새 브랜치 만들기

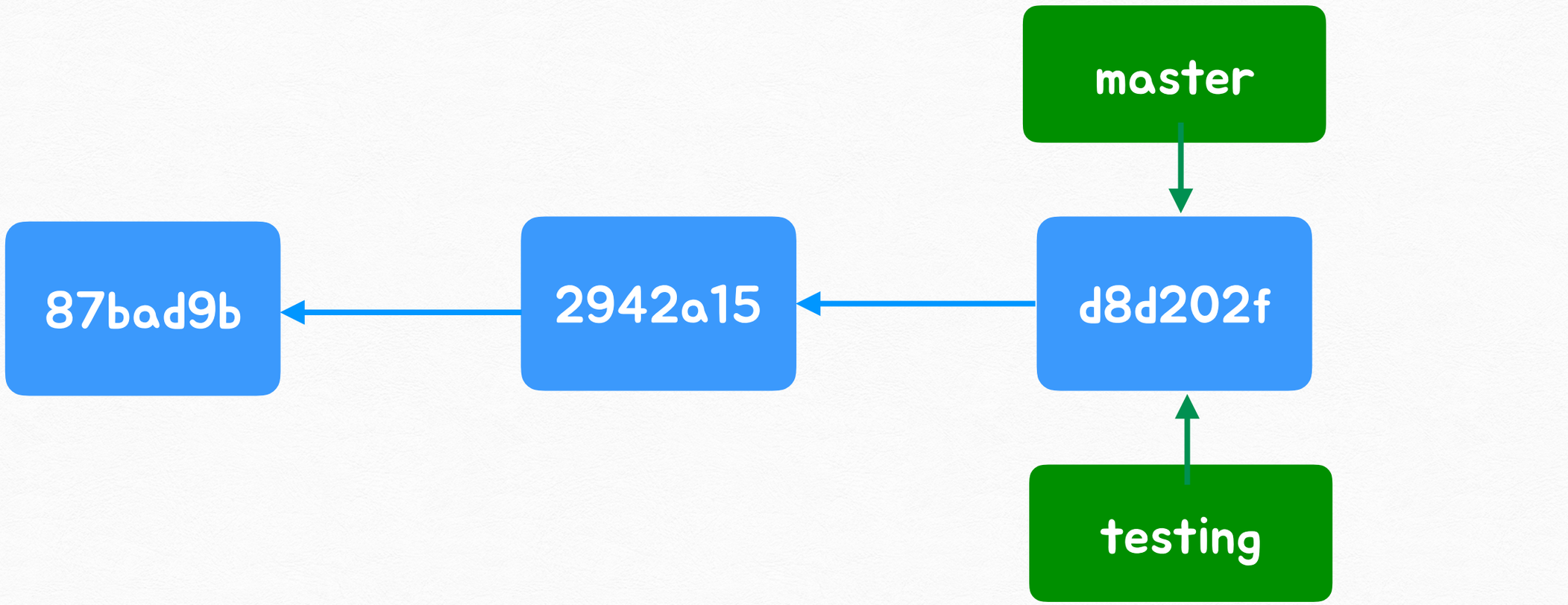
결과

```
$ git branch testing

$ git branch
* master
  testing
$ git branch -v
* master  d8d202f add sub styling
  testing d8d202f add sub styling
$
```

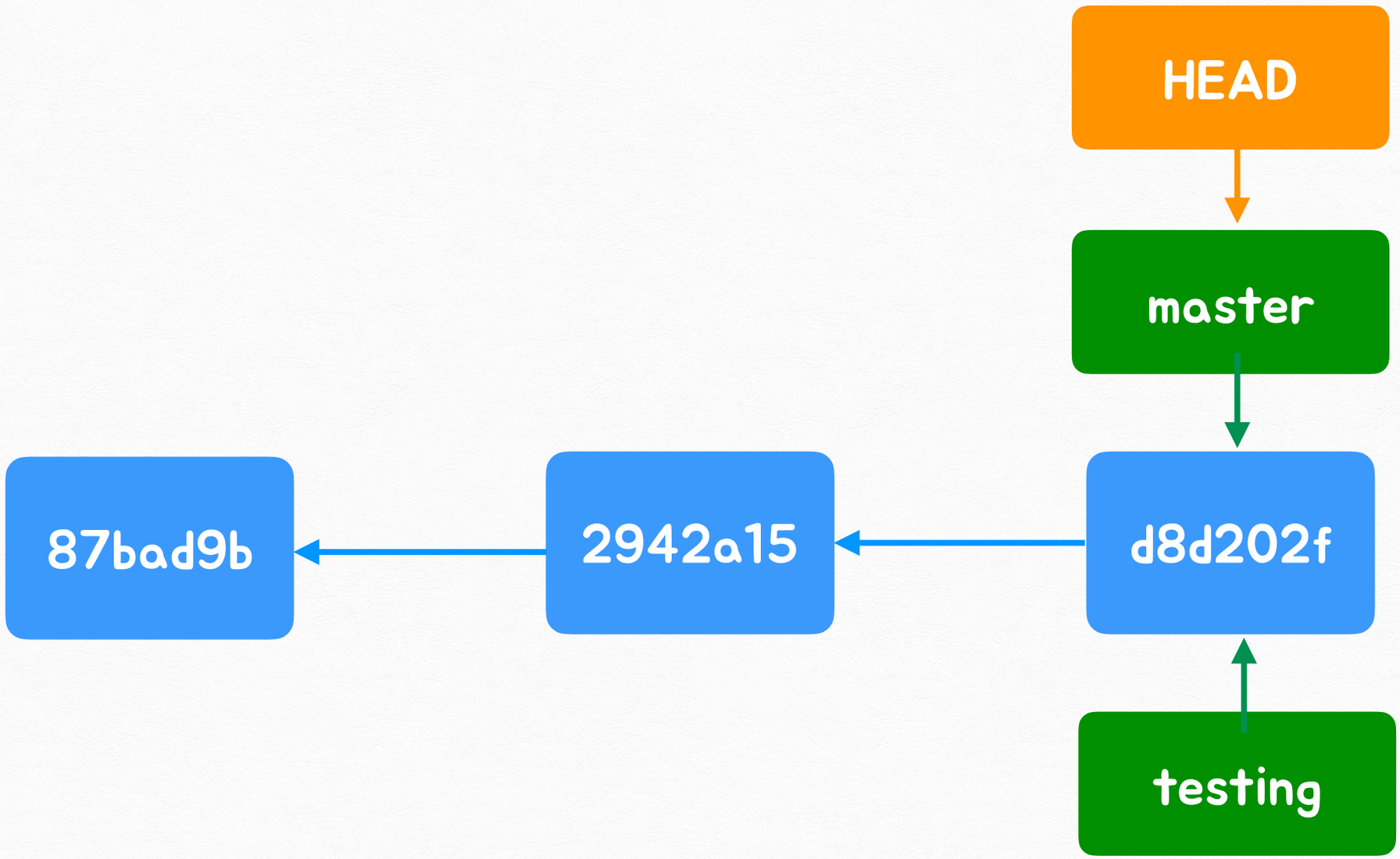
master/testing 브랜치 상황 (1)



master/testing 브랜치 상황 (1)



master/testing 브랜치 상황 (1)



HEAD

각 브랜치는 특정 (최종) 커밋을 가리킴

HEAD는 현재 브랜치를 가리킴

결국, HEAD는 현재 브랜치의 최종커밋을 가리킴

실습 D2 - 새 브랜치로 전환

결과

```
$ git checkout testing
```

실습 D2 - 새 브랜치로 전환

결과

```
$ git checkout testing
Switched to branch 'testing'

$ git branch -v
```

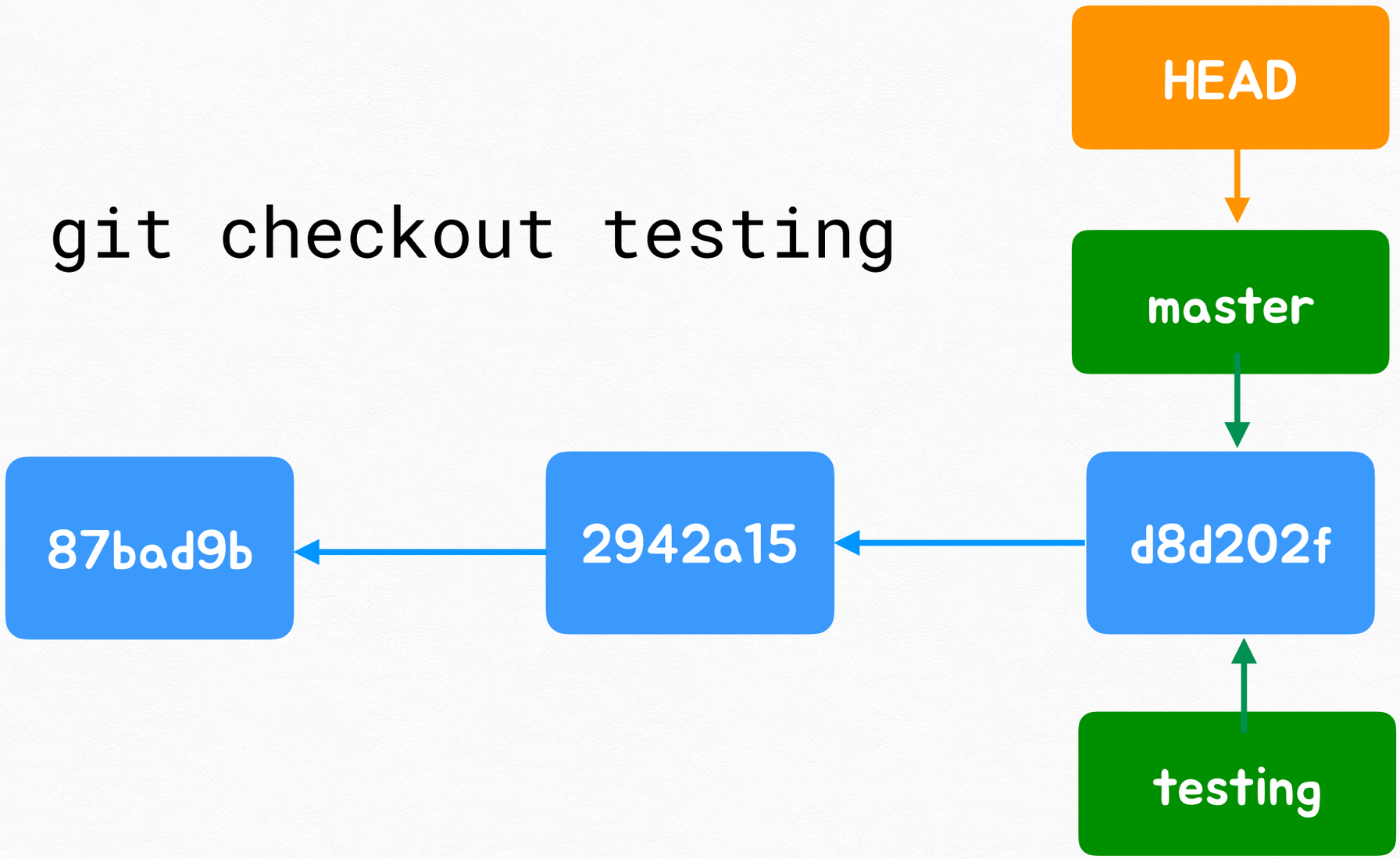
실습 D2 - 새 브랜치로 전환

결과

```
$ git checkout testing
Switched to branch 'testing'

$ git branch -v
  master  d8d202f add sub styling
* testing d8d202f add sub styling

$
```

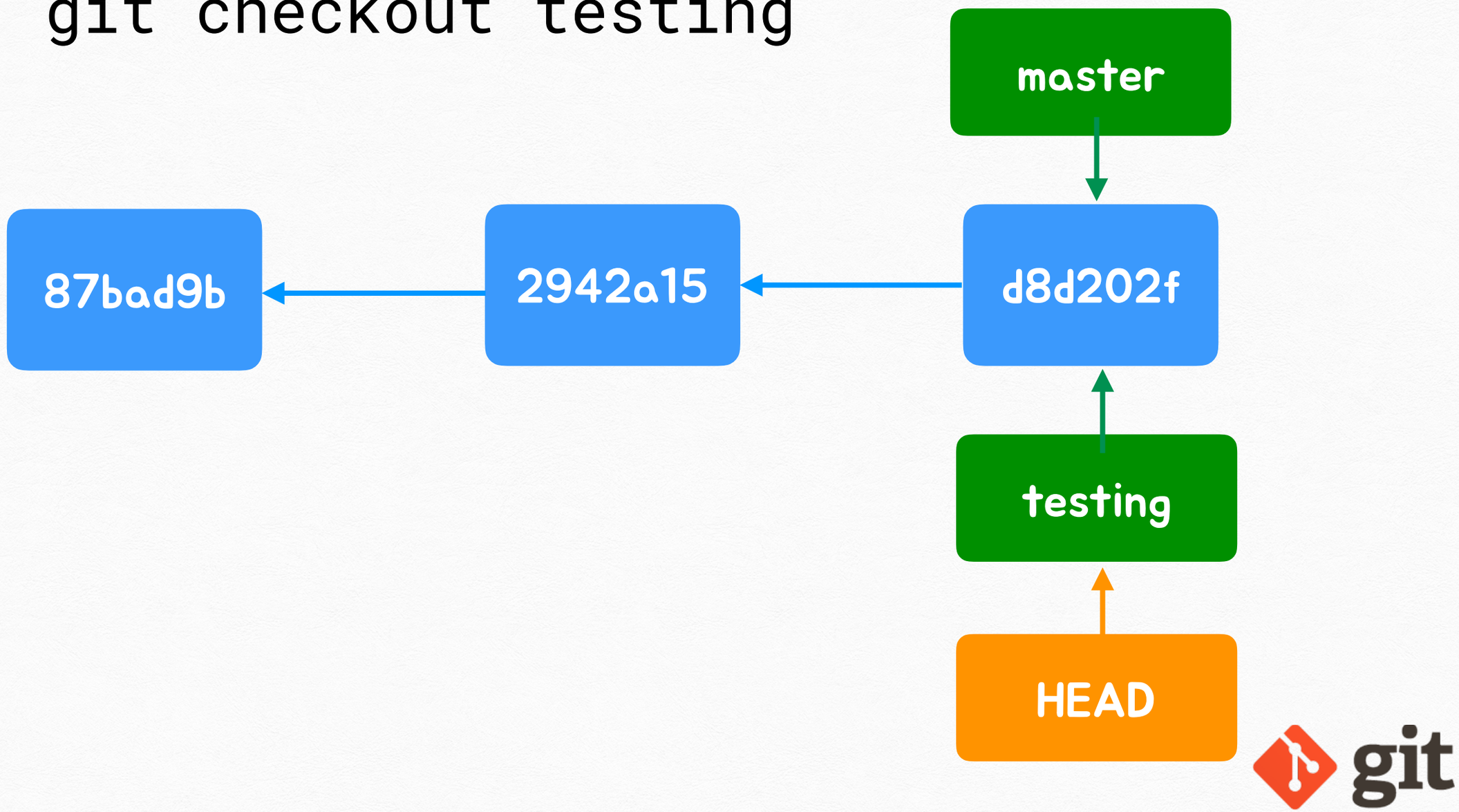
master/testing 브랜치 상황 (2)

```
git checkout testing
```



master/testing 브랜치 상황 (2)

```
git checkout testing
```



실습 D3 - testing 브랜치에 추가작업

test.js

```
1   console.log("test.js 로드")
2   alert("테스트")
3
```

test.js | hello.html | main.css

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="utf-8">
5       <title>Git 실습</title>
6       <link rel="stylesheet" type="text/css"
7         href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
8         integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb"
9         crossorigin="anonymous">
10      <link rel="stylesheet" type="text/css" href="main.css">
11      <link rel="stylesheet" type="text/css" href="sub.css">
12    </head>
13    <body>
14      <main>
15      <h1>안녕하세요</h1>
16      </main>
17      <script src="test.js"></script>
18    </body>
19  </html>
20
```

git

실습 D3 - testing 브랜치 상황

결과

```
$ git status
```

실습 D3 - testing 브랜치 상황

결과

```
$ git status
On branch testing
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
directory)

    modified:   hello.html

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

    test.js

no changes added to commit (use "git add" and/or "git commit -a”)
$
```

실습 D3 - testing 브랜치에 커밋

결과

```
$ git add test.js
```

실습 D3 - testing 브랜치에 커밋

결과

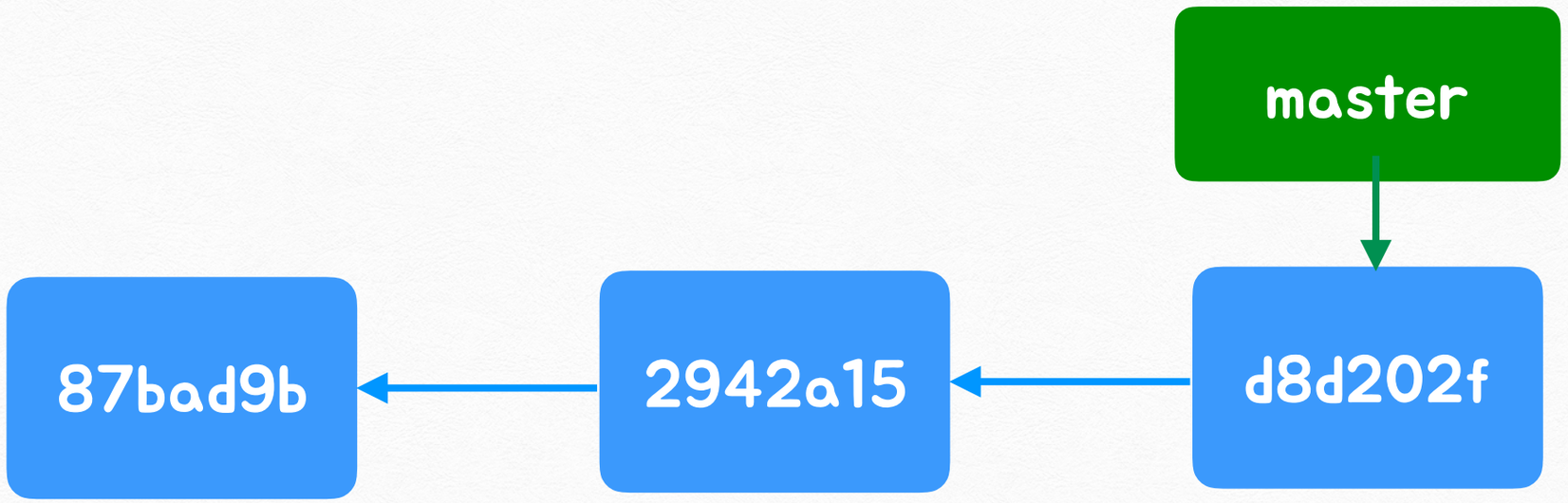
```
$ git add test.js
$ git commit -a -m "add test.js"
```

실습 D3 - testing 브랜치에 커밋

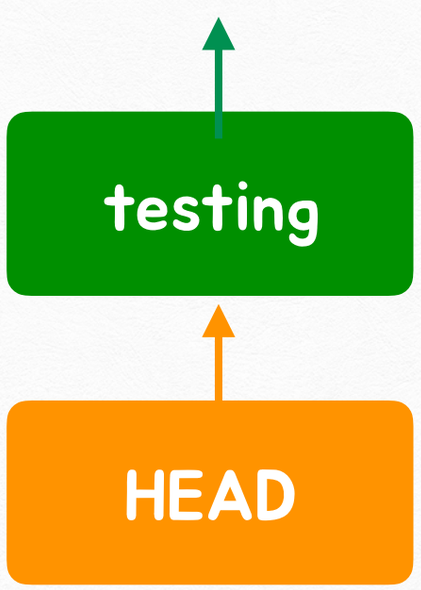
결과

```
$ git add test.js
$ git commit -a -m "add test.js"
[testing a5f5bef] add test.js
 2 files changed, 3 insertions(+)
 create mode 100644 test.js
$
```

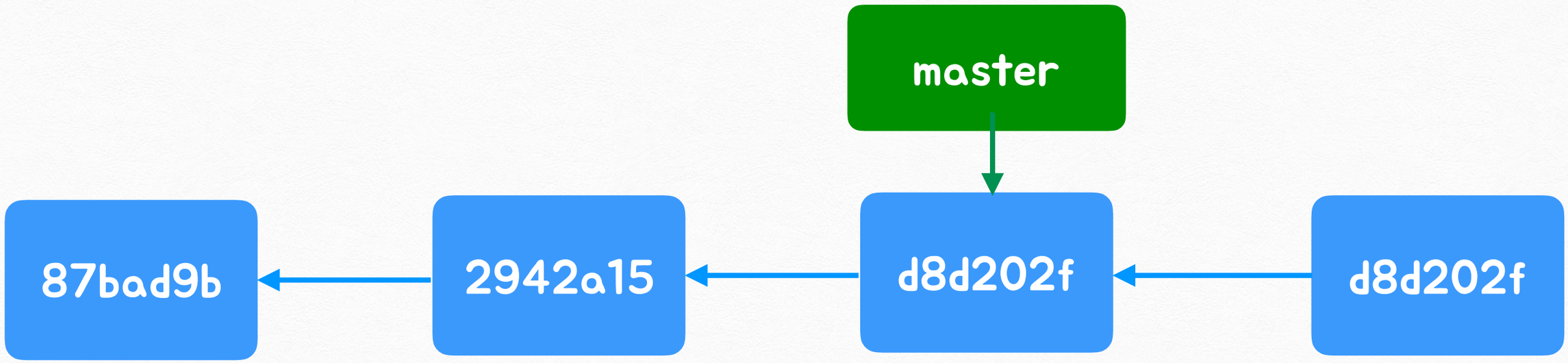
master/testing 브랜치 상황 (3)

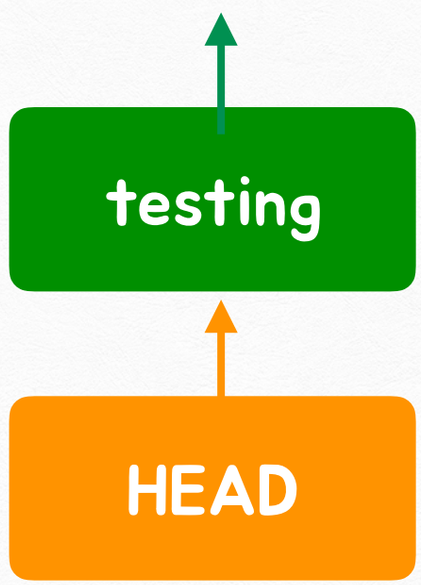



```
git commit
```



master/testing 브랜치 상황 (3)






```
git commit
```

실습 D4 - master 브랜치로 복귀

결과

```
$ git branch -v
```

실습 D4 - master 브랜치로 복귀

결과

```
$ git branch -v
  master  d8d202f add sub styling
* testing a5f5bef add test.js
$ git checkout master
```

실습 D4 - master 브랜치로 복귀

결과

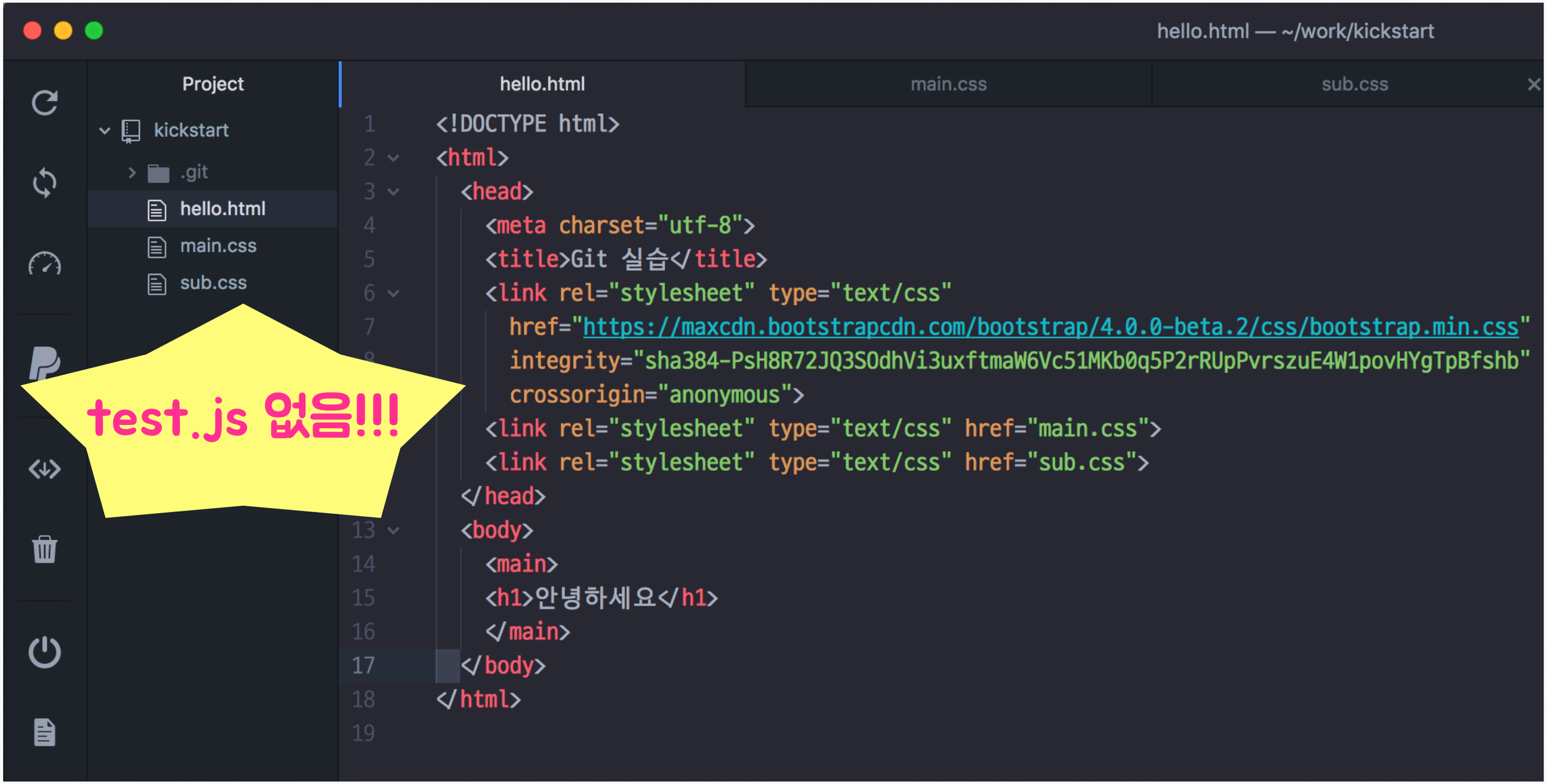
```
$ git branch -v
  master  d8d202f add sub styling
* testing a5f5bef add test.js
$ git checkout master
Switched to branch 'master'
$ git branch -v
```

실습 D4 - master 브랜치로 복귀

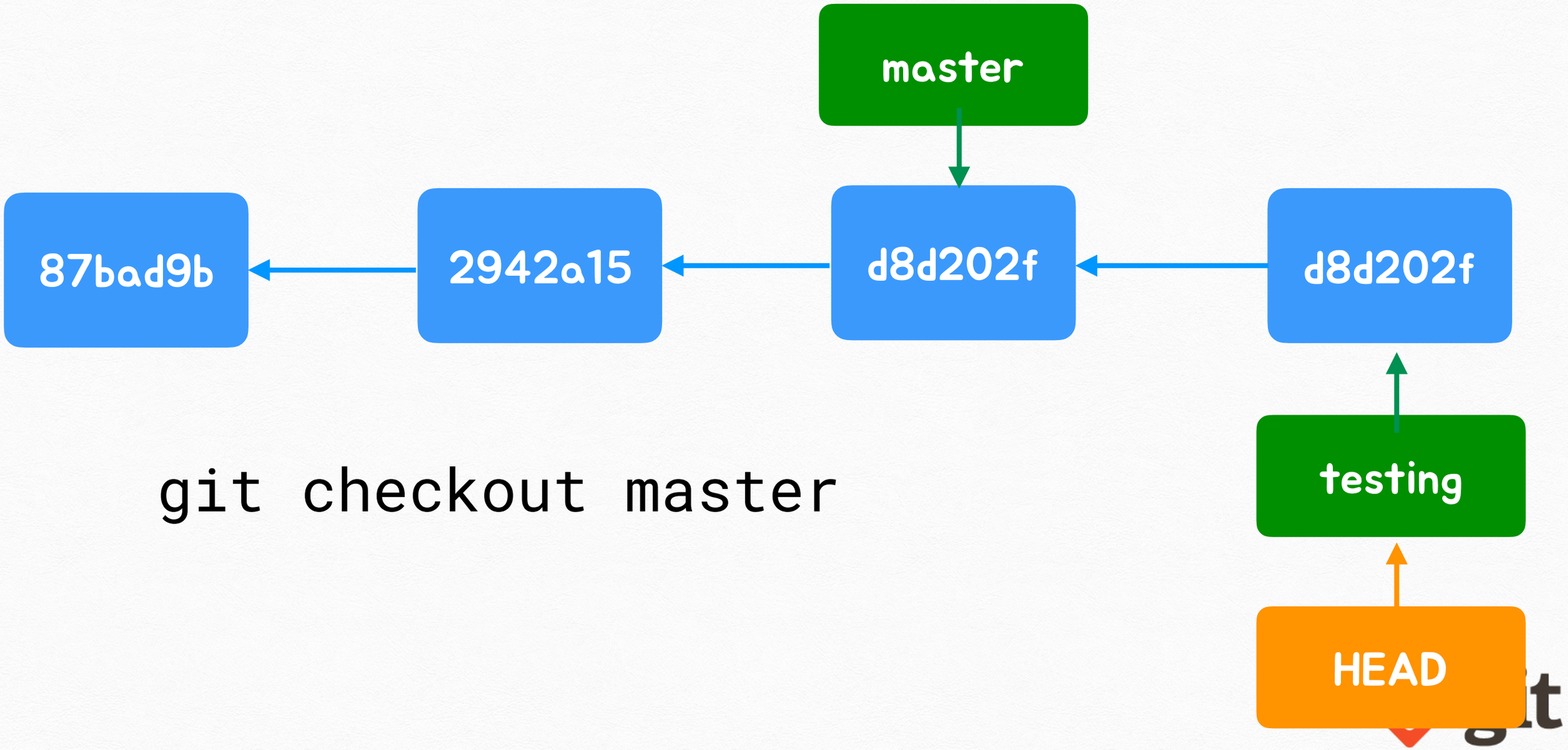
결과

```
$ git branch -v
  master  d8d202f add sub styling
* testing a5f5bef add test.js
$ git checkout master
Switched to branch 'master'
$ git branch -v
* master  d8d202f add sub styling
  testing a5f5bef add test.js
$
```

실습 D4 - master 브랜치 복귀 후

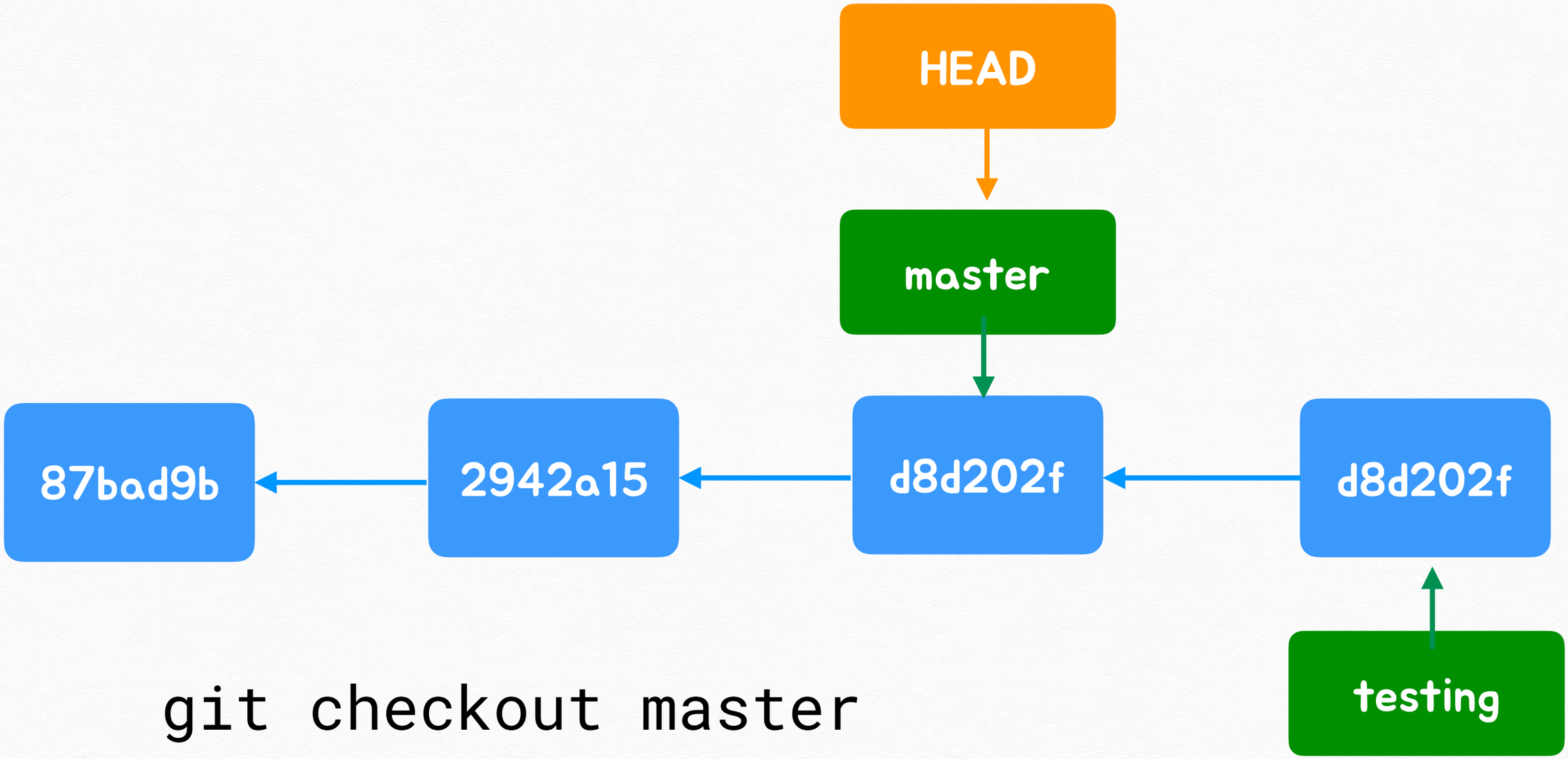


master/testing 브랜치 상황 (4)




```
git checkout master
```

master/testing 브랜치 상황 (4)




```
git checkout master
```

실습 D4 - 브랜치 변경 유의사항

브랜치를 전환하면 작업 디렉토리 내용도 함께 바뀜

실습 D5 - 테스트 성공, 마스터 반영

testing 브랜치 작업이 성공적

master에도 변경 내역을 반영합시다

master와 testing를 합치기merge

실습 D5 - merge

결과

```
$ git merge testing
```

실습 D5 - merge

결과

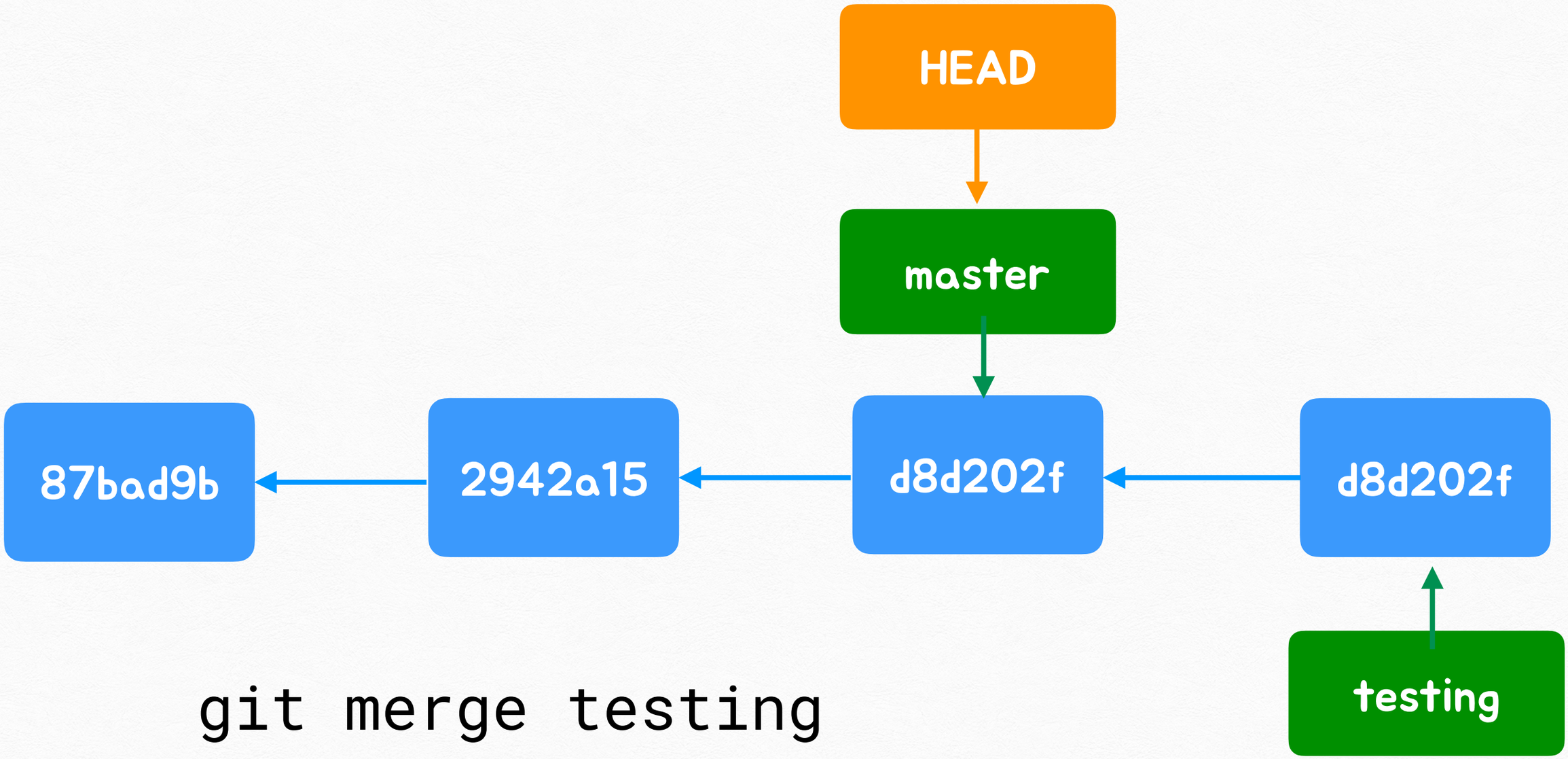
```
$ git merge testing
Updating d8d202f..a5f5bef
Fast-forward
 hello.html | 1 +
 test.js    | 2 ++
 2 files changed, 3 insertions(+)
 create mode 100644 test.js

$ git branch -v
```

실습 D5 - merge

결과

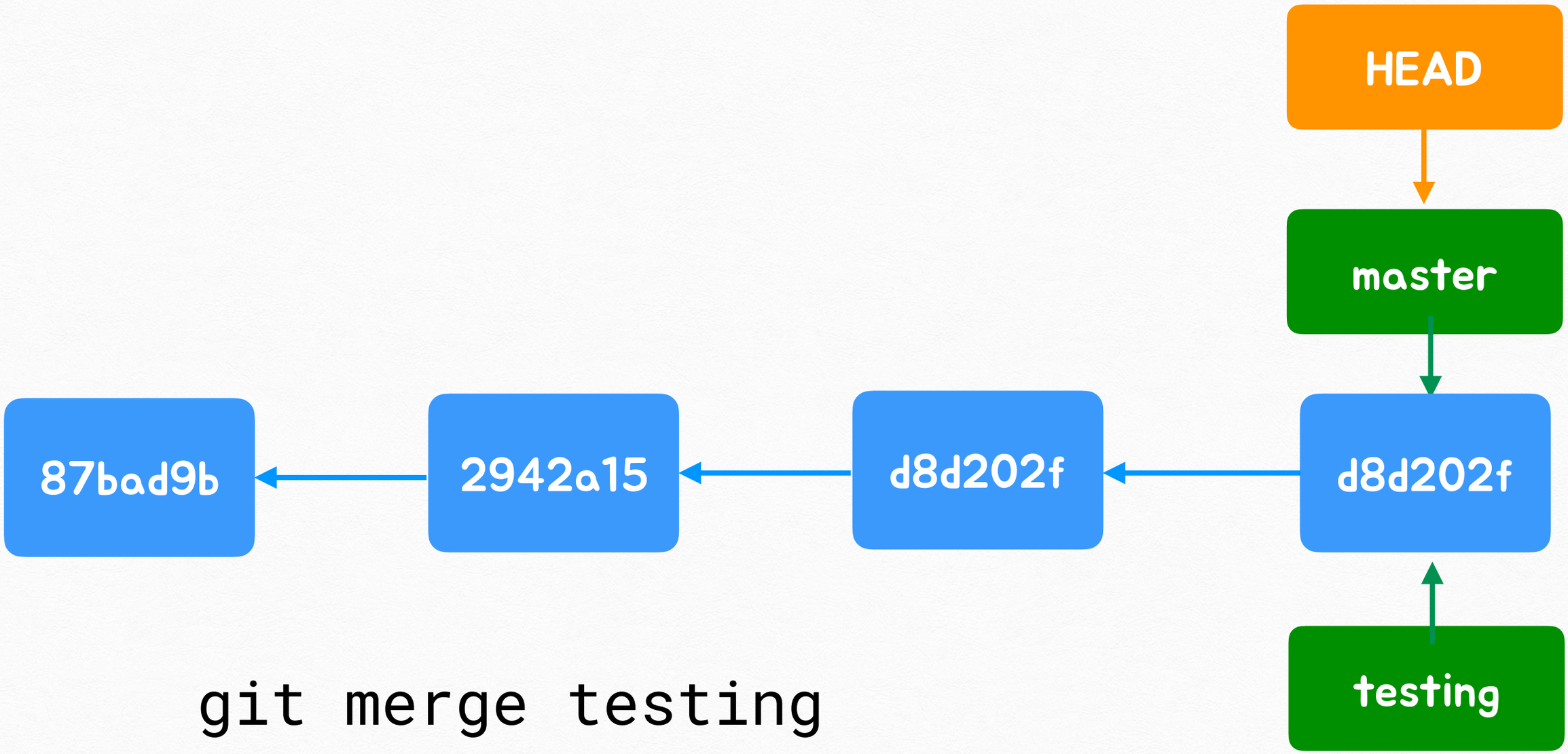
```
$ git merge testing
Updating d8d202f..a5f5bef
Fast-forward
 hello.html | 1 +
 test.js    | 2 ++
 2 files changed, 3 insertions(+)
 create mode 100644 test.js

$ git branch -v
* master  a5f5bef add test.js
  testing a5f5bef add test.js
$
```

master/testing 브랜치 상황 (5)




```
git merge testing
```

master/testing 브랜치 상황 (5)




```
git merge testing
```

실습 D6 - 앗차! 실수! 취소!

아직 master 브랜치에 testing 브랜치 합치면 안됨

뒤늦게 실수임을 인지

간단하게 되돌립니다

실습 D5 - reset --hard

결과

```
$ git branch -v
* master  a5f5bef add test.js
  testing a5f5bef add test.js
$ git reset --hard d8d202f
```

git reset --hard 커밋ID

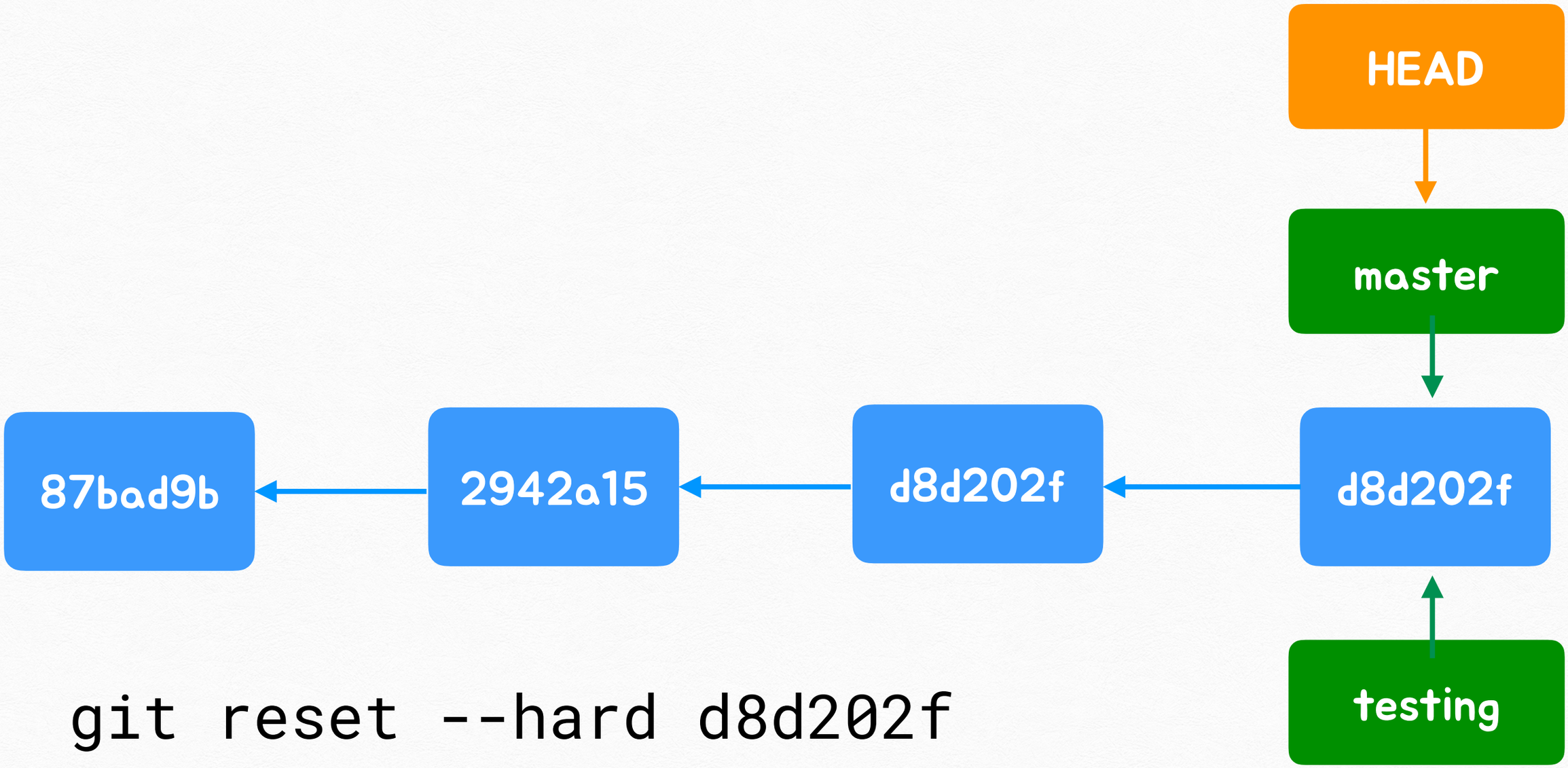
실습 D5 - reset --hard

결과

```
$ git branch -v
* master  a5f5bef add test.js
  testing a5f5bef add test.js
$ git reset --hard d8d202f
HEAD is now at d8d202f add sub styling
$
```

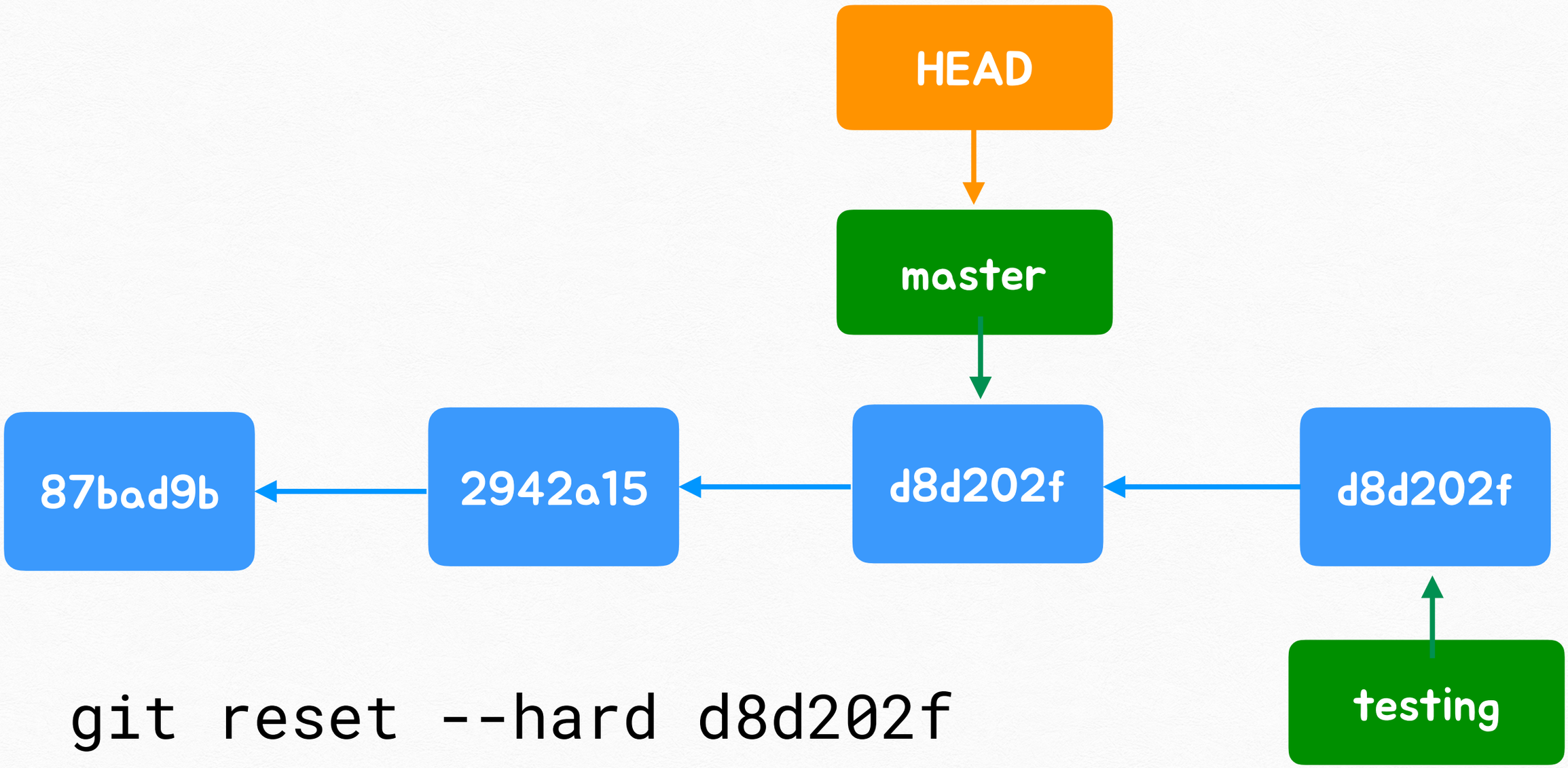
git reset --hard 커밋ID

master/testing 브랜치 상황 (6)




```
git reset --hard d8d202f
```

master/testing 브랜치 상황 (6)




```
git reset --hard d8d202f
```

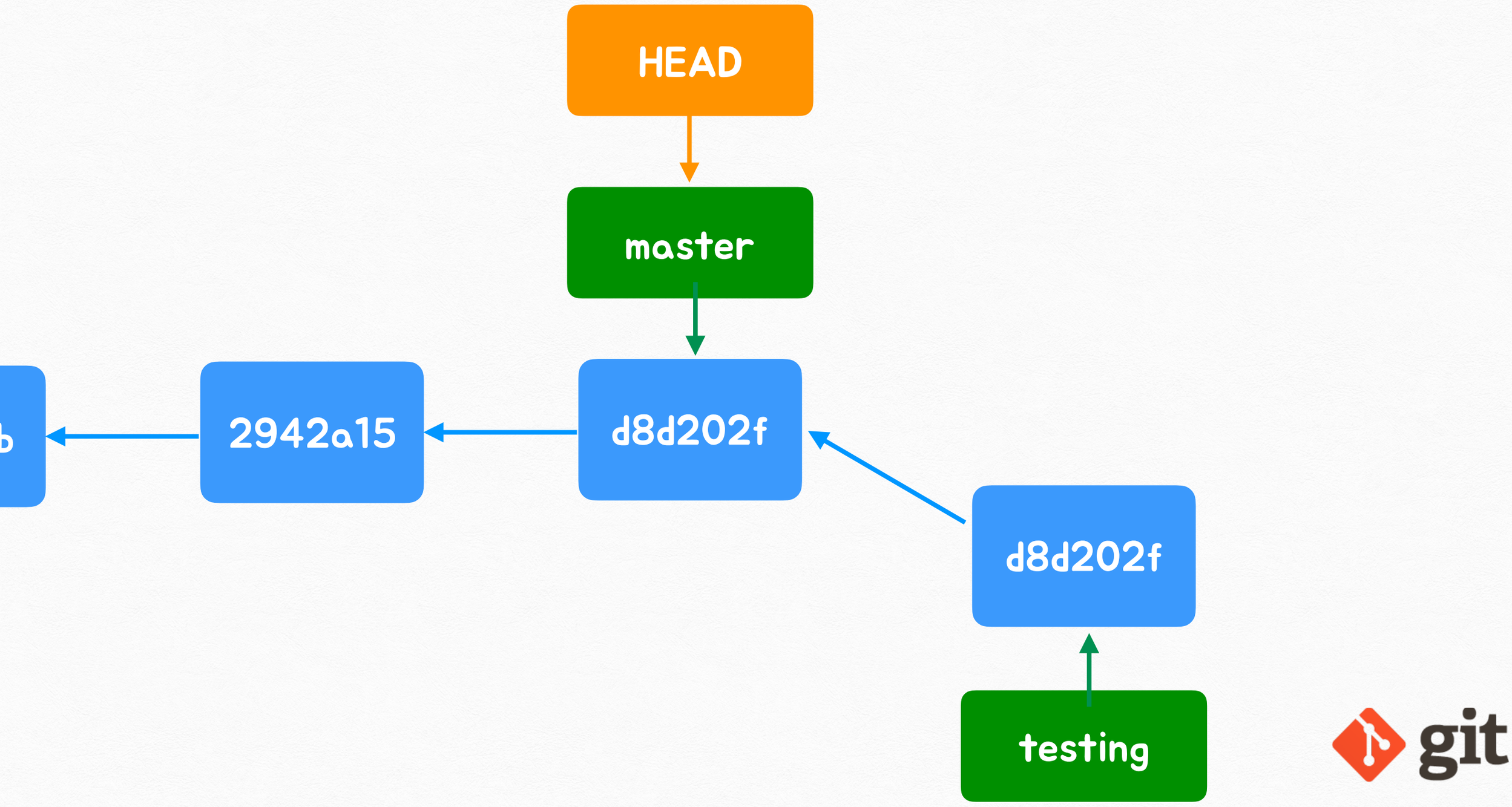
과제 #2 - master브랜치에 커밋후 머지

hello.html | main.css | sub.css

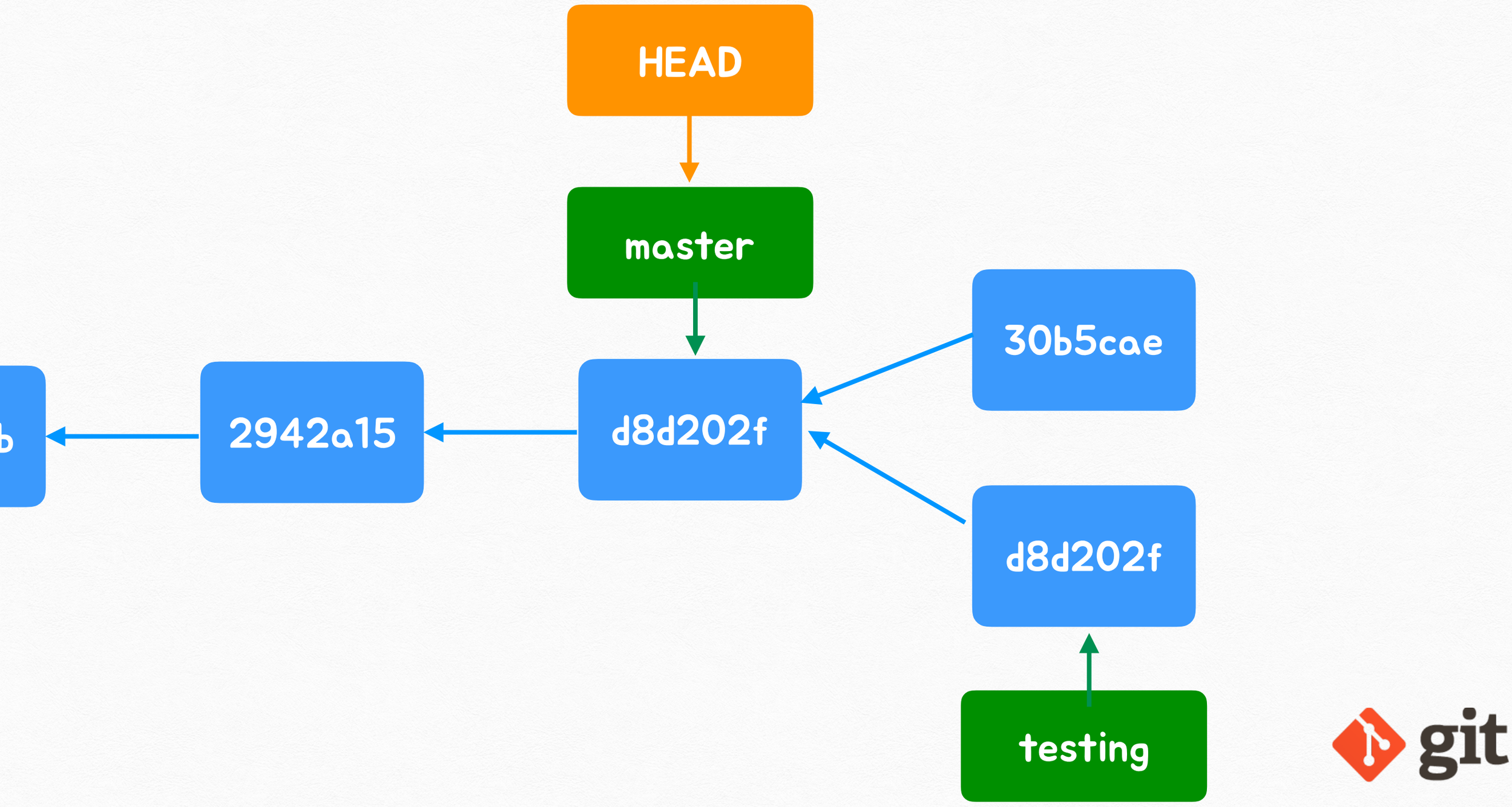
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Git 실습</title>
    <link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
      integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb"
      crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="main.css">
    <link rel="stylesheet" type="text/css" href="sub.css">
  </head>
  <body>
    <main>
    <h1>안녕하세요</h1>
    <div>대학생을 위한 Git 실습</div>
    </main>
  </body>
</html>

```

1. hello.html에 아래 한 줄 추가 git add
2. 새로 커밋 추가 git commit -m " "
3. testing브랜치 합치기 git merge testing
4. 종이에 현재 상황 그림 그리기

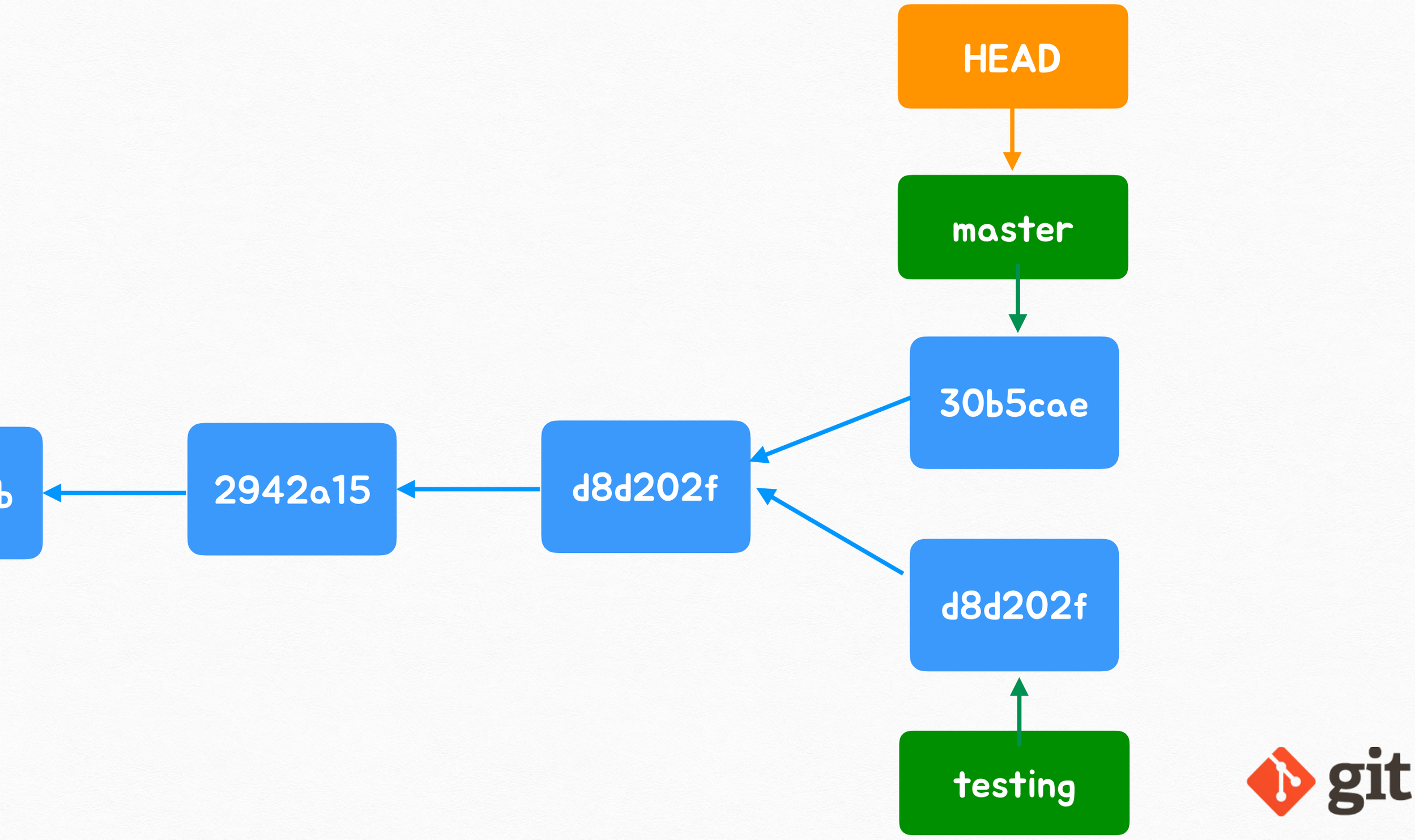
master/testing 브랜치 상황 (7)



master/testing 브랜치 상황 (7)



master/testing 브랜치 상황 (7)

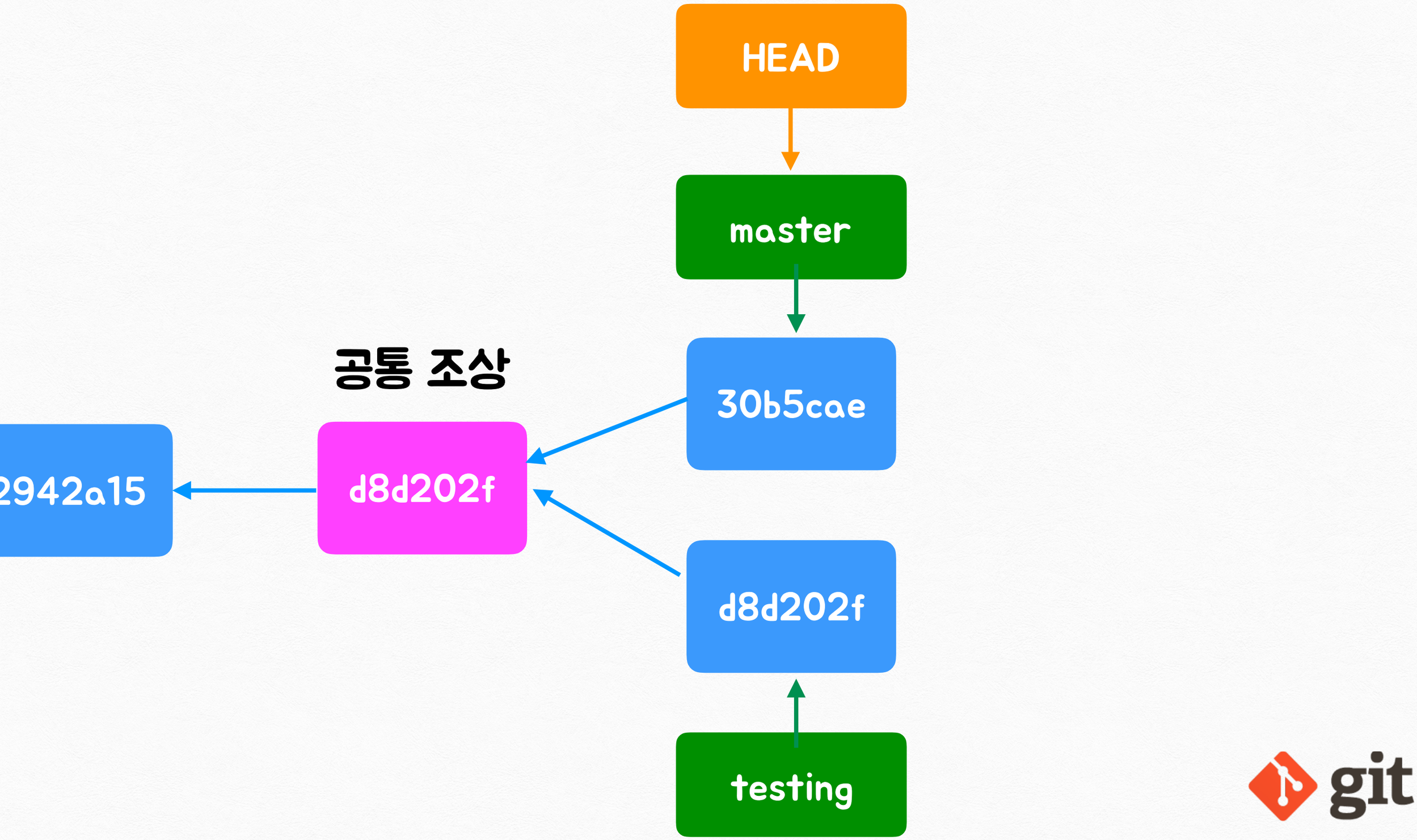


실습 D6 - 나뉜 브랜치를 합치기

master 브랜치와 testing브랜치가 나뉘어 각자의 길을 간 상황

git이 합치려는 브랜치들의 공통 조상을 찾아서 처리

master/testing 브랜치 상황 (8)



실습 D6 - merge (2)

결과

```
$ git branch -v
* master  30b5cae add a div
  testing a5f5bef add test.js

$ git merge testing
```

실습 D6 - merge (2)

결과

```
$ git branch -v
* master  30b5cae add a div
  testing a5f5bef add test.js

$ git merge testing
Auto-merging hello.html
Merge made by the 'recursive' strategy.
 hello.html | 1 +
 test.js    | 2 ++
 2 files changed, 3 insertions(+)
 create mode 100644 test.js

$ git branch -v
```

merge커밋 메시지를 요구!

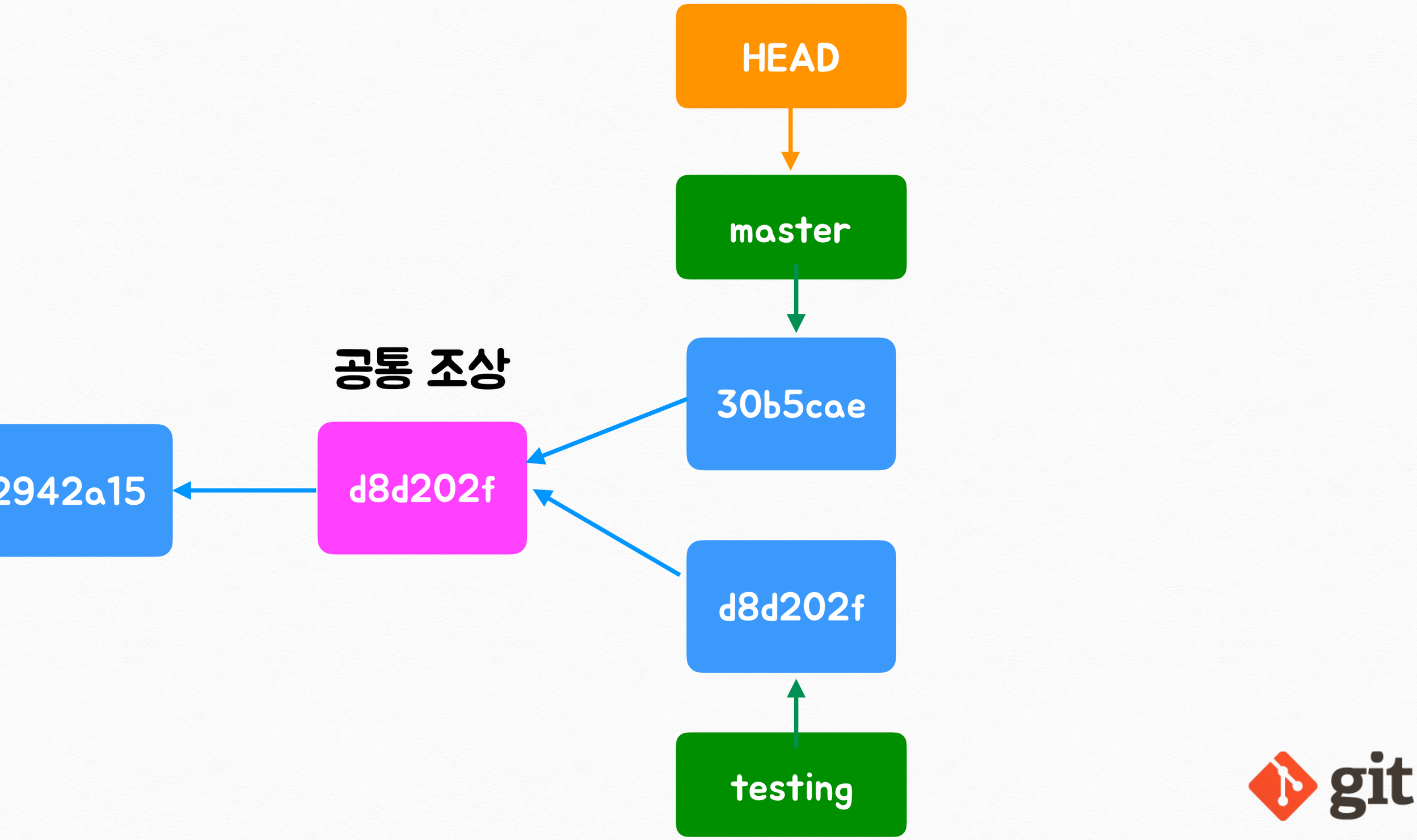
실습 D6 - merge (2)

결과

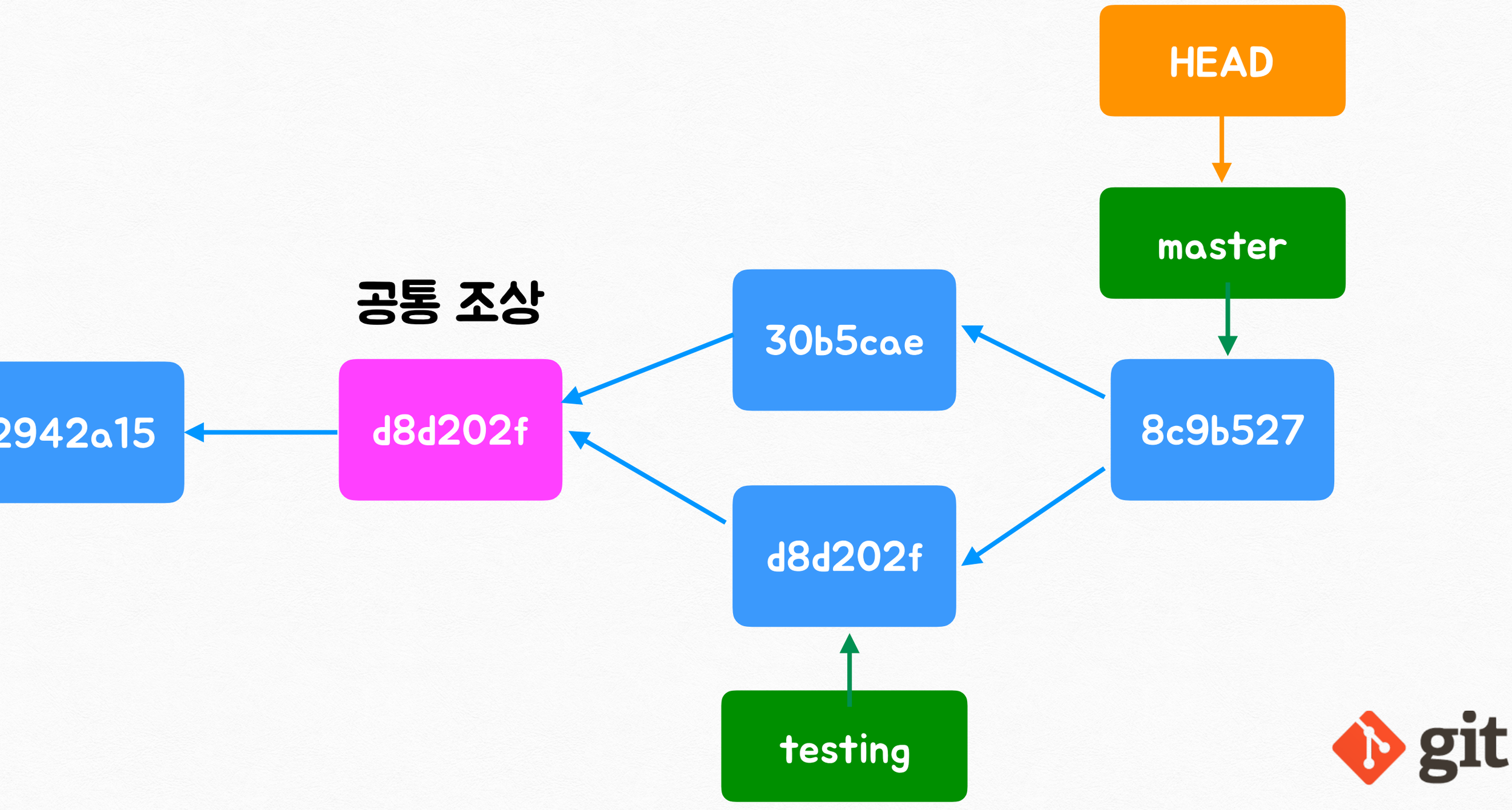
```
$ git branch -v
* master  30b5cae add a div
  testing a5f5bef add test.js

$ git merge testing
Auto-merging hello.html
Merge made by the 'recursive' strategy.
 hello.html | 1 +
 test.js    | 2 ++
 2 files changed, 3 insertions(+)
 create mode 100644 test.js

$ git branch -v
* master  8c9b527 Merge branch 'testing'
  testing a5f5bef add test.js
$
```

merge커밋 메시지를 요구!

master/testing 브랜치 상황 (8)



master/testing 브랜치 상황 (8)



실습 D6 - 현상황 로그 간단히 보기

결과

```
$ git log --oneline
```

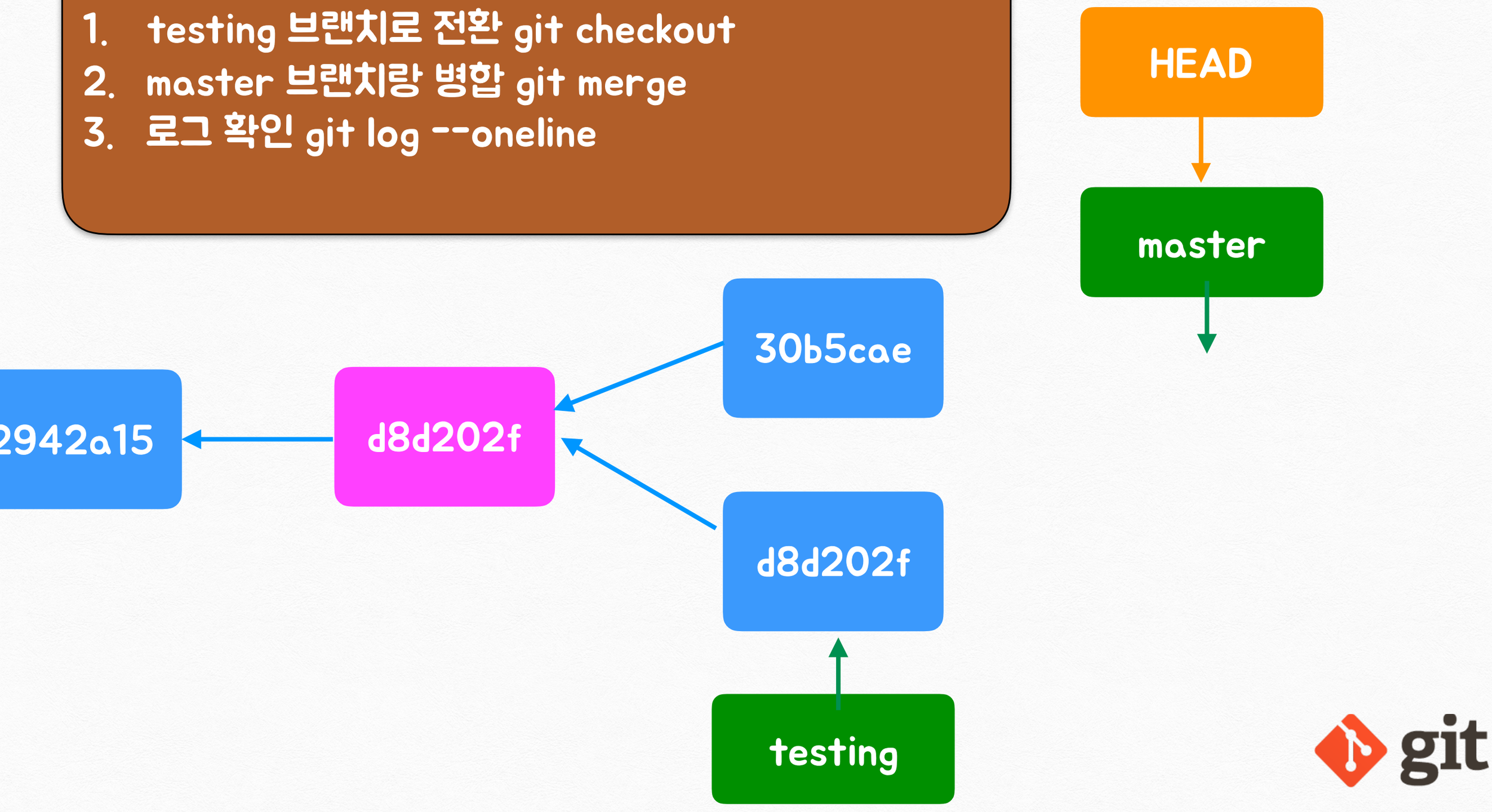
실습 D6 - 현상황 로그 간단히 보기

결과

```
$ git log --oneline
8c9b527 (HEAD -> master) Merge branch 'testing'
30b5cae add a div
a5f5bef (testing) add test.js
d8d202f add sub styling
2942a15 add styling
87bad9b add hello.html
$
```

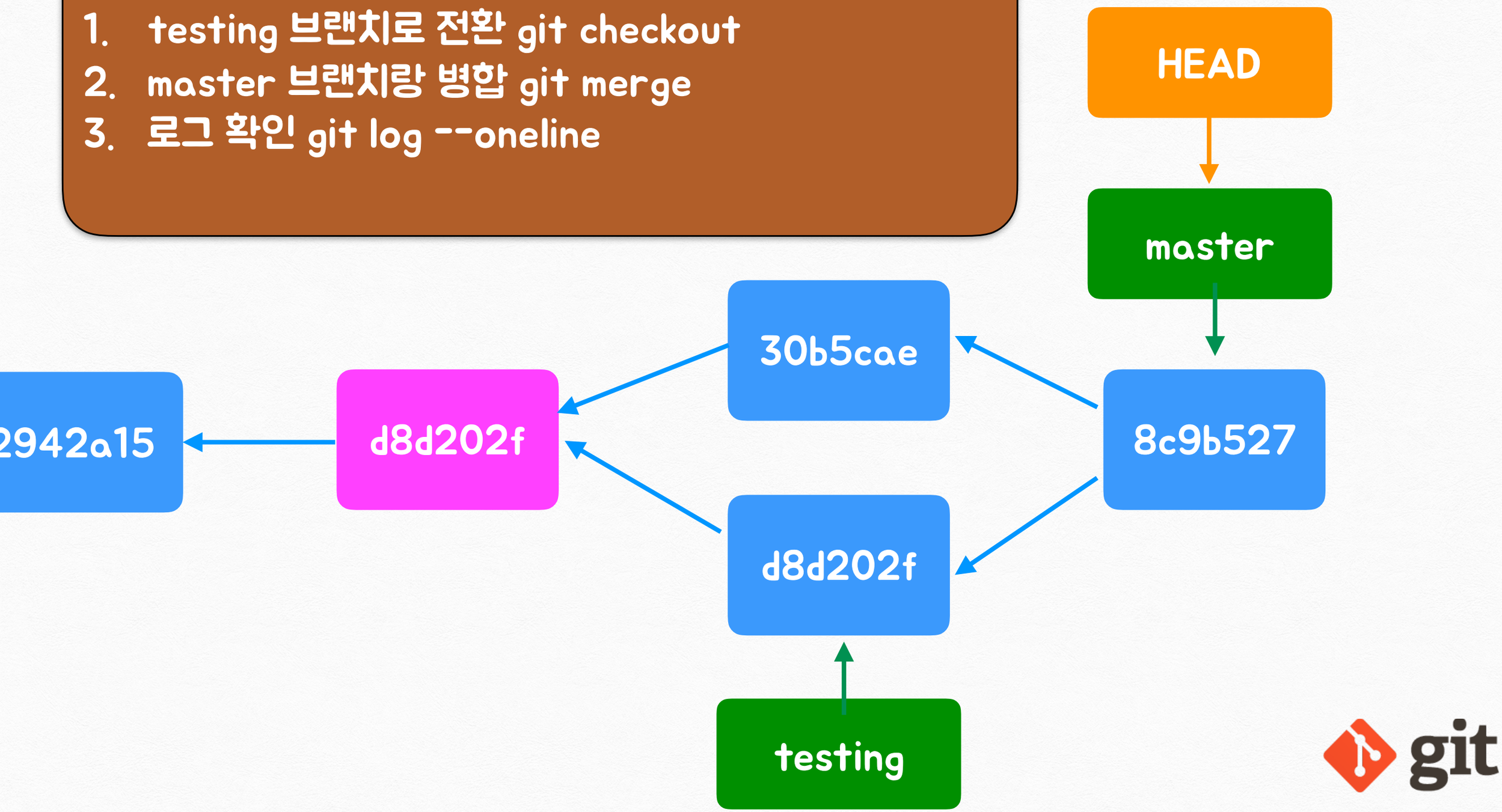
과제 #3 - testing 브랜치도 master로 병합

1. testing 브랜치로 전환 git checkout
2. master 브랜치랑 병합 git merge
3. 로그 확인 git log --oneline



과제 #3 - testing 브랜치도 master로 병합

1. testing 브랜치로 전환 git checkout
2. master 브랜치랑 병합 git merge
3. 로그 확인 git log --oneline



여기까지 요약 정리

git init

git add

git status

git commit

git log

git branch

git checkout

git merge

git reset --hard

리베이스 rebase

merge와 함께 자주 쓰임

푸쉬 전 로컬에서만 쓰도록 유의

리모트 저장소 설명 후 다시 언급

대졸 신입 A 이야기 (4)

고객사 별로 브랜치를 썼으면 좋았을 텐데

고객사별로 별도 디렉토리에 보관

심지어, 고객사 서버에 최종작업 내용이 있고, 우리에겐 없음

대리 승진 제안, 그러나 퇴사

두번째 직장은 신세계

다음 시간에 계속...

GitHub

활용 실습

GitHub 주요기능

Git 리모트 저장소

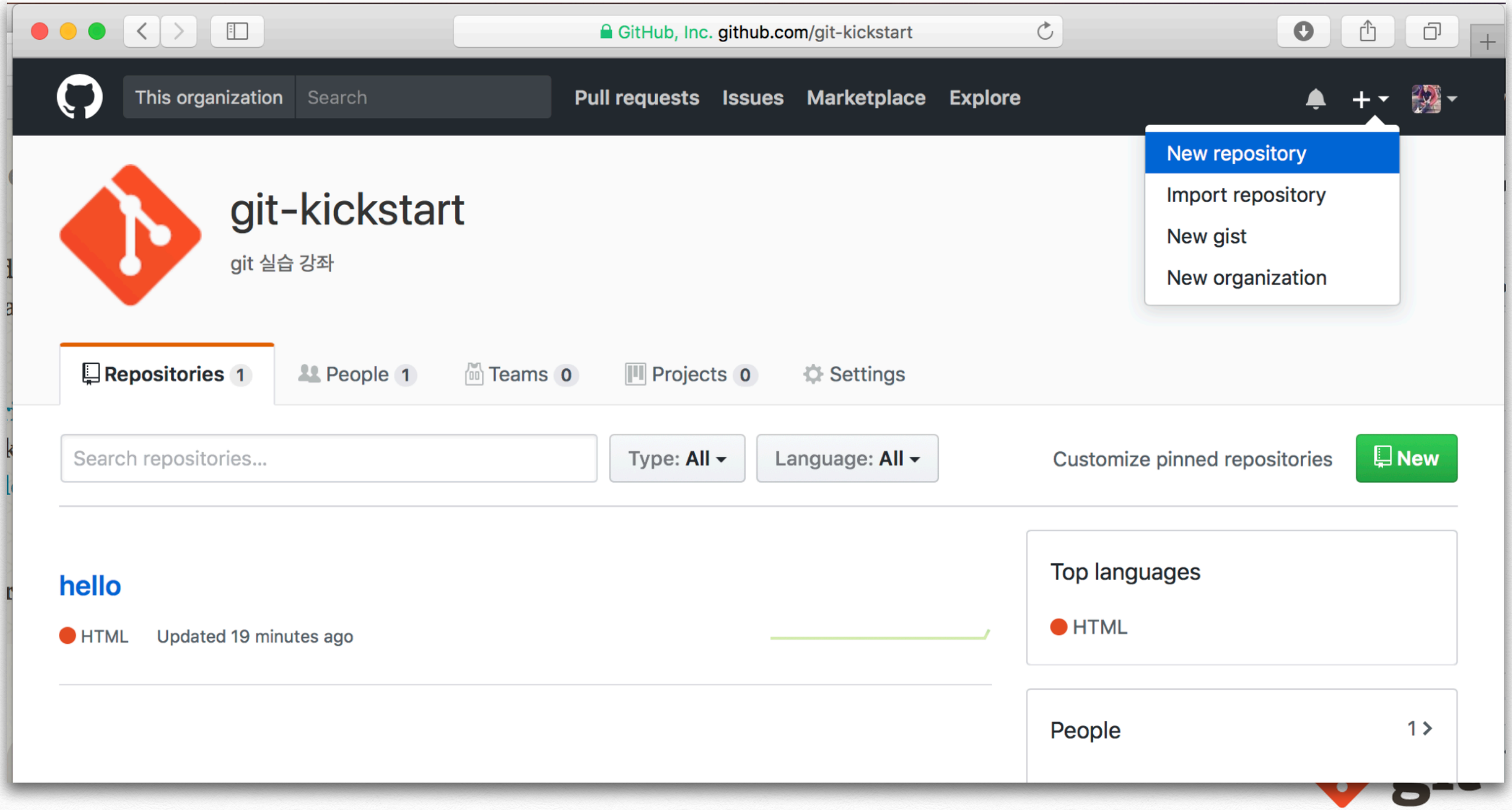
코드 리뷰

프로젝트 관리

문서화 위키

오픈 소스 프로젝트 & 커뮤니티 & 이슈 토론

실습 E1 - 첫 리모트 저장소 만들기



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner **Repository name**

hatemogi / hello

Great repository names are short and memorable. Need inspiration? How about **animated-winner**.

Description (optional)

첫 저장소

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

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

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** Add a license: **None**

Create repository

hatemogi / hello

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/hatemogi/hello.git

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# hello" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/hatemogi/hello.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/hatemogi/hello.git
git push -u origin master
```

...or import code from another repository

실습 E2 - 리모트 저장소로 push

결과

```
$ git remote add origin https://github.com/hatemogi/hello.git
$ git push -u origin master
```

GitHub 아이디/패스워드 입력

git-kickstart / hello

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided.

Edit

Add topics

6 commits **1** branch **1** release **0** contributors

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

홍길동 Merge branch 'testing' Latest commit 8c9b527 a day ago

hello.html	Merge branch 'testing'	a day ago
main.css	add styling	a day ago
sub.css	add sub styling	a day ago
test.js	add test.js	a day ago

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

실습 E3 - 리모트 저장소에서 클론

This repository Search Pull requests Issues Marketplace Explore

git-kickstart / hello

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided.

Edit

Add topics

6 commits 1 branch 1 release 0 contributors

Branch: master New pull request

Create new file Upload files Find file Clone or download

홍길동 Merge branch 'testing'

hello.html	Merge branch 'testing'	
main.css	add styling	
sub.css	add sub styling	
test.js	add test.js	3 days ago

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/git-kickstart/hello

Open in Desktop Download ZIP

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

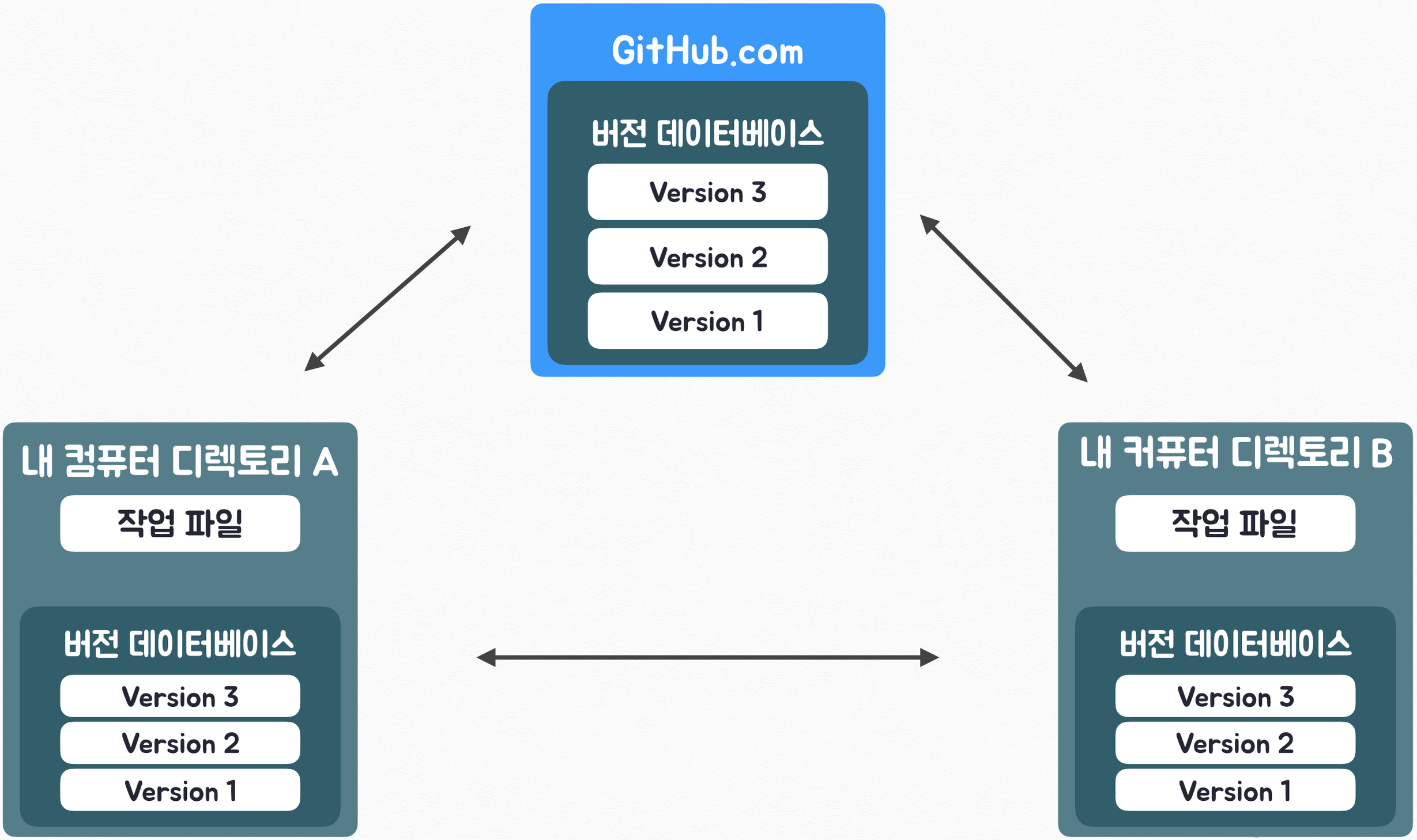
Add a README

실습 E3 - 리모트 저장소에서 클론

결과

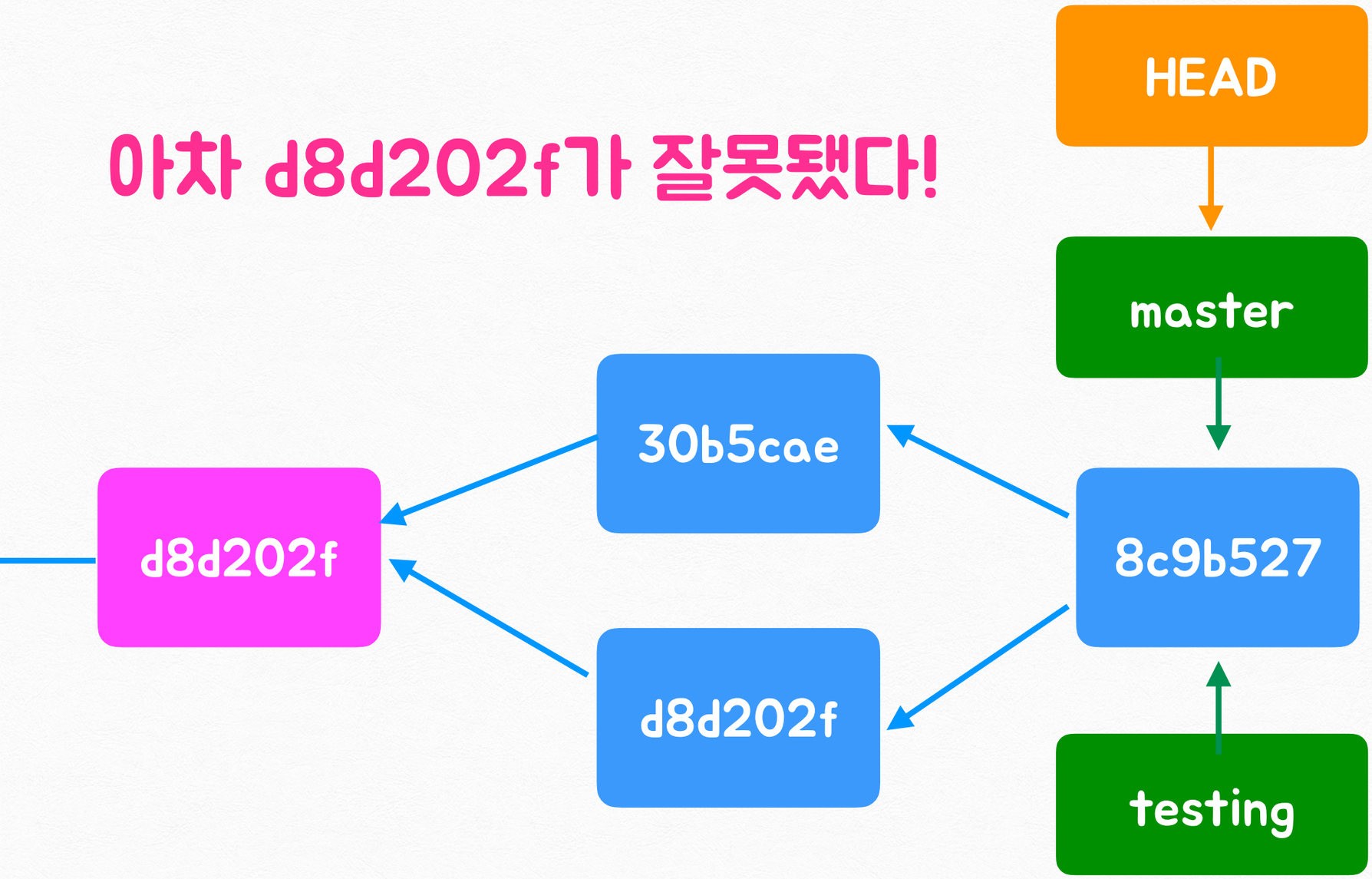
```
$ git clone https://github.com/your-id/hello.git hello2
```

실습 E3 - 클론 후 상태


GitHub.com
버전 데이터베이스
Version 3
Version 2
Version 1
내 컴퓨터 디렉토리 A
작업 파일
버전 데이터베이스
Version 3
Version 2
Version 1
내 커퓨터 디렉토리 B
작업 파일
버전 데이터베이스
Version 3
Version 2
Version 1

실습 E4 - revert로 복구

아차 d8d202f가 잘못됐다!



실습 E4 – git revert로 복구

결과

```
$ git revert d8d202f
```

실습 E4 - git revert로 복구

결과

```
$ git revert d8d202f
[master ceb82c5] Revert "add sub styling"
 1 file changed, 3 deletions(-)
 delete mode 100644 sub.css
$ git log --oneline
```

실습 E4 - git revert로 복구

결과

```
$ git revert d8d202f
[master ceb82c5] Revert "add sub styling"
 1 file changed, 3 deletions(-)
 delete mode 100644 sub.css
$ git log --oneline
ceb82c5 (HEAD -> master) Revert "add sub styling"
8c9b527 (origin/master, testing) Merge branch 'testing'
30b5cae add a div
a5f5bef add test.js
d8d202f add sub styling
2942a15 add styling
87bad9b (tag: c2) add hello.html
$
```

실습 E4 - revert로 복구

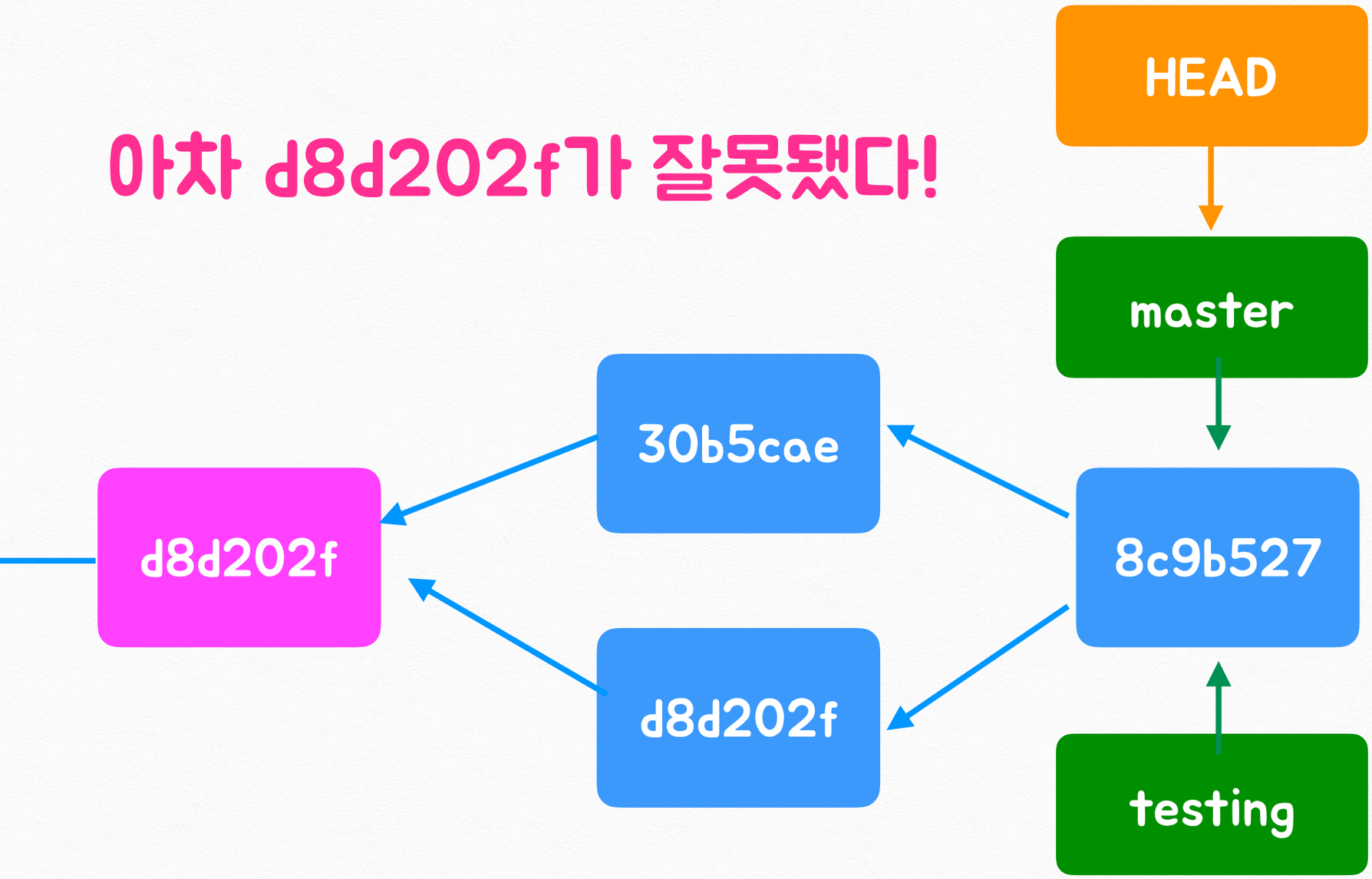
특정 커밋의 변경분을 복구하는 커밋을 기록

해당 커밋 내용을 원래대로 돌이키는 커밋

reset과의 차이점?

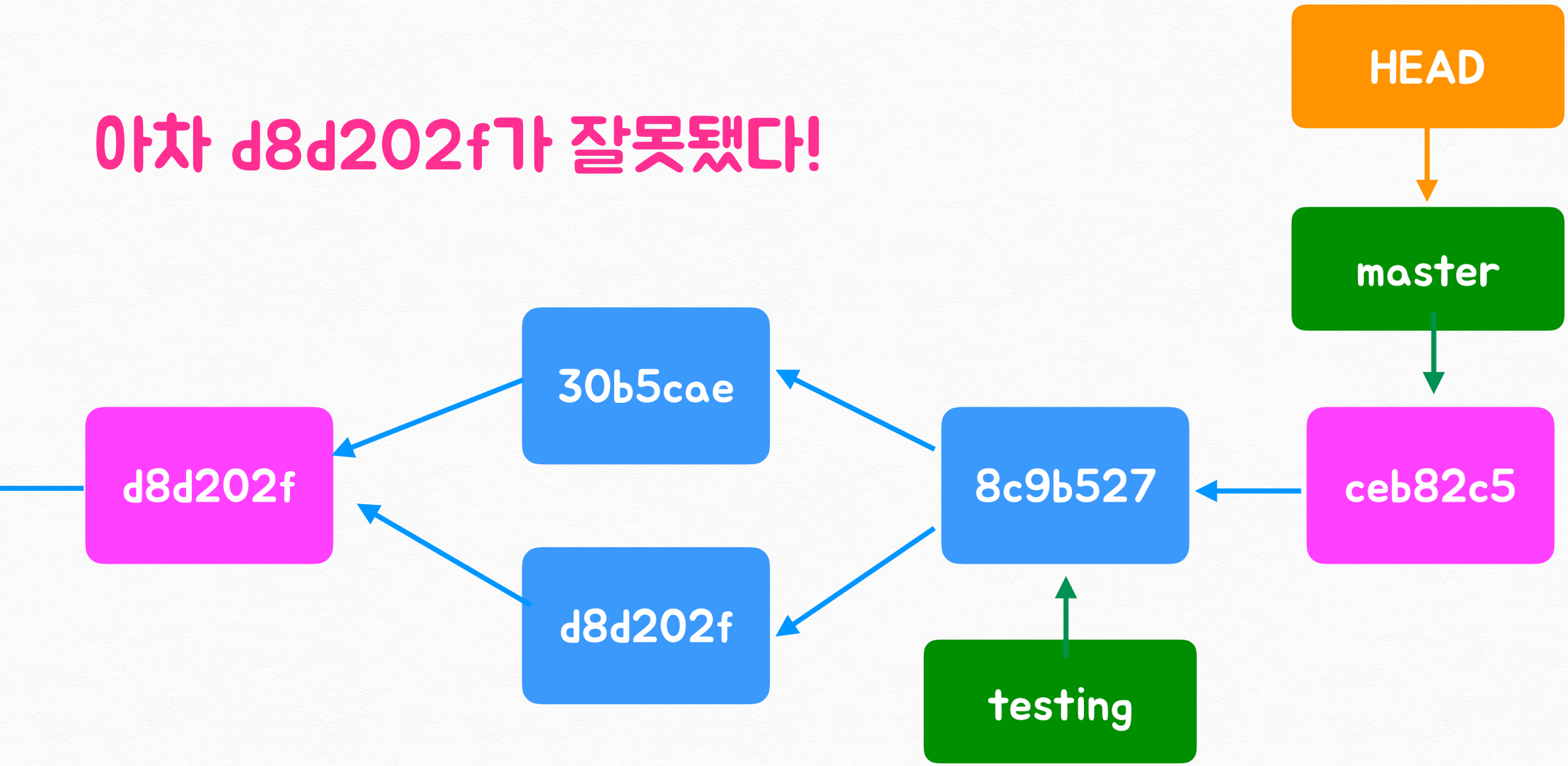
실습 E4 - revert로 복구

아차 d8d202f가 잘못됐다!



실습 E4 - revert로 복구

아차 d8d202f가 잘못됐다!



실습 E4 – reset과 revert의 차이점

reset

revert

참고: cherry-pick

실습 E5 - 프로젝트 포크 (Fork)

특정 저장소를 내 영역 (단체/개인)으로 복제

별도 작업 진행 후, 풀리퀘스트 (Pull-Request) 전송

수락되면 merge됨

과제용 단체 Organization

https://github.com/git-kickstart-숫자

조별 실습

조장이 단체orgarnization을 하나 만듭니다

조원은 조장에게 GitHub 아이디를 알려줍니다

GitHub, Inc. github.com/git-kickstart

This organization | Search | Pull requests | Issues | Marketplace | Explore

git-kickstart

git 실습 강좌

Repositories 1 | People 1 | Teams 0 | Projects 0 | Settings

Search repositories... | Type: All | Language: All | Customize pinned repositories | New

hello

HTML Updated 7 minutes ago

Top languages

HTML

People 1 >

hatemogi
Daehyun Kim

Invite someone

과제 #5

조장

단체를 만듭니다

조원들에게 초대장을 보냅니다

조원

조장에게 자신의 깃헙 아이디를 알려줍니다

대졸 신입 A 이야기 (5)

두번째 직장은 신세계

진정한 팀 협업 개발. XP가 유행하던 시기.

CVS를 한참 쓰다가 SVN으로 일부 전환

다음 시간에 계속...

GitHub

Desktop

Current Repository
hello

Current Branch
master

Fetch origin
Last fetched just now

An updated version of GitHub Desktop is available and will be installed at the next launch. See what's new or restart now.

Changes | History

- Merge branch 'testing' — 3 days ago by 홍길동
- add a div — 3 days ago by 홍길동
- add test.js — 3 days ago by 홍길동
- add sub styling — 3 days ago by 홍길동
- add styling — 3 days ago by 홍길동
- add hello.html — 4 days ago by 홍길동

Merge branch 'testing'

홍길동 8c9b527 2 changed files

- hello.html
- test.js

```
@@ -15,5 +15,6 @@
15 15      <h1>안녕하세요</h1>
16 16      <div>대학생을 위한 Git 실습</div>
17 17      </main>
   18 +    <script src="test.js"></script>
18 19    </body>
19 20  </html>
```

Git 고급 주제

실전 활용 Git 고급 주제

Git stash

Git tag

멀티 리모트 저장소

Git bisect

Git reflog

대졸 신입 A 이야기 (6)

SVN으로 통합되다가, 세상에 Git이 출현

사람들은 변화를 좋아하지 않음

SVN대비 충분한 매력이 뭐야? 브랜치? 그게 왜 필요?

Git 내부 구조

Git 기초 오브젝트

https://medium.com/happyprogrammer-in-jeju/git-internals-81b34f85fe53

감사합니다