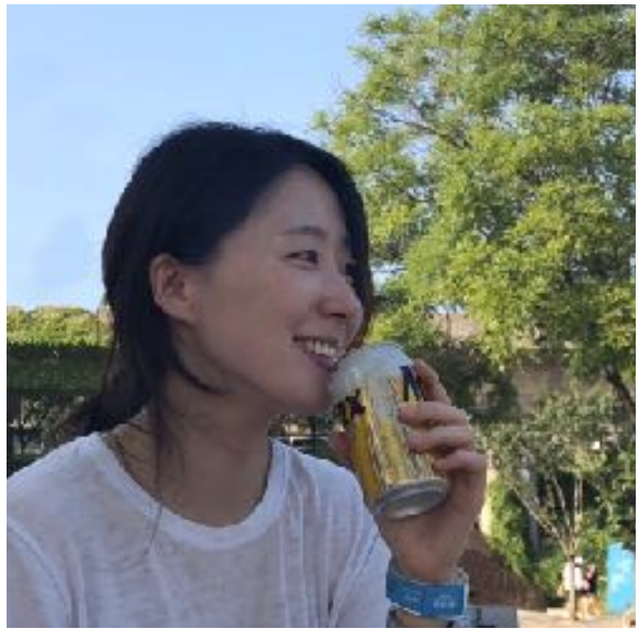
미정의 친절한 git 강의

Part 1.

2019.07.09.

소프트웨어 마에스트로 10기



전 미 정

Blink

BFT

BBL

maptales

Keras

Microsoft® Most Valuable Professional

Award Categories
AI

First year awarded:
2018

Number of MVP Awards:
1


GitLab

Part 1.

git 개념 / 기본 사용법

Part 2.

git 추가 명령어 / git lab

Part 1.

git 개념 / 기본 사용법

- git, 뭔가요?
- 어떻게 사용하나요?
- branch, 뭔가요?
- 충돌이 무서워요 😫
- remote를 연결하고 싶어요

Part 2.

git 추가 명령어 / git lab

- commit, 되돌릴 수 없나요?
- rebase, 대체 뭐죠?
- 추가 명령어가 궁금해요
- 오픈소스에 기여하고 싶어요
- gitlab의 엄청난 기능을 알려주세요

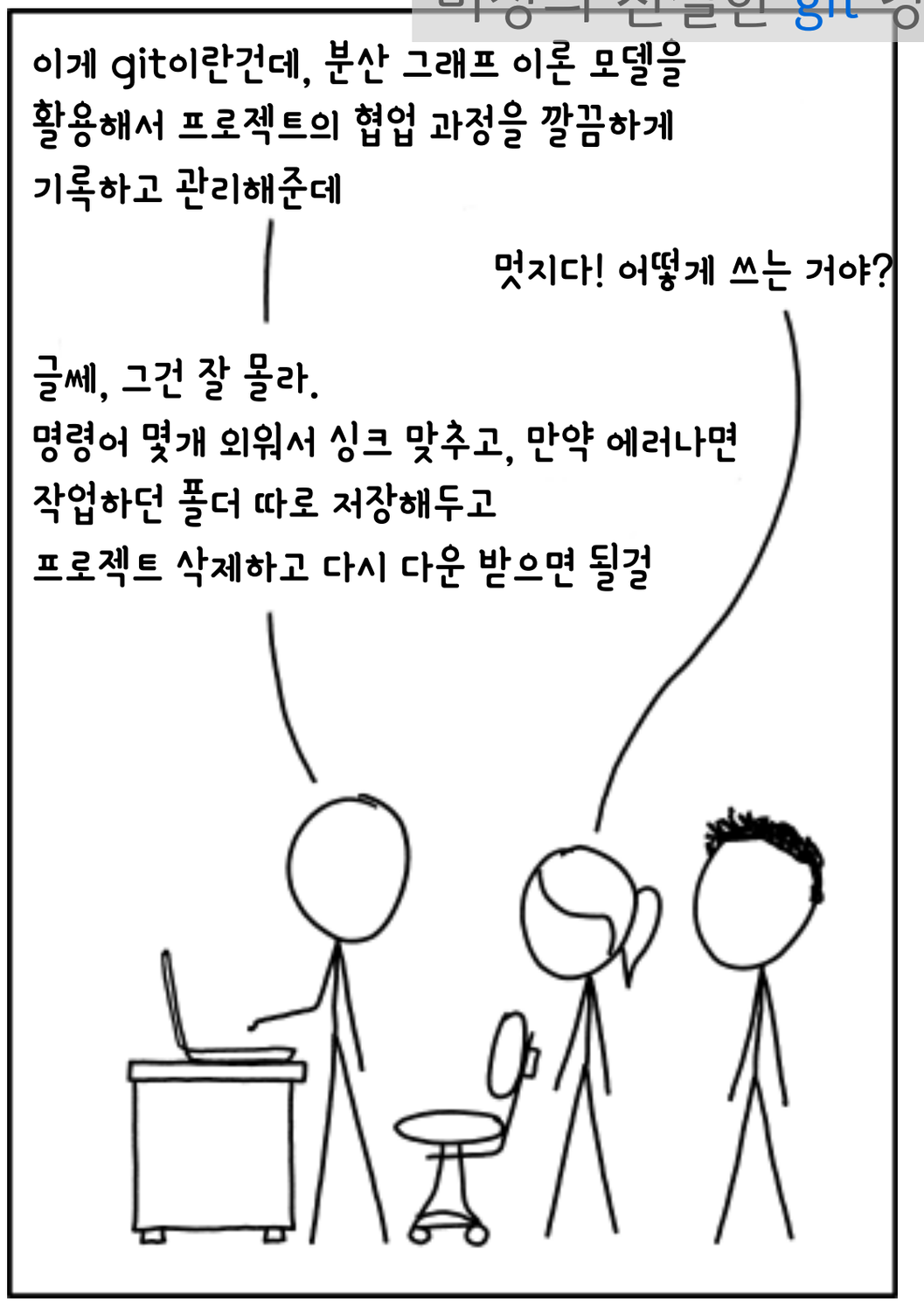
“Whoah, I’ve just read this quick tuto about git and oh my god it is cool. I feel now super comfortable using it, and I’m not afraid at all to break something.”.

— said no one ever.

“와아, 나 방금 git 튜토리얼 읽었는데 이거 정말 너무 멋지다! 이제 git 사용하는게 너무 편하게 느껴지고 아무것도 두렵지 않아”.

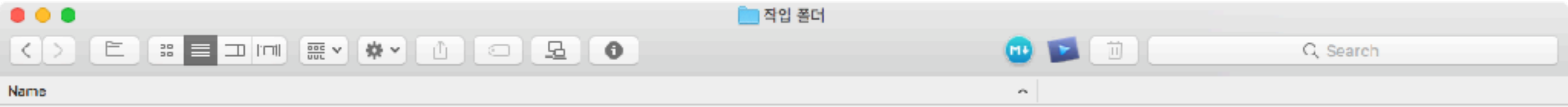
— 아무도 이렇게 말한 적 없음

https://medium.freecodecamp.org/understanding-git-for-real-by-exploring-the-git-directory-1e079c15b807



https://medium.freecodecamp.org/understanding-git-for-real-by-exploring-the-git-directory-1e079c15b807

Mary 작업



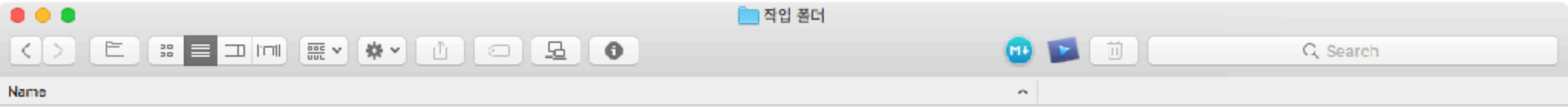
0203_프로젝트 파일.git

0205_프로젝트 파일0.9.git

0207_프로젝트 파일1.0_수정.git

프로젝트 파일1.2_Mary수정.git

(최종)프로젝트 파일2_Mary수정.git



0203_프로젝트 파일.git

0205_프로젝트 파일0.9.git

0207_프로젝트 파일1.0_수정.git

프로젝트 파일1.2_Mary수정_업데이트_John추가0210.git

프로젝트 파일1.2_Mary수정_최종_John수정_0215.git

(최종)프로젝트 파일2_Mary수정_John최종.git

버전 관리 시스템

Version Control System

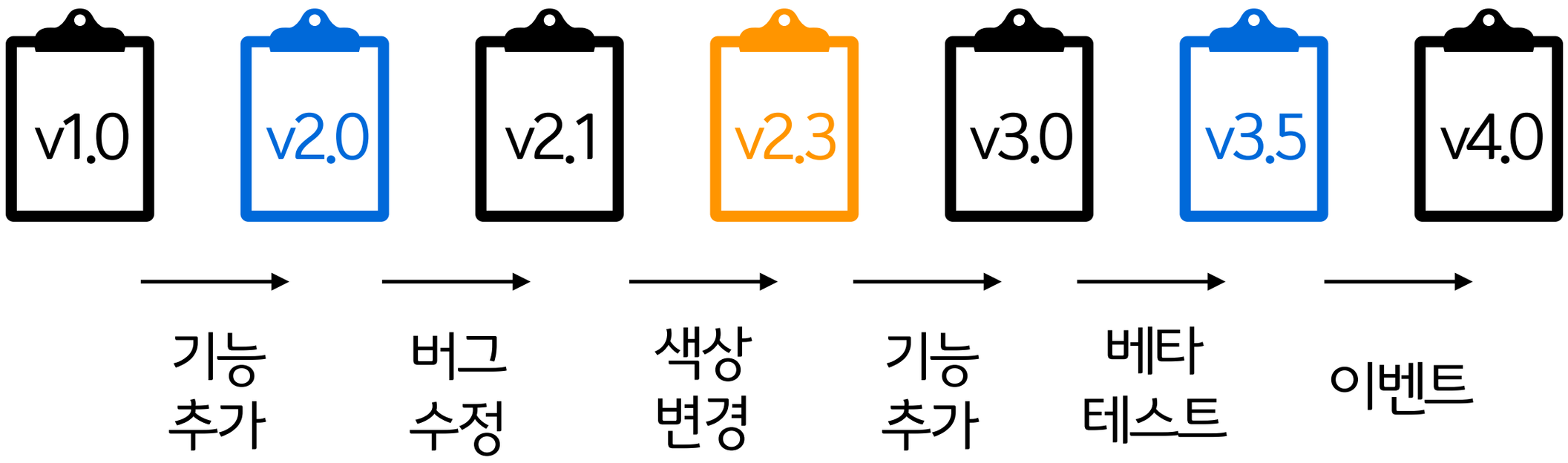
(VCS)

Software Configuration Management

(SCM)

버전 관리 시스템

(VCS)



버전 관리 시스템

(VCS)

누가

언제

무엇을

어떻게

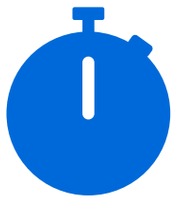
버전 관리 시스템

(VCS)

기록

추적

보관



복원


git


SUBVERSION


Visual Studio
Team Foundation Server

Linus Torvalds

git

가볍고

빠르고

유연한

분산형

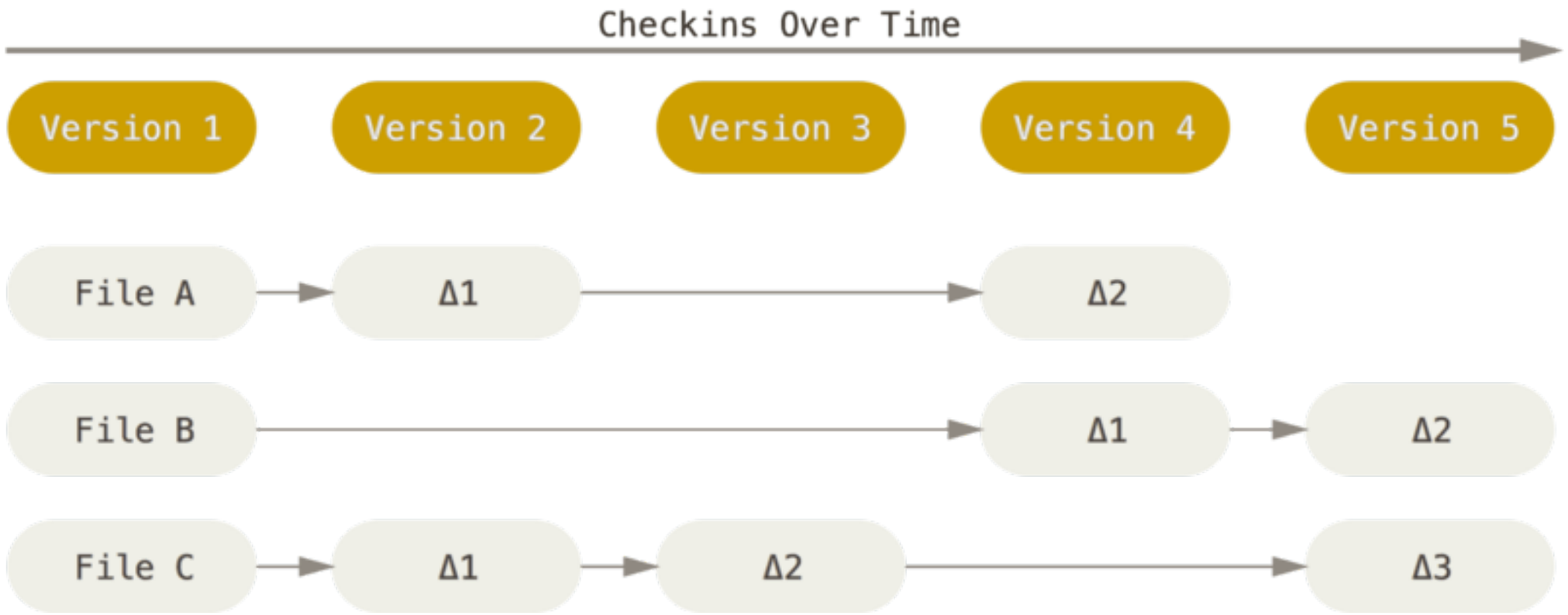
VCS

핵심 of git

현재 상태를 그대로 저장한다

Snapshot!! not a delta!

기존 방식



각 파일의 변화된 내용 저장

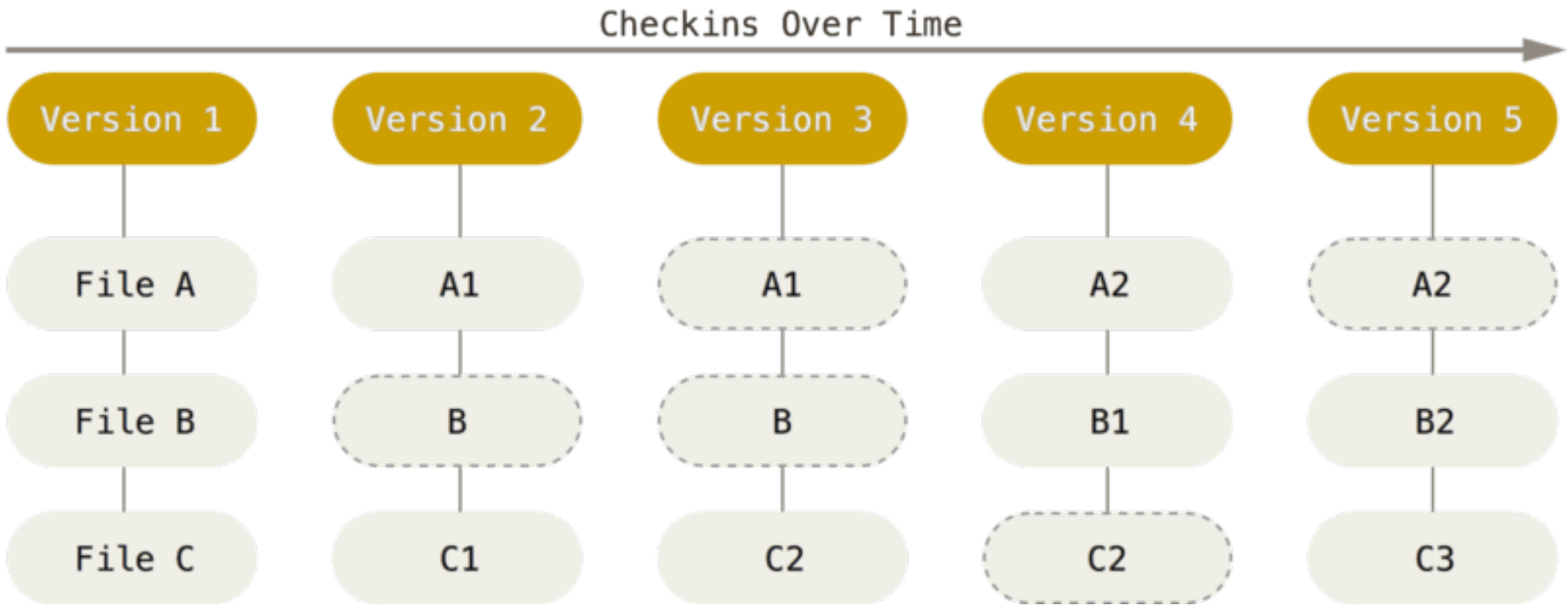
새로운 파일과 기존 파일 비교

파일 변화값 저장

...

https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

git 방식



commit할 때의 상태를 snapshot으로 저장

https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

snapshot

snapshot



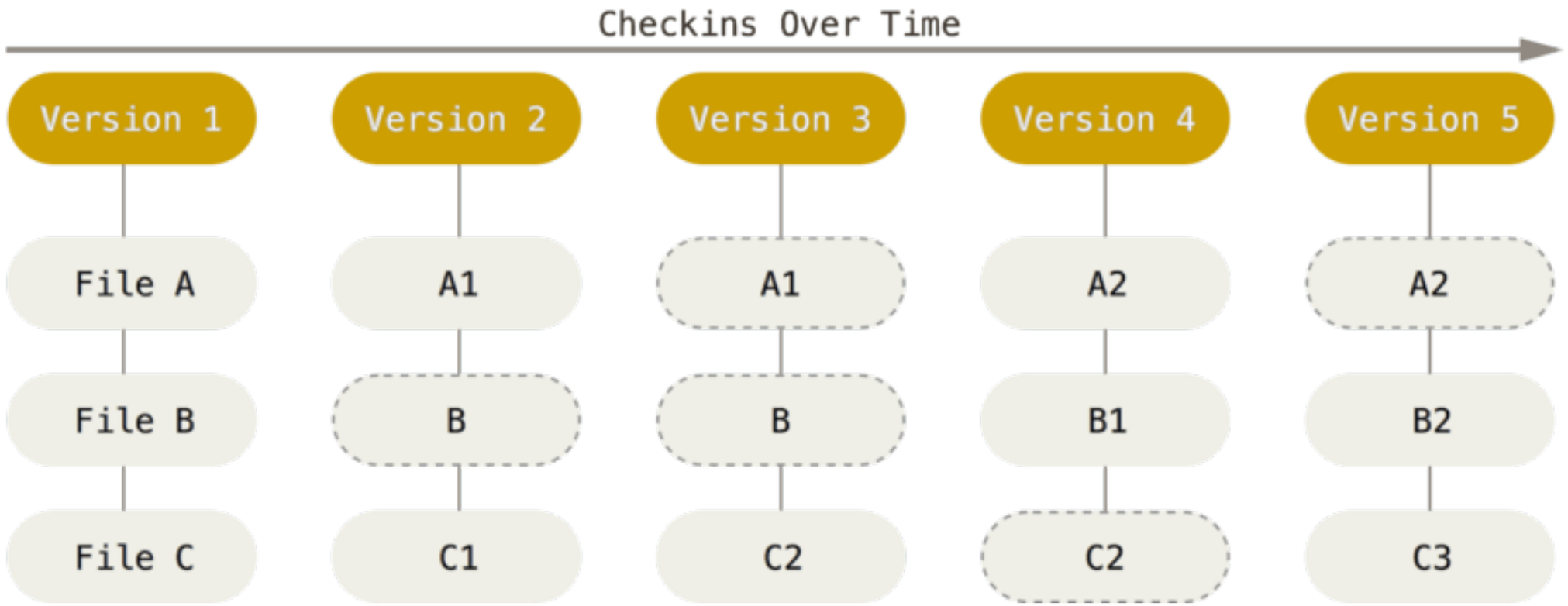
영화 장면을 screen shot으로 찍어둔다

--

commit을 할때의 파일의 상태를

Sanpshot 으로 남긴다

git 방식

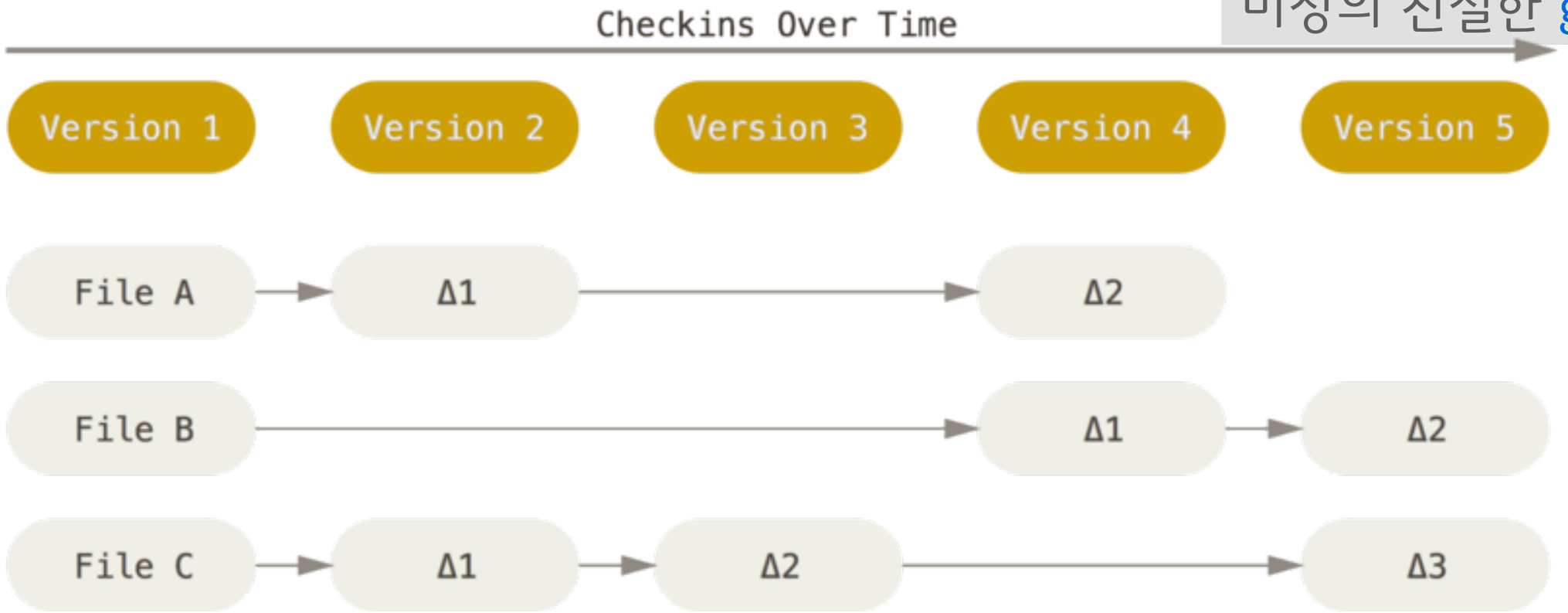


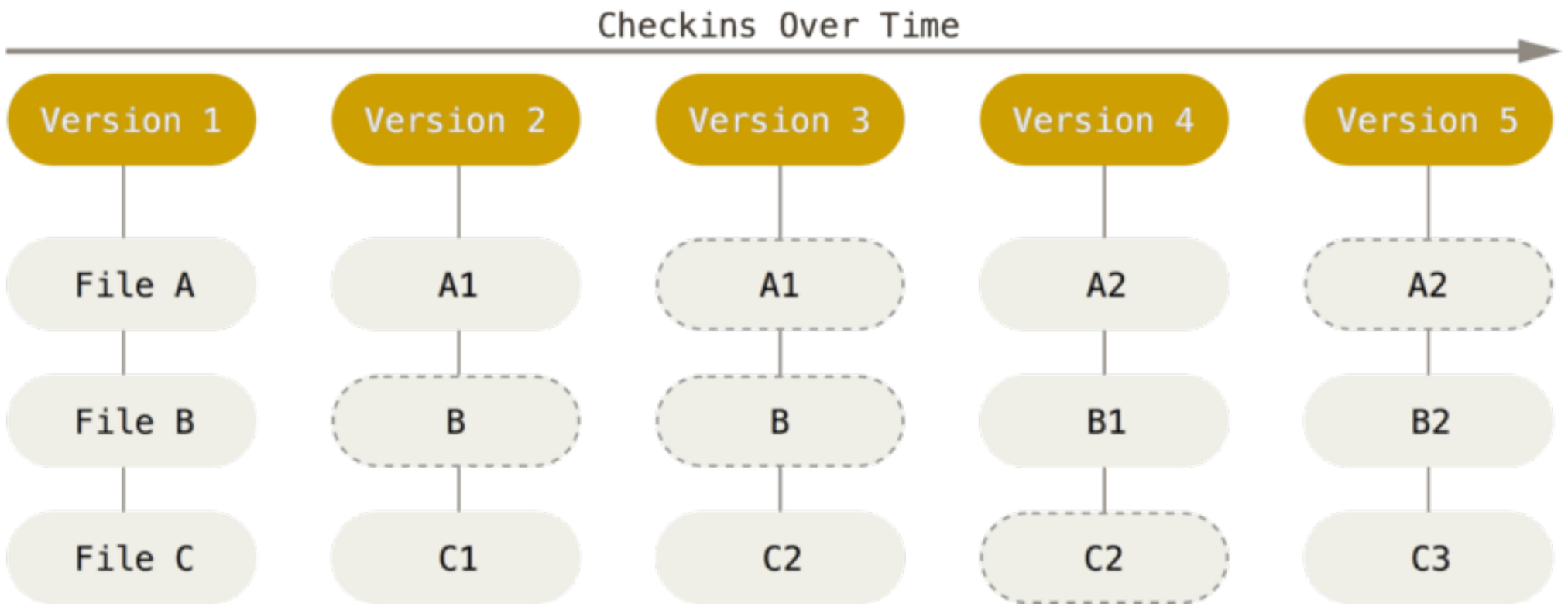
commit할 때의 상태를 snapshot으로 저장

현재 상태를 snapshot으로 남기고, snapshot에 대한 참조를 저장

변화없는 파일은 파일에 대한 링크 유지

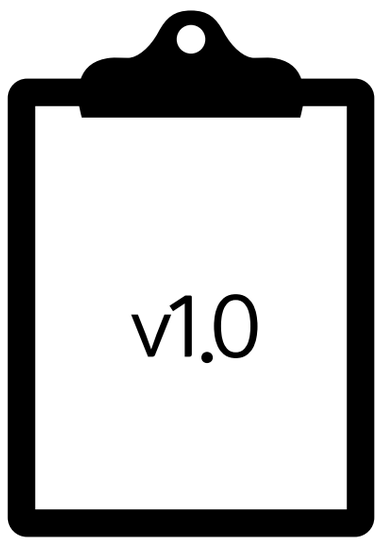
https://git-scm.com/book/en/v2/Getting-Started-Git-Basics





https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

커밋



개념적 = snapshot

기술적 = commit

설치 *https://git-scm.com/downloads*

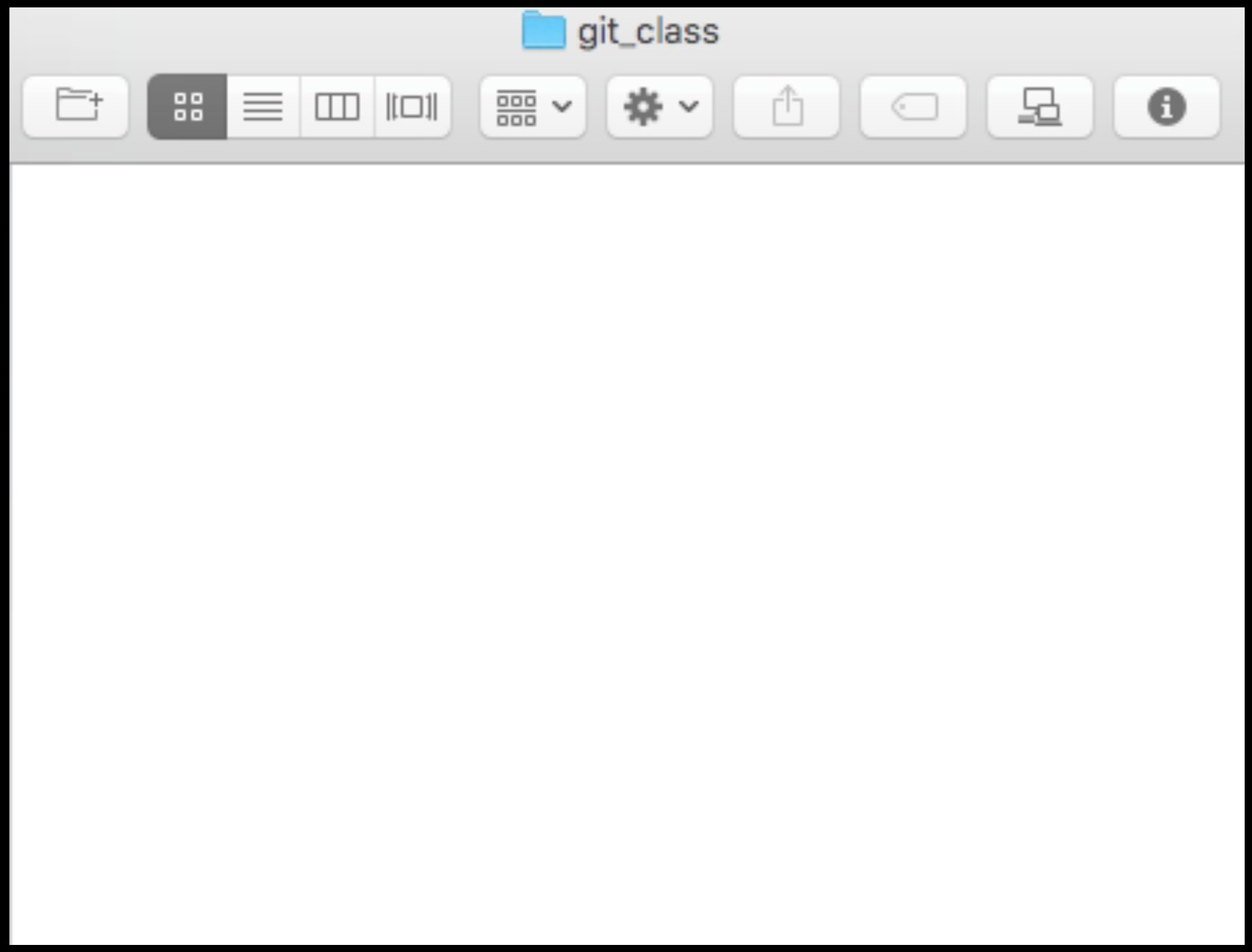
설치 확인

```
git --version
```

Working Tree

git_class

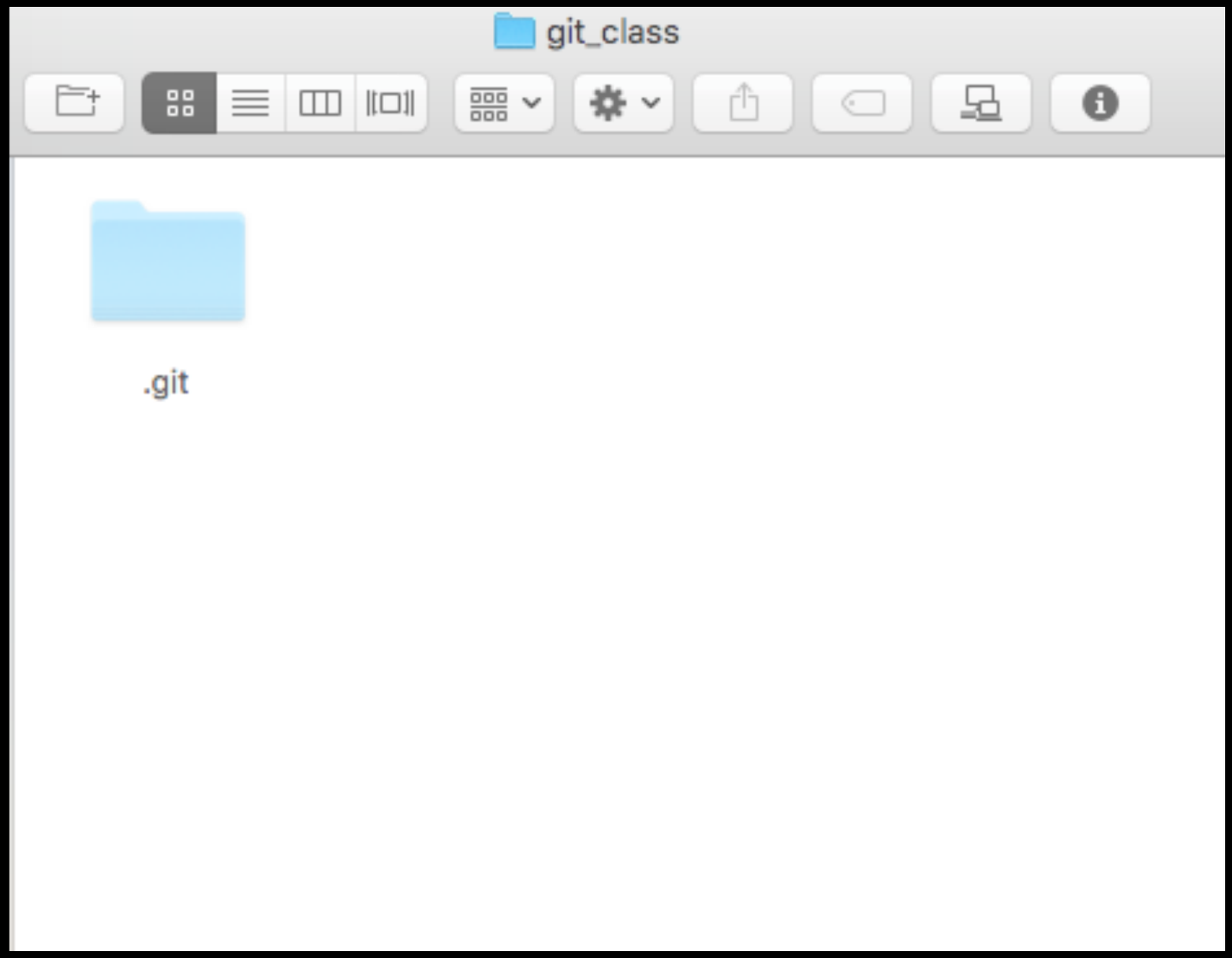

git_class


git_class

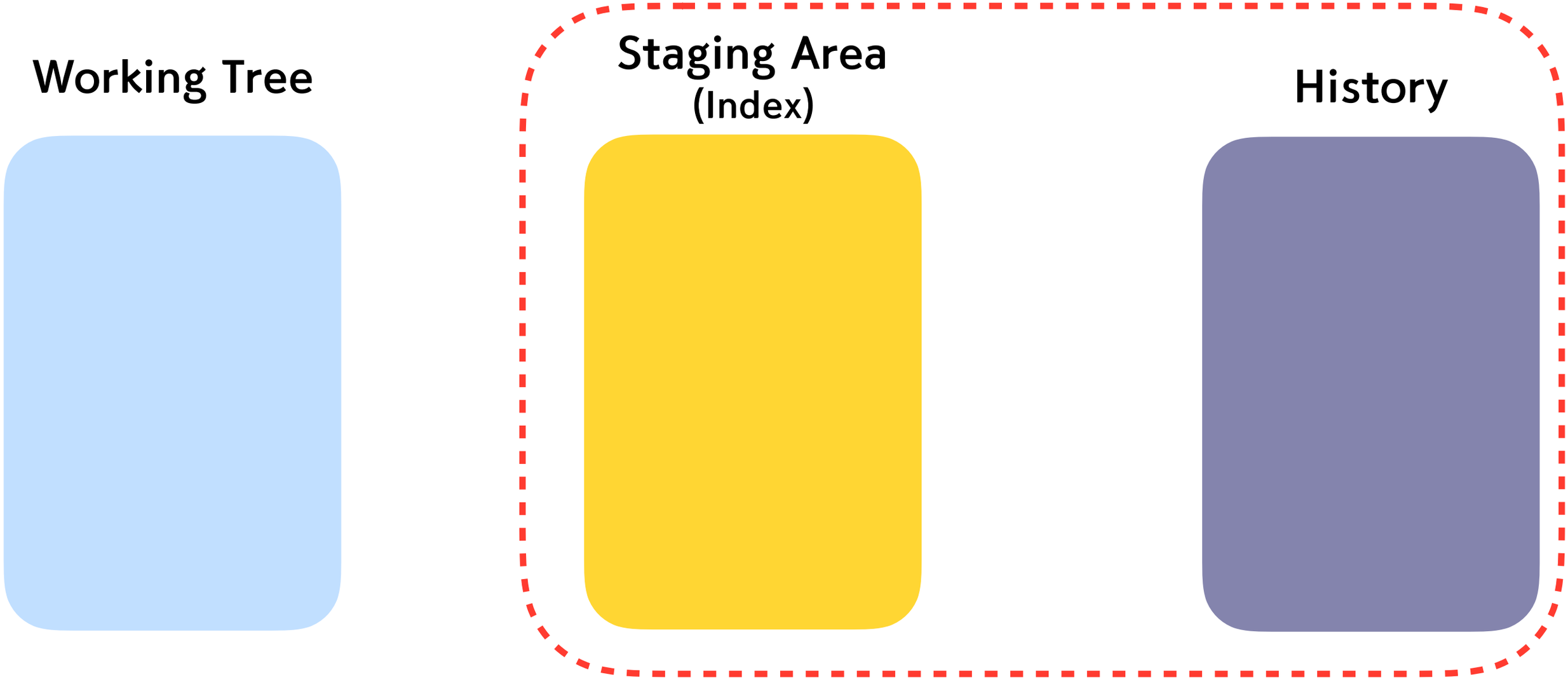
git init

git init

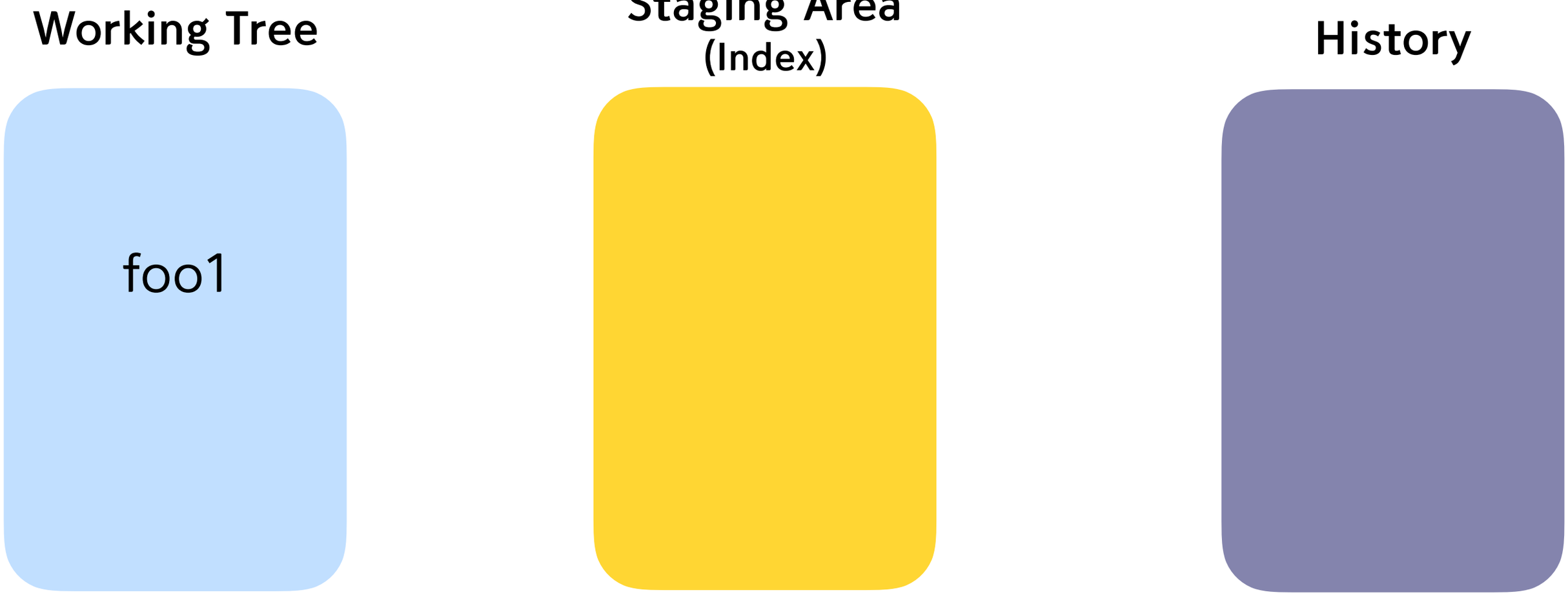

git_class


git_class
.git

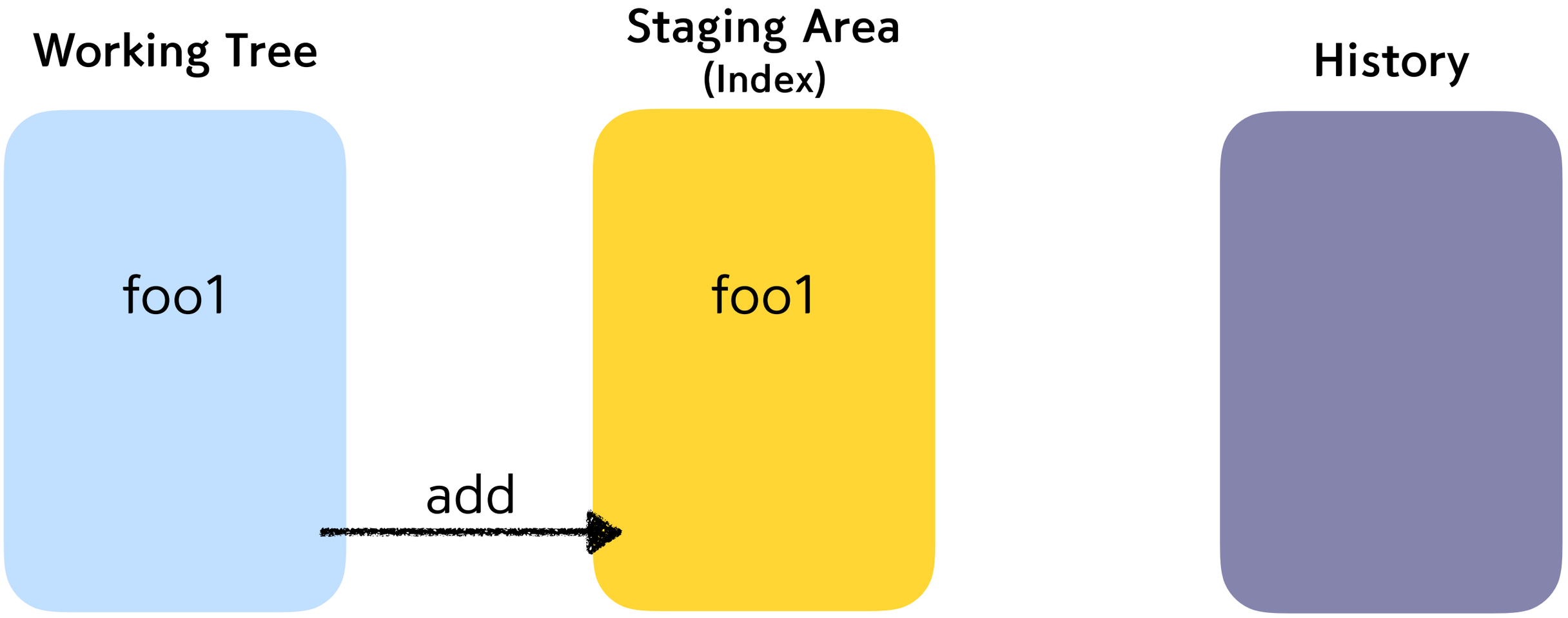
저장소



저장소



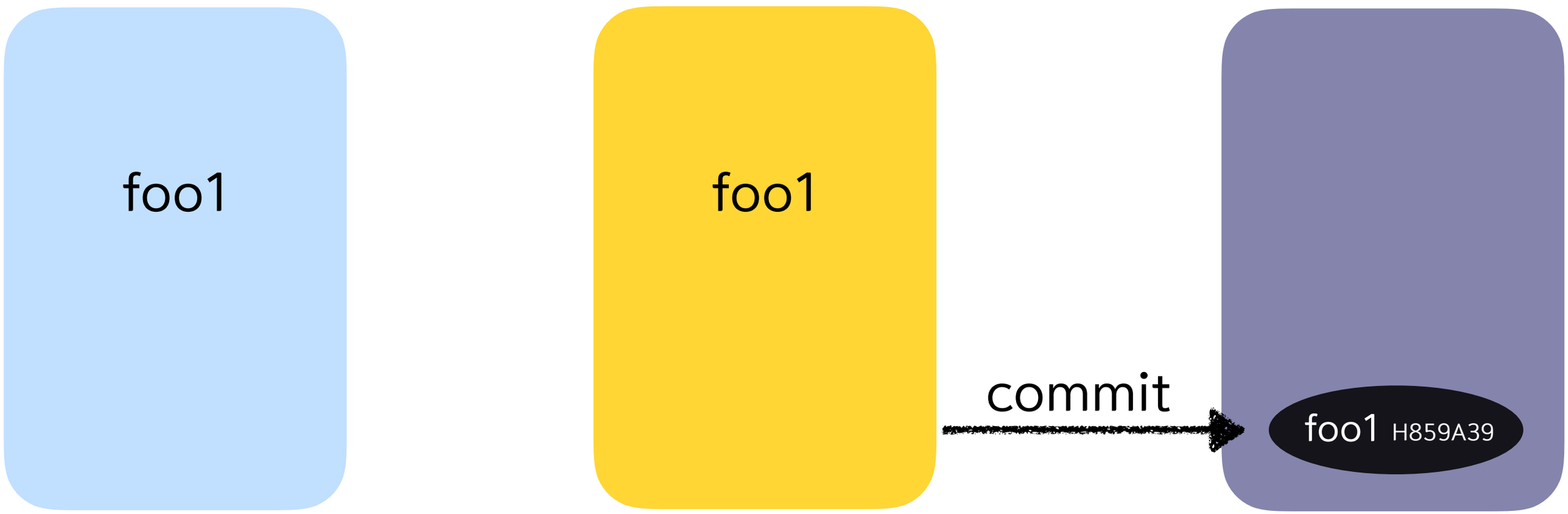
저장소



저장소

확인

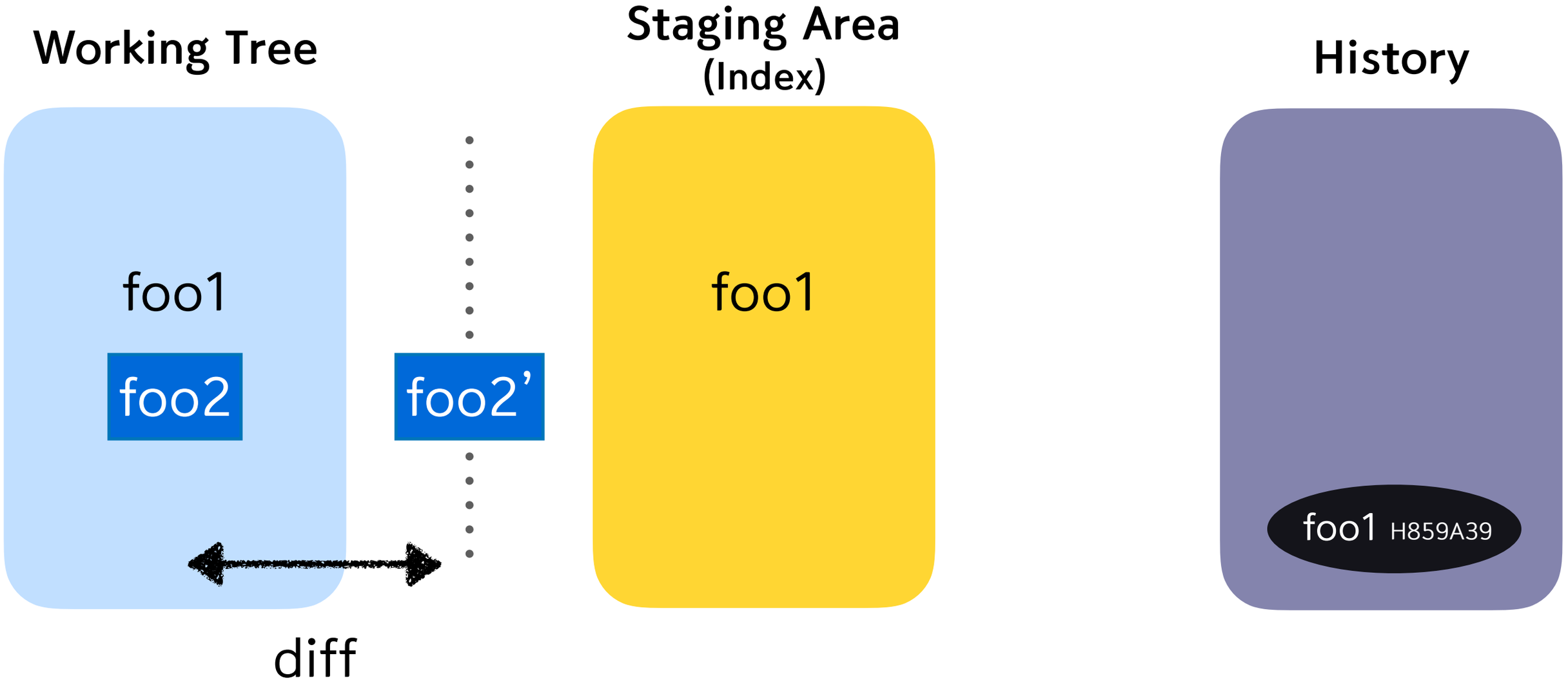
```
git log

git log -p
```



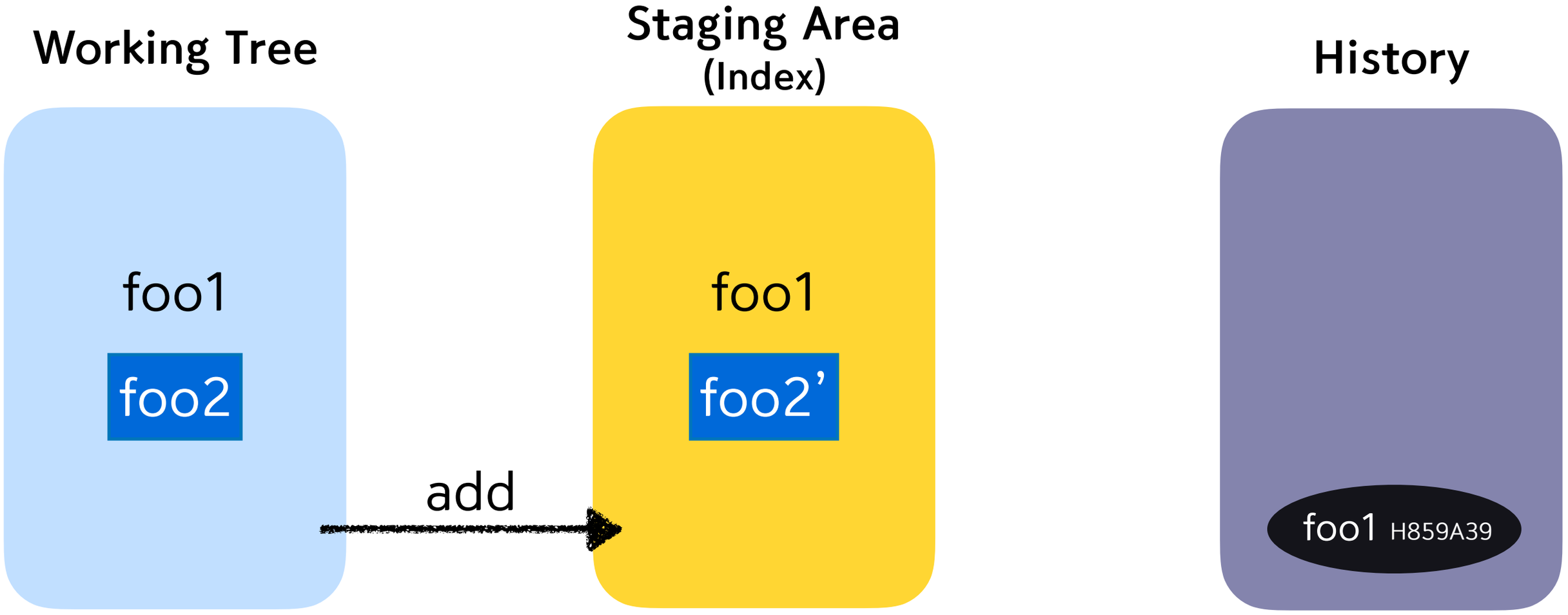
저장소


Working Tree
foo1
foo2
Staging Area
(Index)
foo1
History
foo1 H859A39

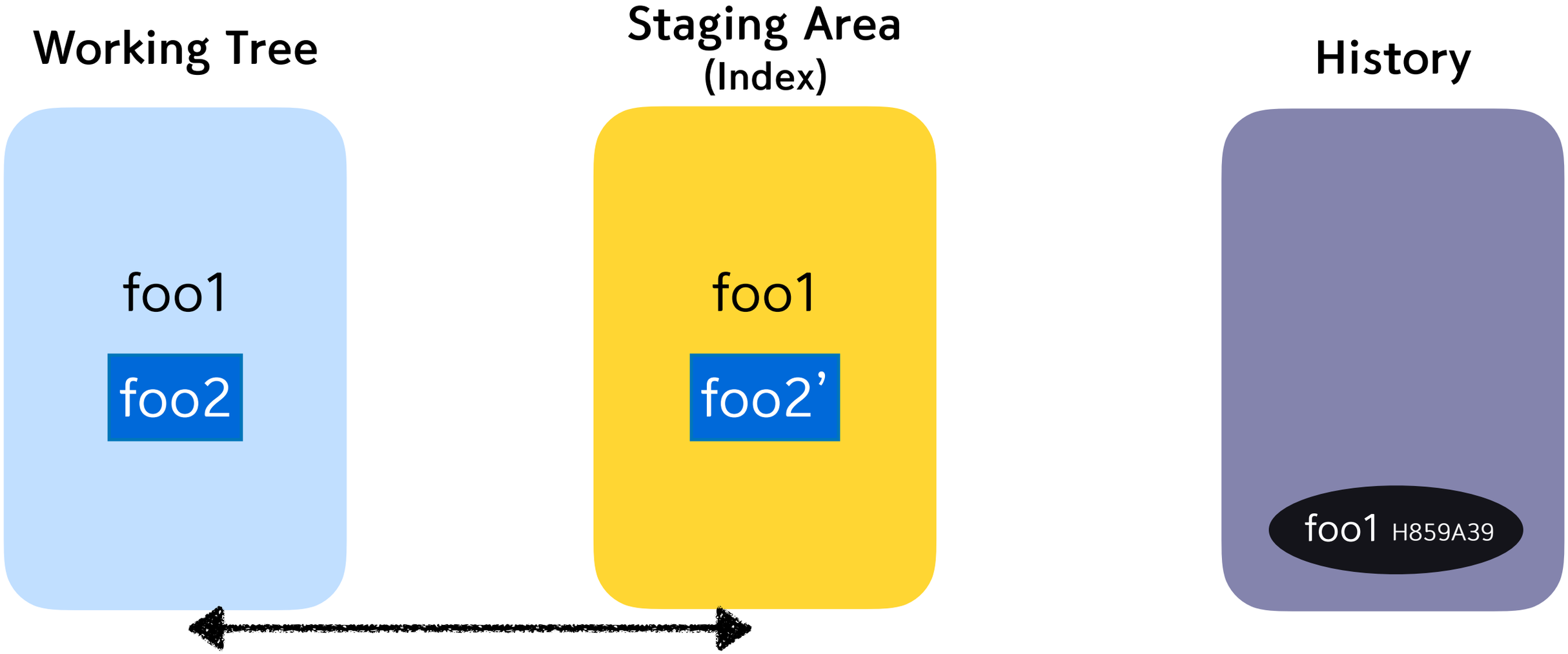
저장소


Working Tree
foo1
foo2
foo2’
Staging Area
(Index)
foo1
History
foo1 H859A39
diff

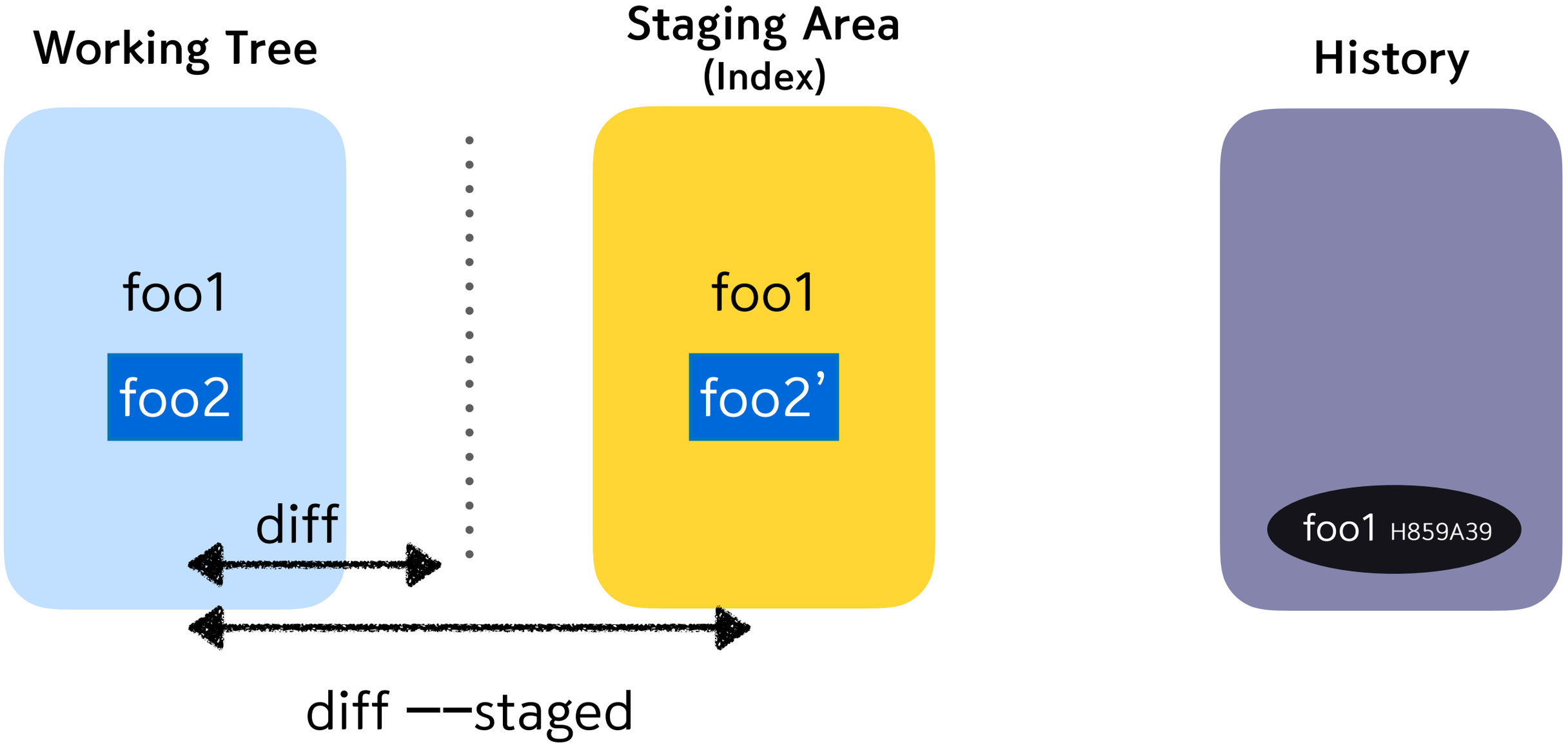
저장소


Working Tree
foo1
foo2
add
Staging Area
(Index)
foo1
foo2’
History
foo1 H859A39

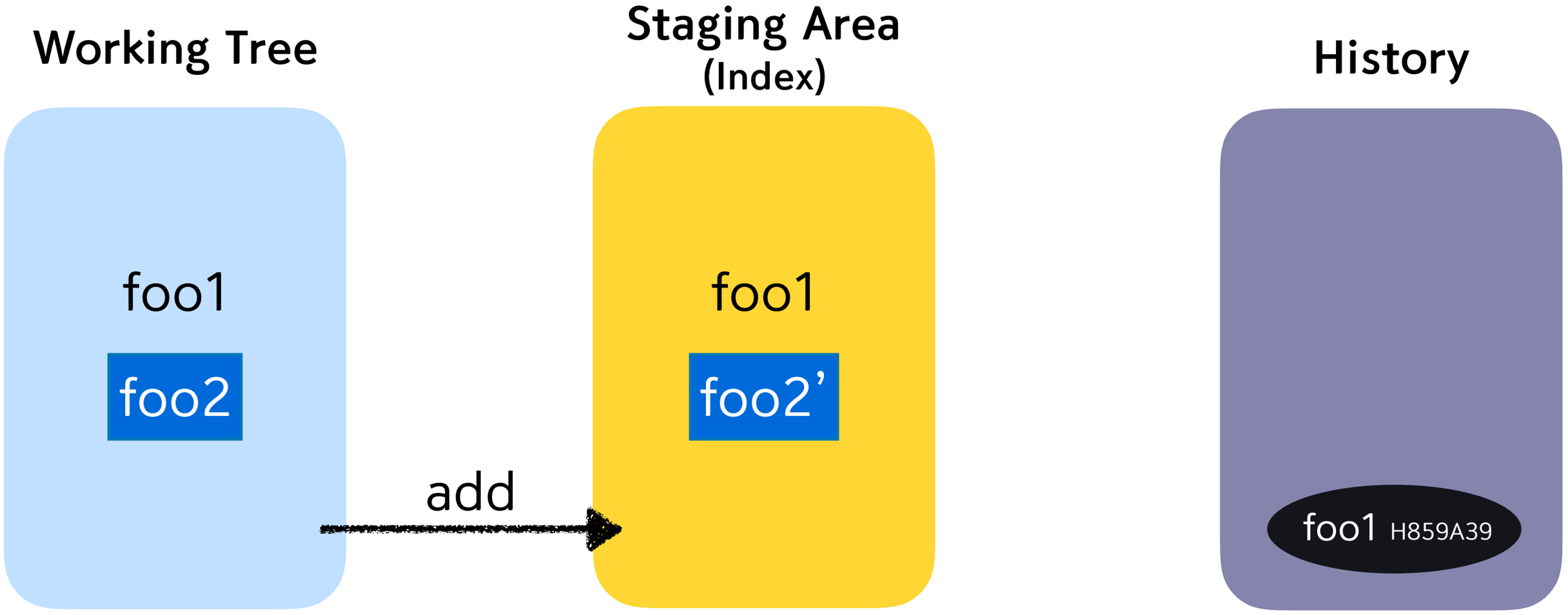
저장소


Working Tree
foo1
foo2
Staging Area
(Index)
foo1
foo2’
History
foo1 H859A39
diff ––staged

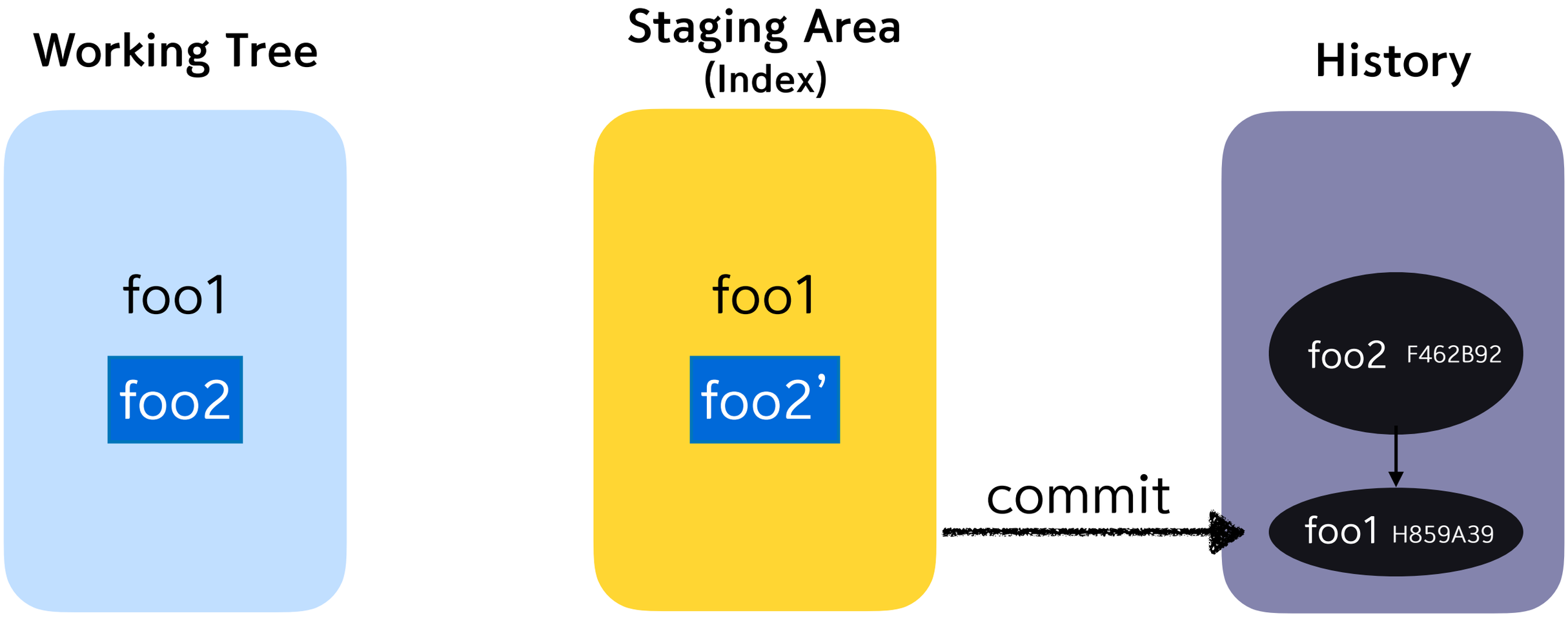
저장소


Working Tree
Staging Area
(Index)
History
foo1
foo2
foo1
foo2’
foo1 H859A39
diff
diff ––staged

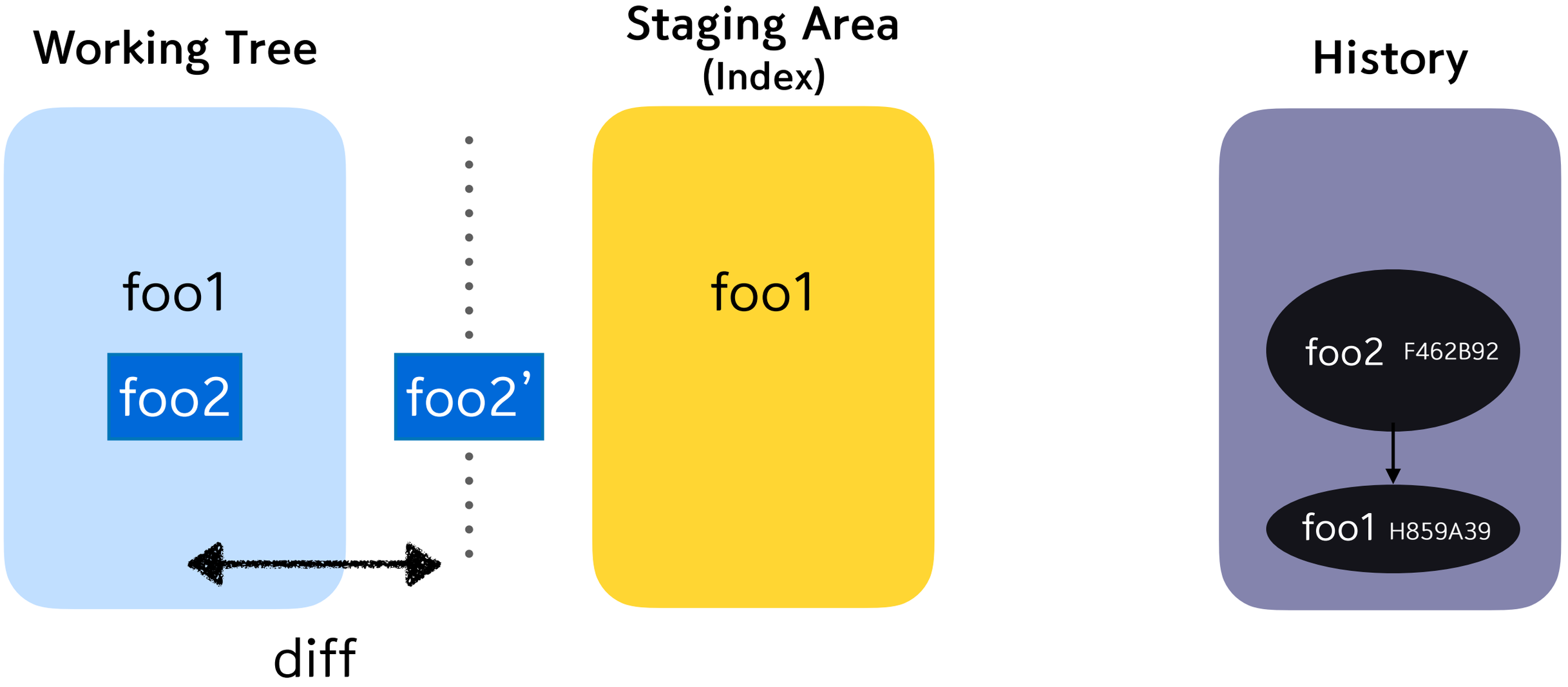
저장소


Working Tree
foo1
foo2
add
Staging Area
(Index)
foo1
foo2’
History
foo1 H859A39

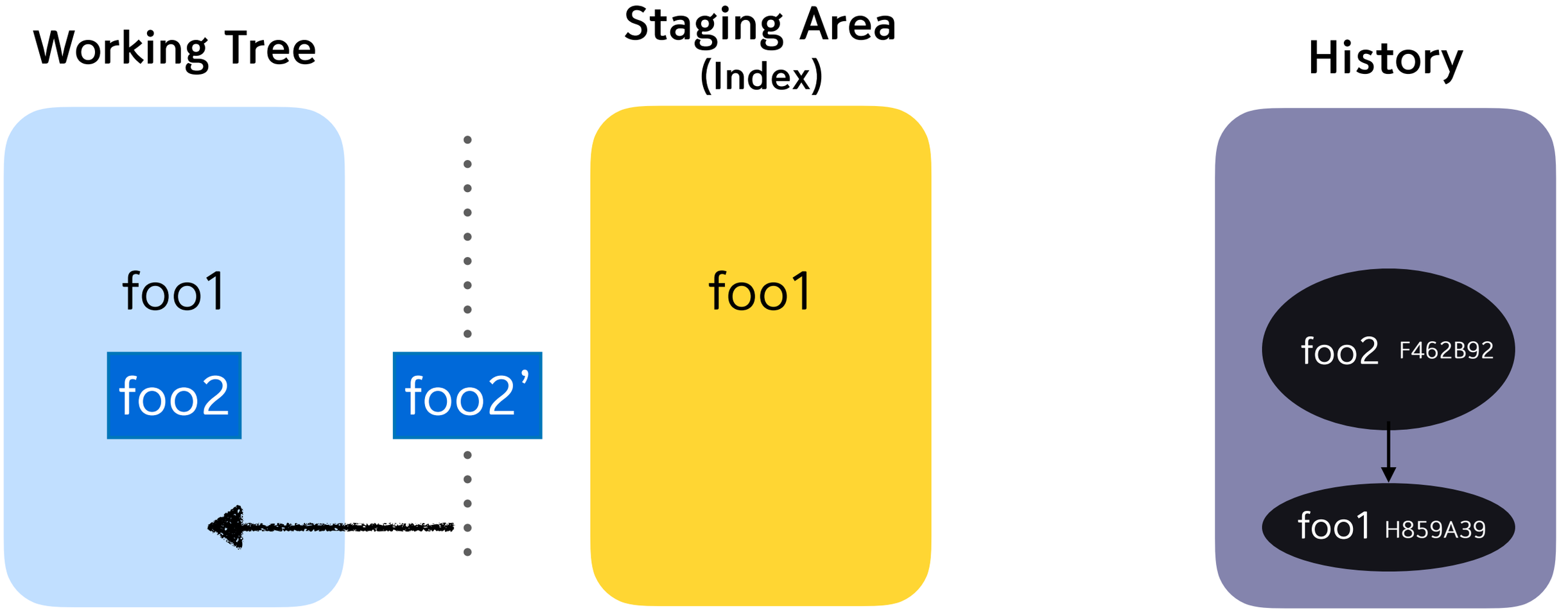
저장소


Working Tree
foo1
foo2
Staging Area
(Index)
foo1
foo2'
History
foo2 F462B92
foo1 H859A39
commit

저장소


Working Tree
foo1
foo2
foo2’
diff
Staging Area
(Index)
foo1
History
foo2 F462B92
foo1 H859A39

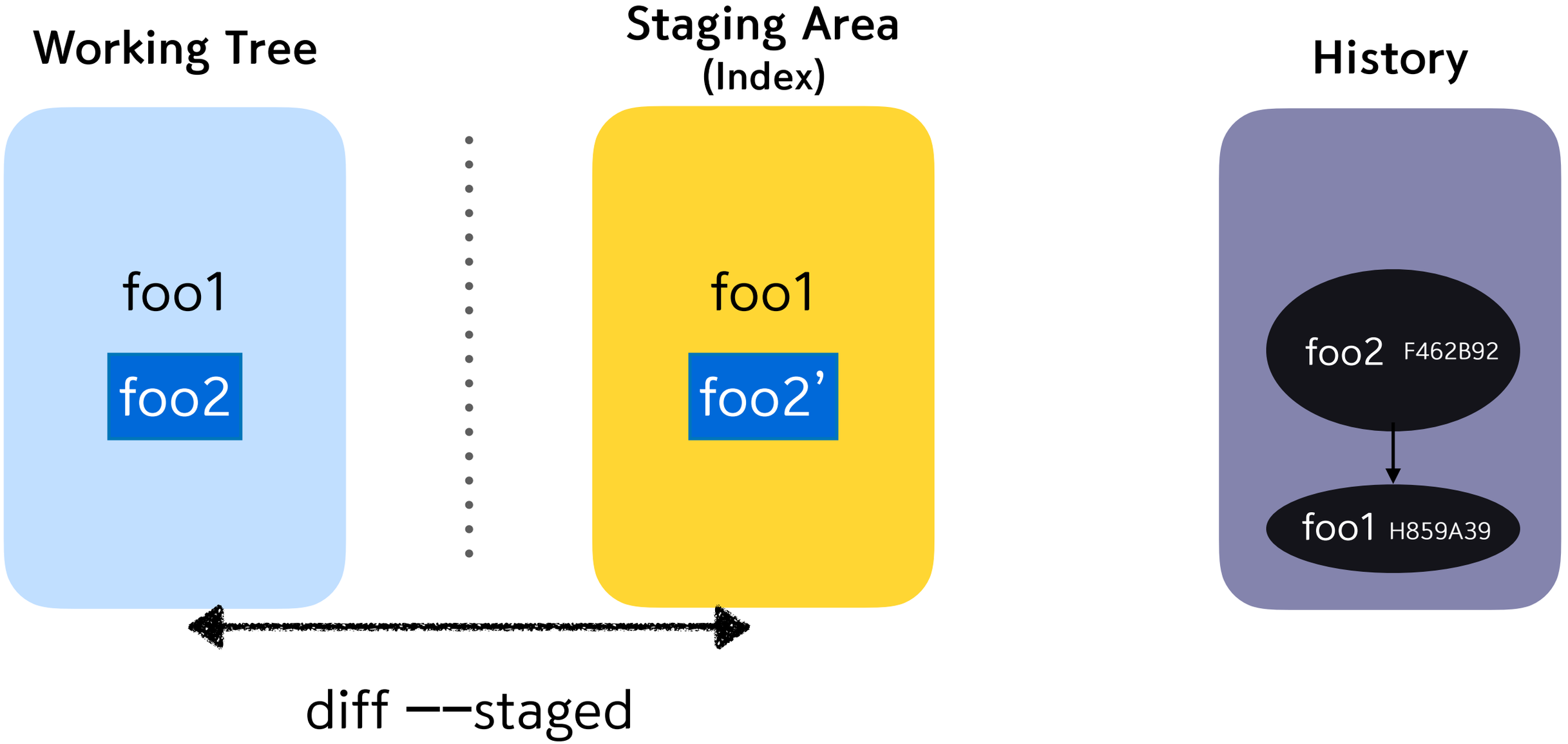
저장소



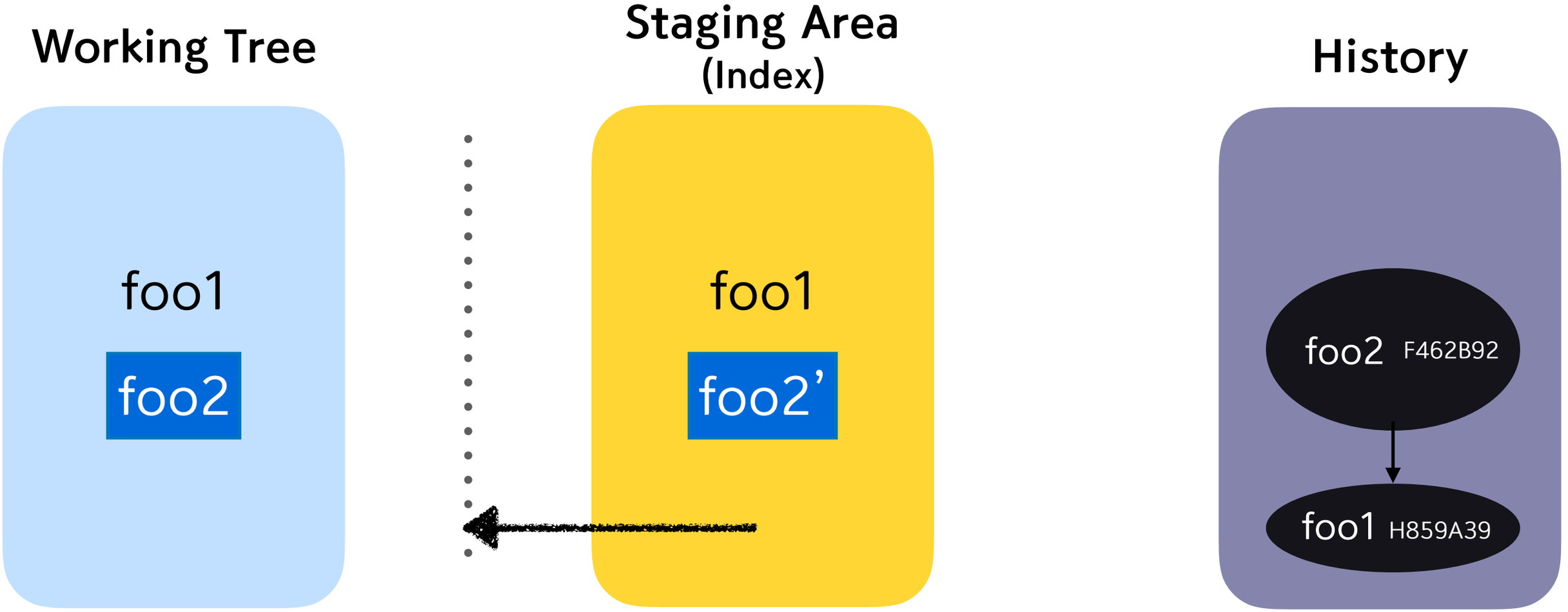
checkout —— foo2

(수정내역 되돌리기)

저장소


Working Tree
foo1
foo2
Staging Area
(Index)
foo1
foo2’
History
foo2 F462B92
foo1 H859A39
diff ––staged

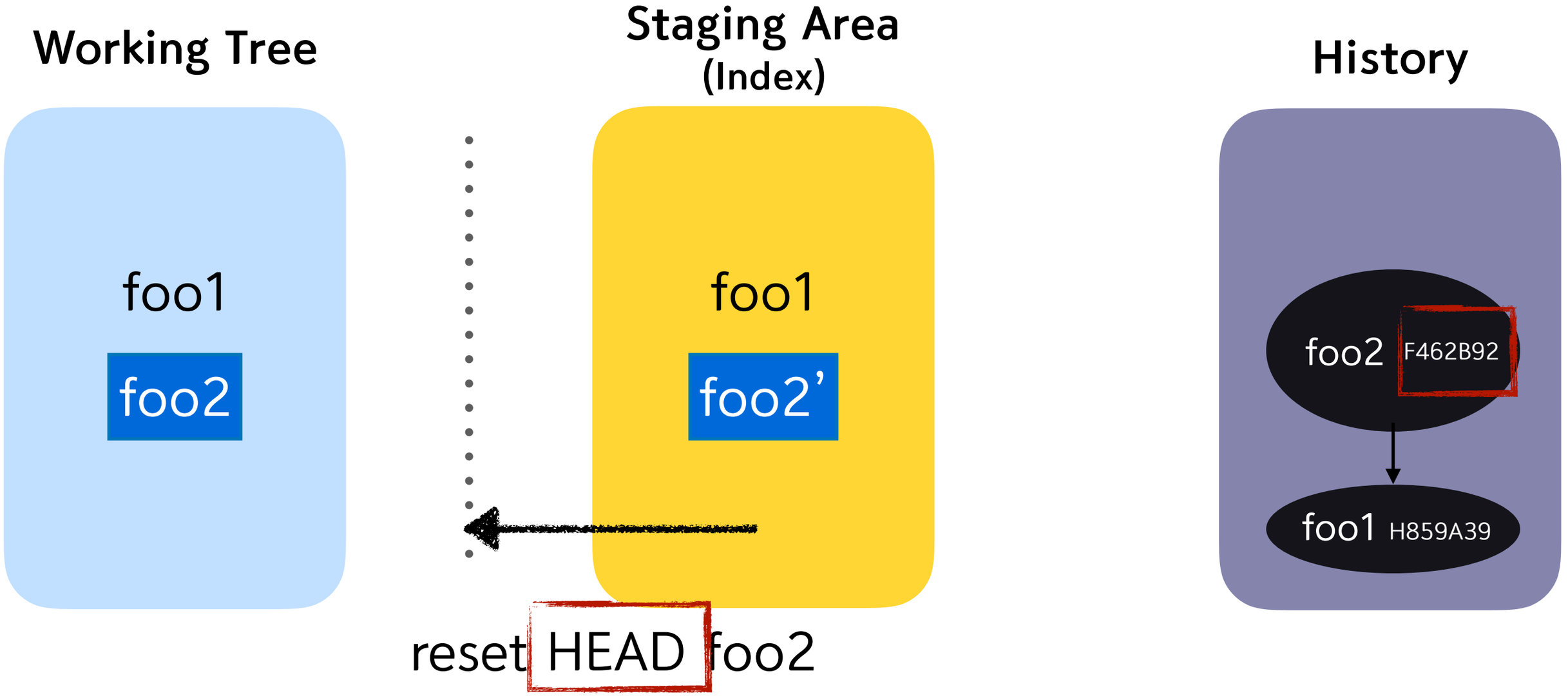
저장소



reset HEAD foo2

(add 후 unstaging, 작업 내역 남아있음)

저장소



(add 후 unstaging, 작업 내역 남아있음)

add / commit

foo1 파일 1개 staging

```
git add foo1
```

수정된 모든 파일 staging

```
git add .
```

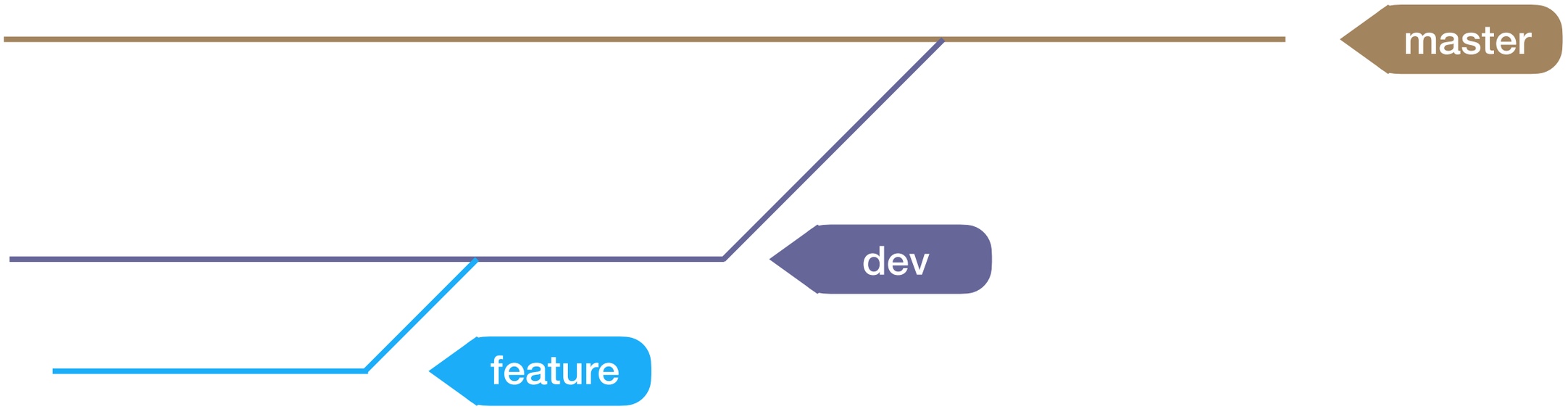
편집기에 들어가지 않고 “메세지”로 바로 커밋하기

```
git commit –m “message”
```

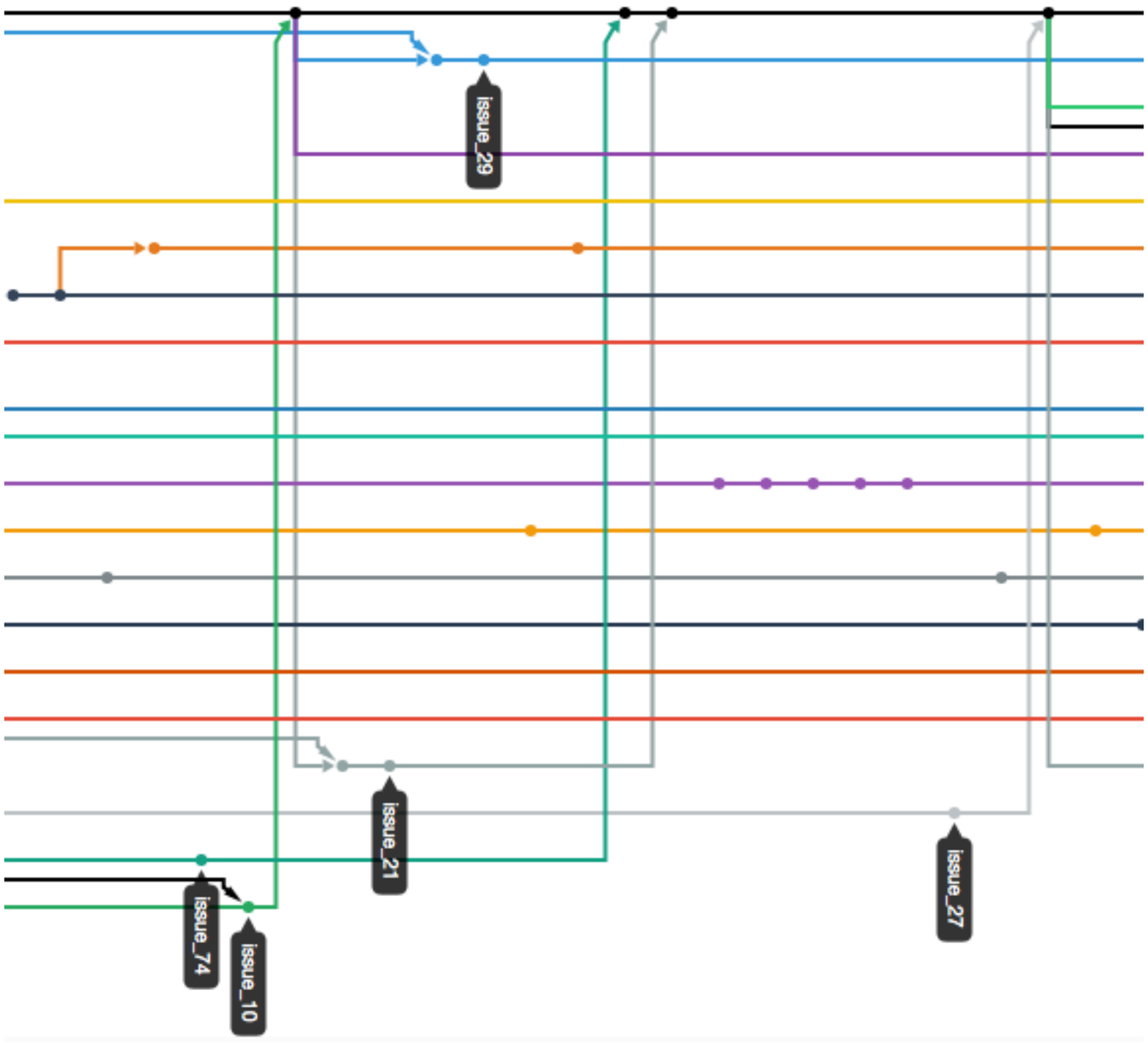
수정된 모든 파일 staging 하고, “메세지”로 한번에 커밋하기

```
git commit –a –m “message”
```

브랜치



브랜치

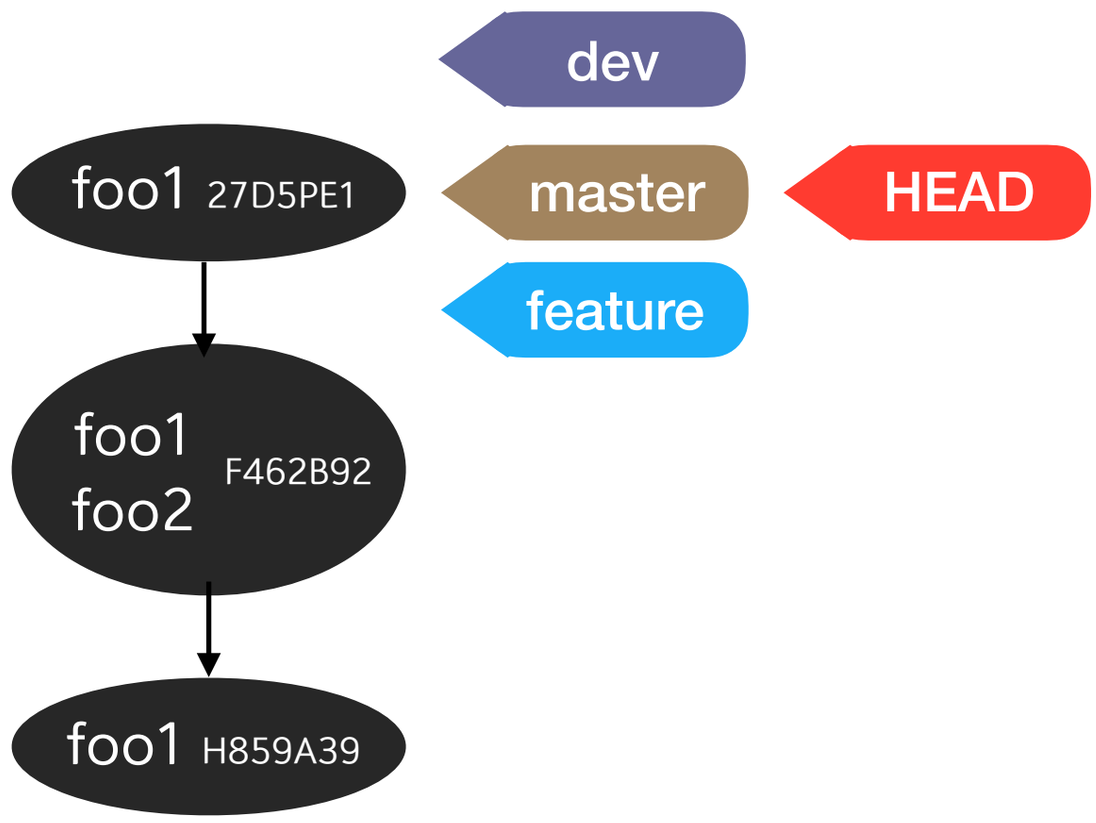


생성

```
git branch dev

git branch feature
```

목록

```
git branch
```



git log --all --decorate --oneline --graph

alias gloga=“git log --all --decorate ––oneline --graph”

생성

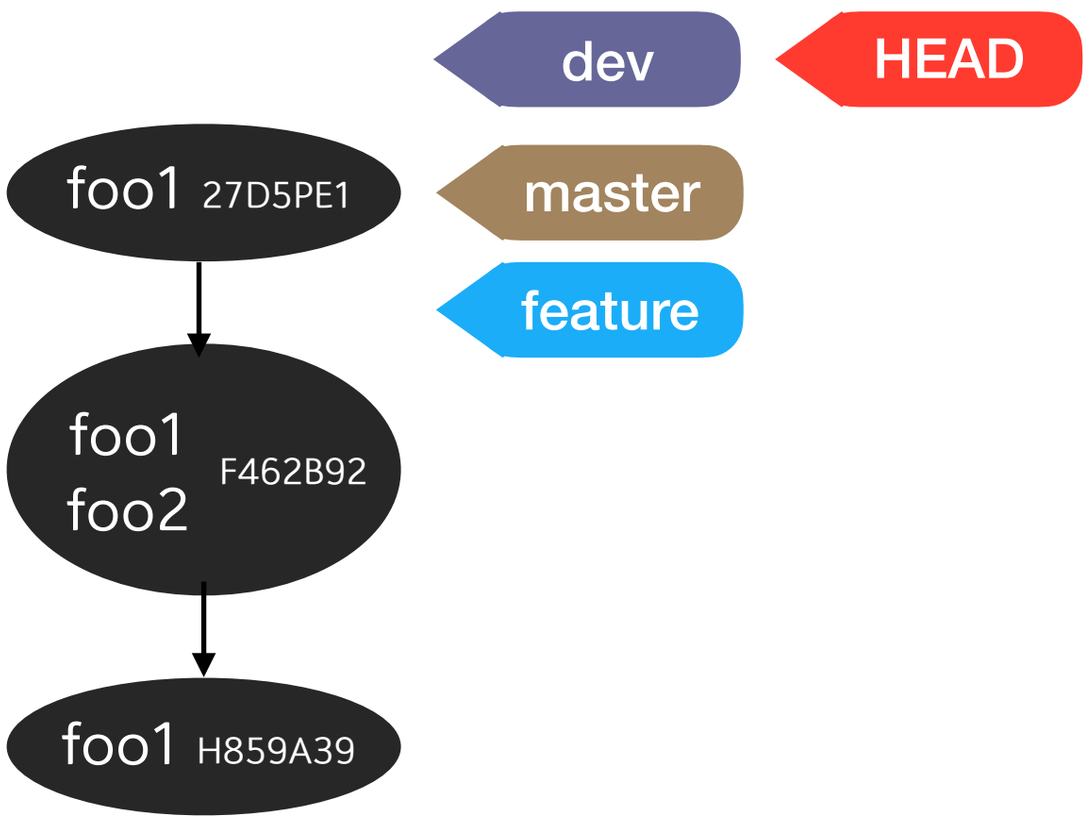
```
git branch dev
```

이동

```
git checkout dev
```

생성 후 바로 이동

```
git checkout -b dev
```

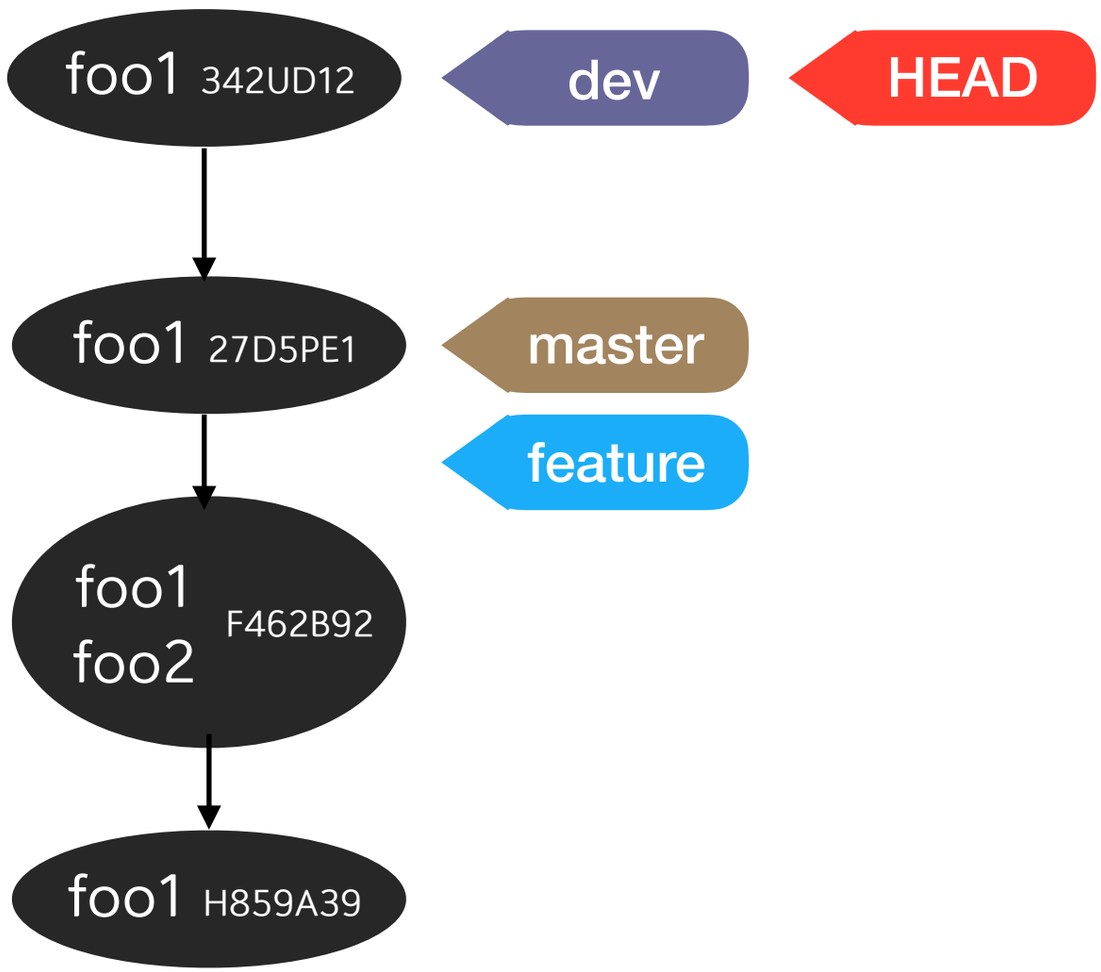


dev 브랜치 - foo1 파일 수정

checkout

add

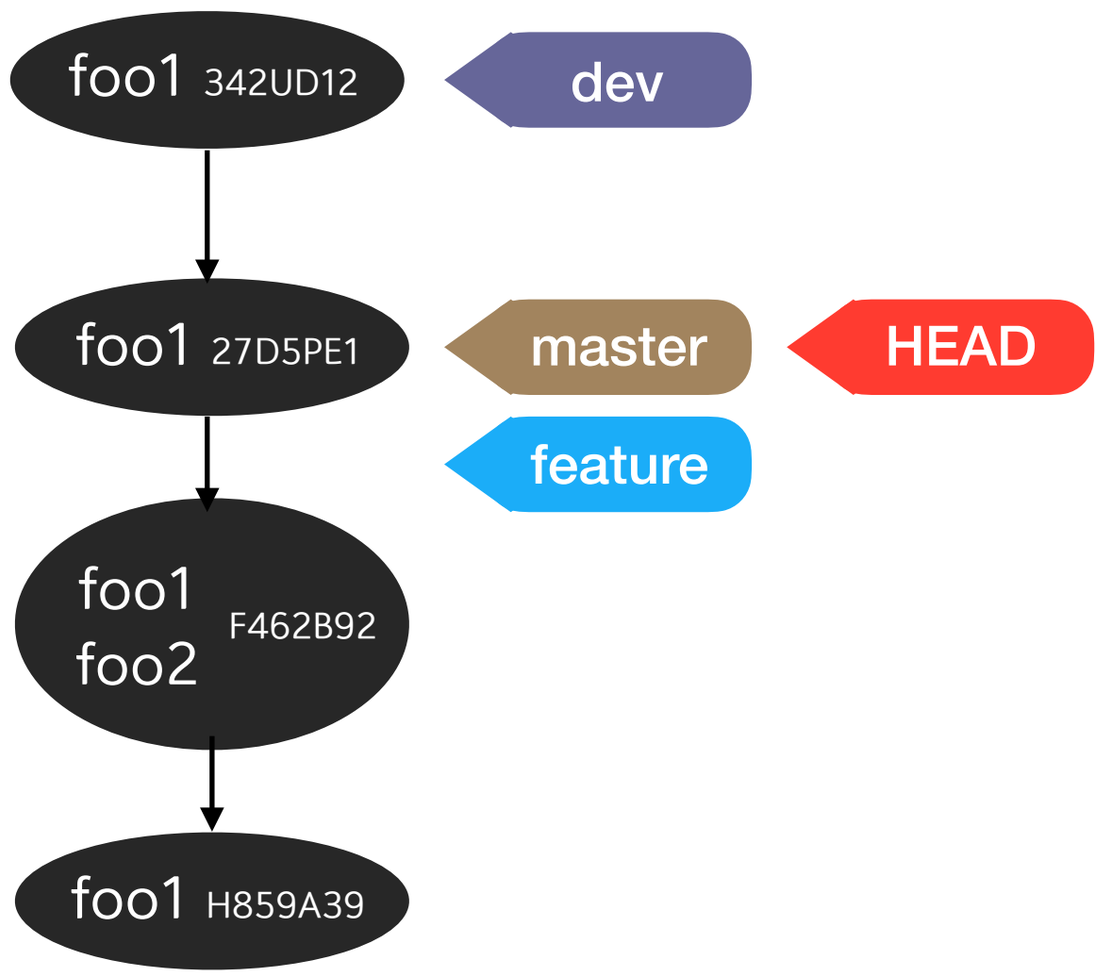
commit



master 브랜치 - dev 작업내역 합치기

이동

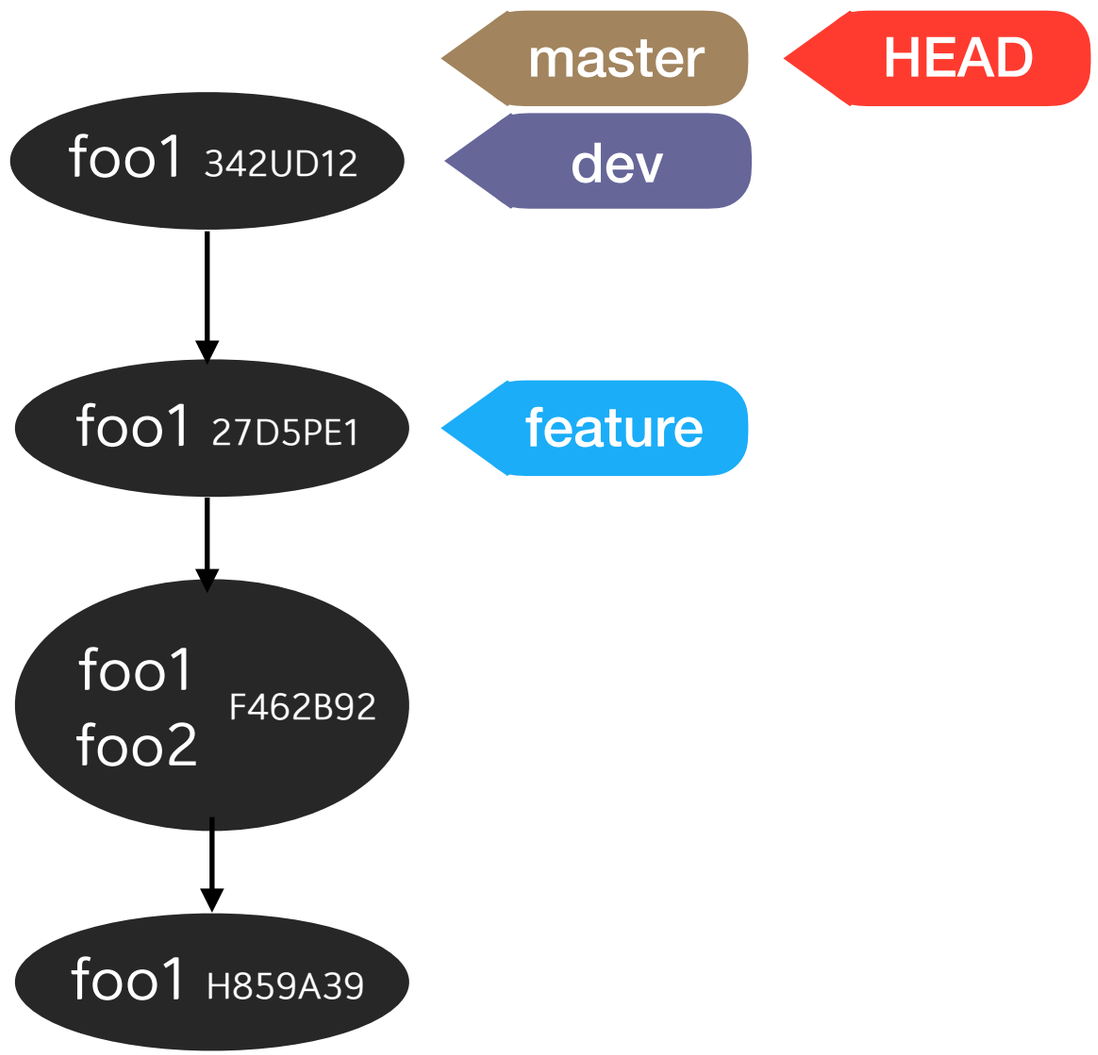
```
git checkout master
```



master 브랜치 - dev 작업내역 합치기

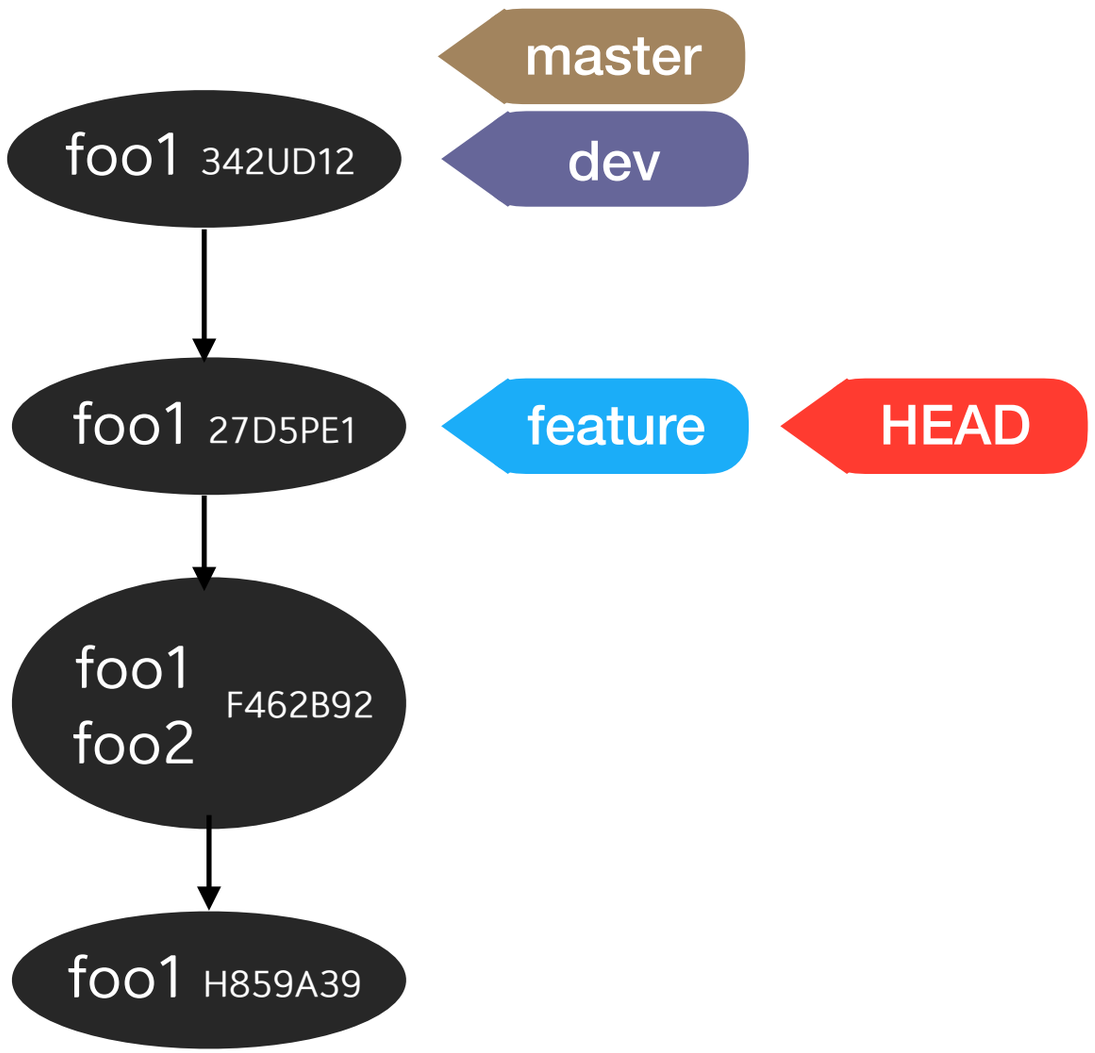
병합

```
git merge dev
```



feature 브랜치 - foo1 파일 수정

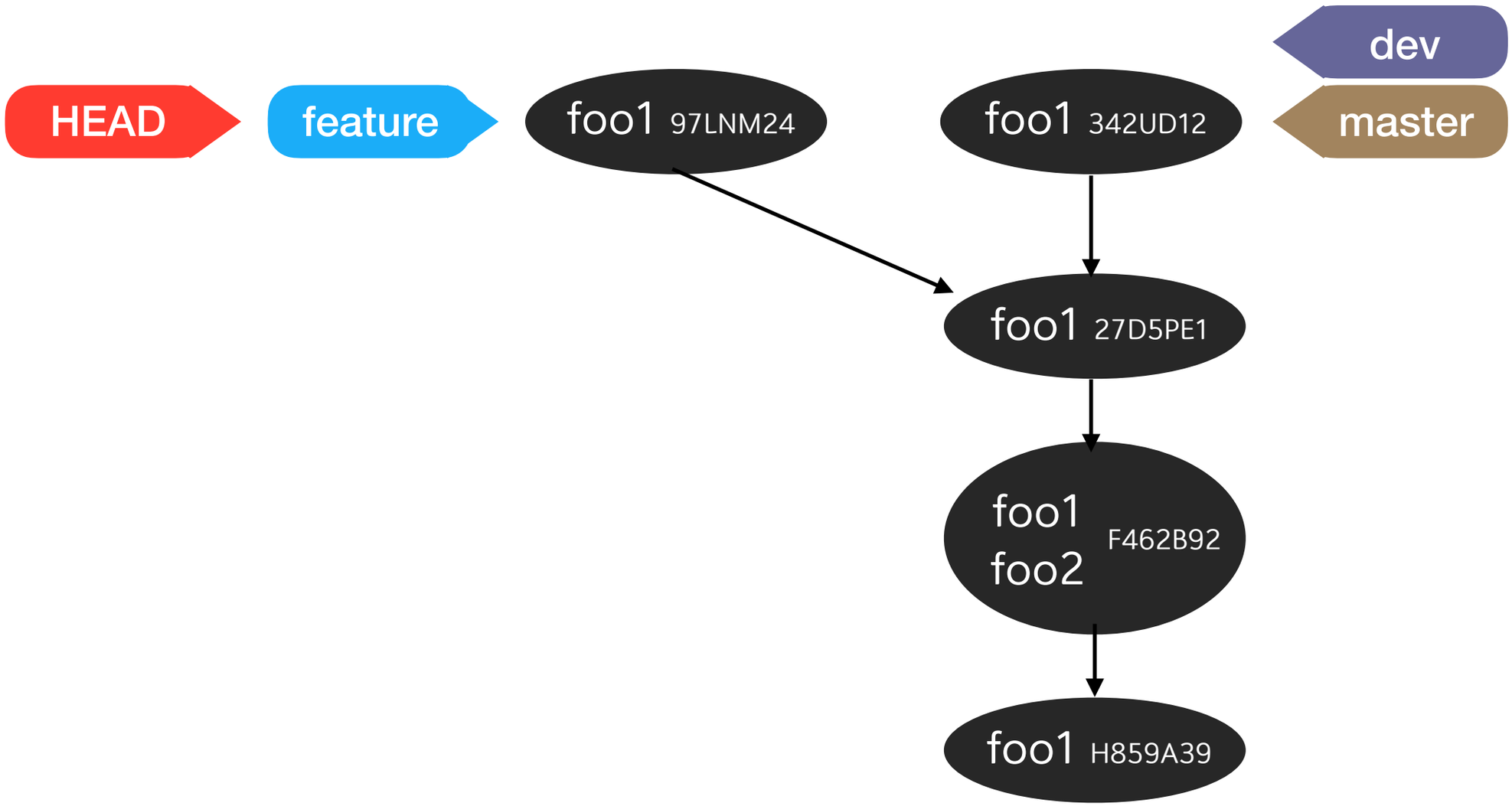
checkout feature



feature 브랜치 - foo1 파일 수정

add

commit

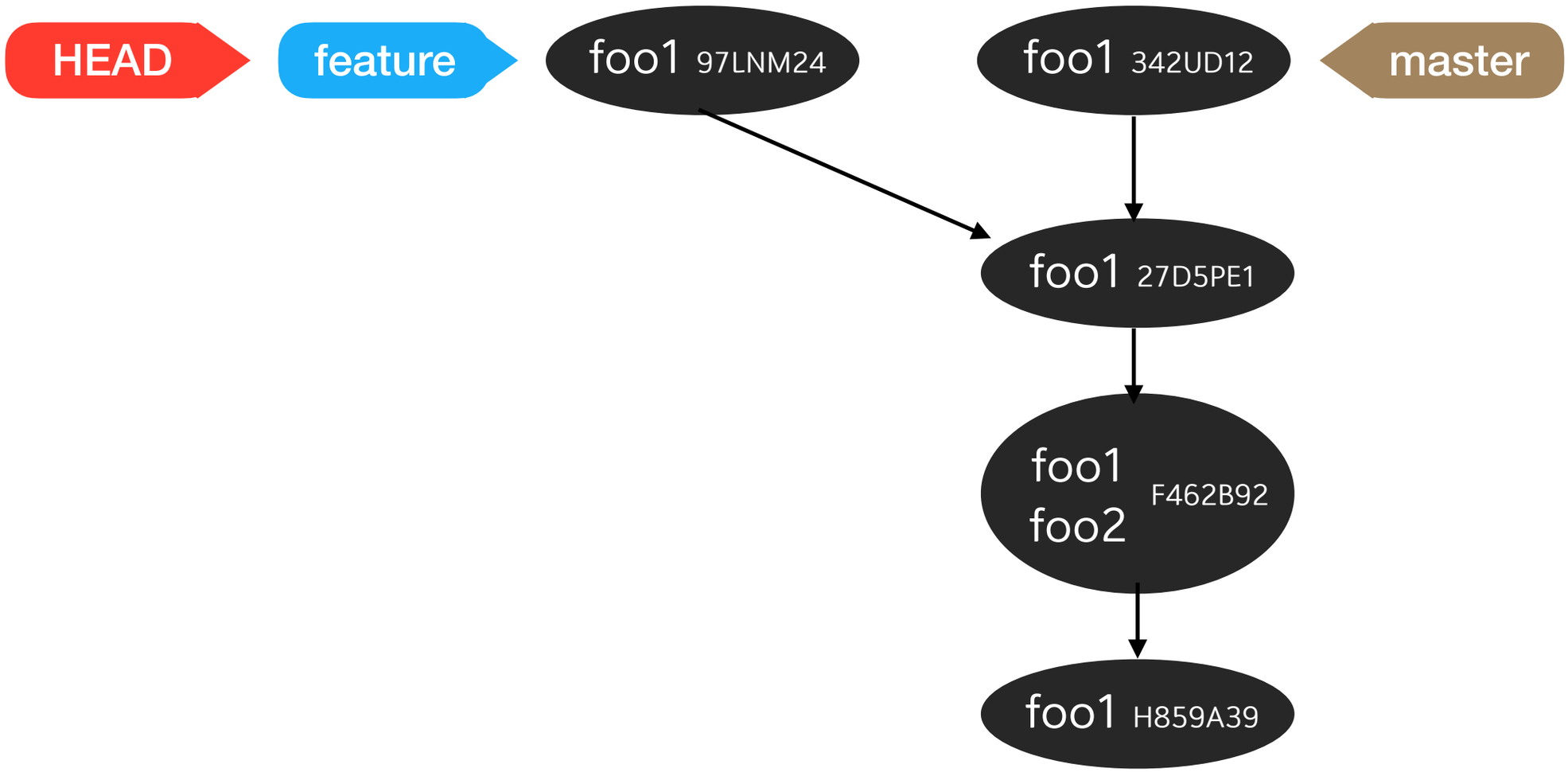


확인

```
git branch --merged
```

삭제

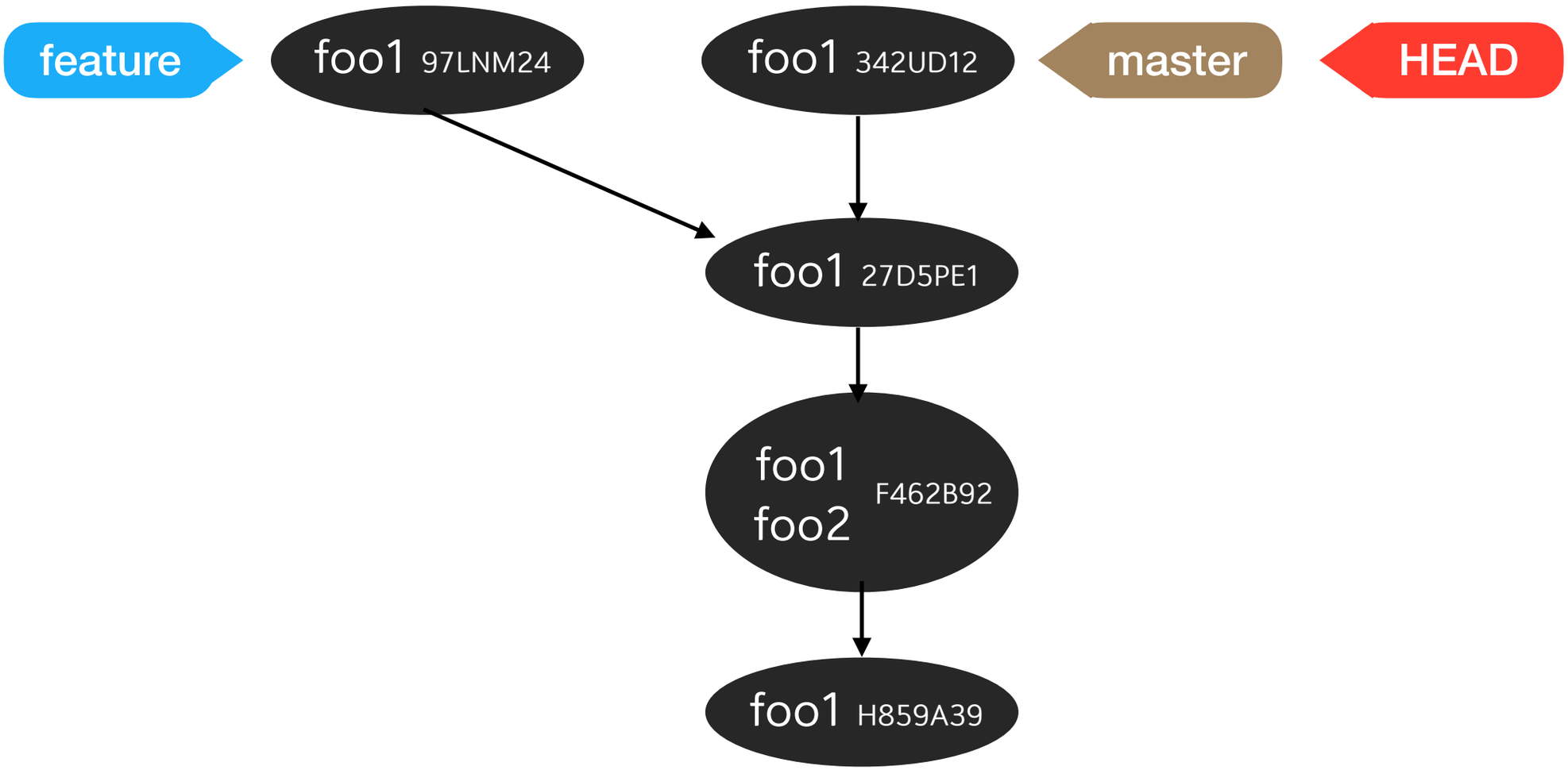
```
git branch -d dev
```



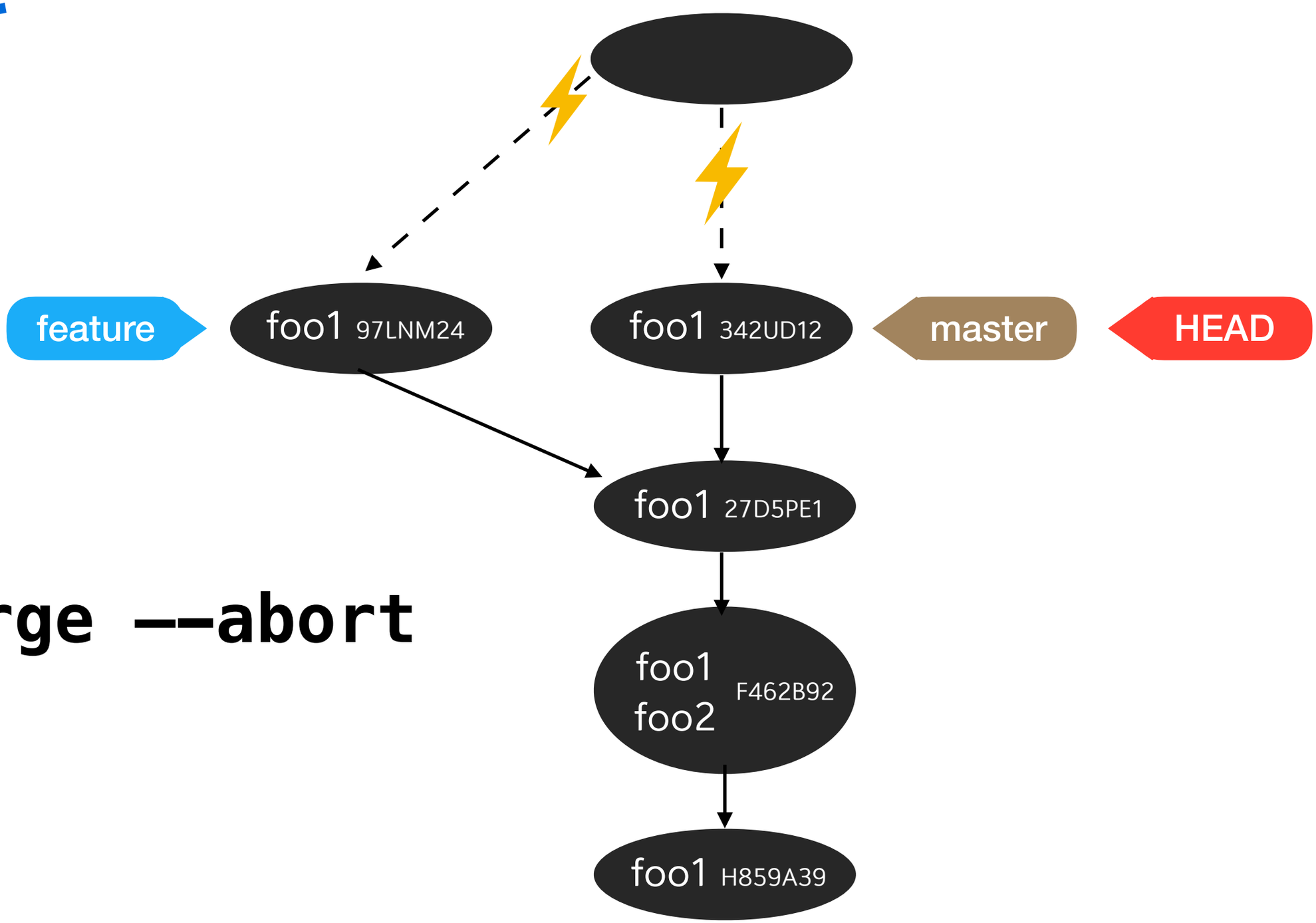
master 브랜치 - feature 작업내역 합치기

병합

```
git merge feature
```



충돌



병합 취소

```
git merge --abort
```

충돌

<<<<<<<<<<<<<< HEAD

==============

>>>>>>>>>>>>>>> feature

* <, =, > 문자는 사람이 인식하기 위한 표시이므로 충돌 해결 후 삭제!

branch

branch 이름 변경

```
git branch -m <oldbranch> <newbranch>
```

branch grouping

```
group1/foo group2/foo
group1/bar group2/bar
```

grouping branch 목록 확인

```
git branch --list “group1/*”
git branch --list “*/foo”
```

https://github.com/chrisjlee/git-style-guide#naming

git remote

*https://cdn-images-1.medium.com/max/800/1*7HpNPipvf7O_92JrJYOmcA.png*

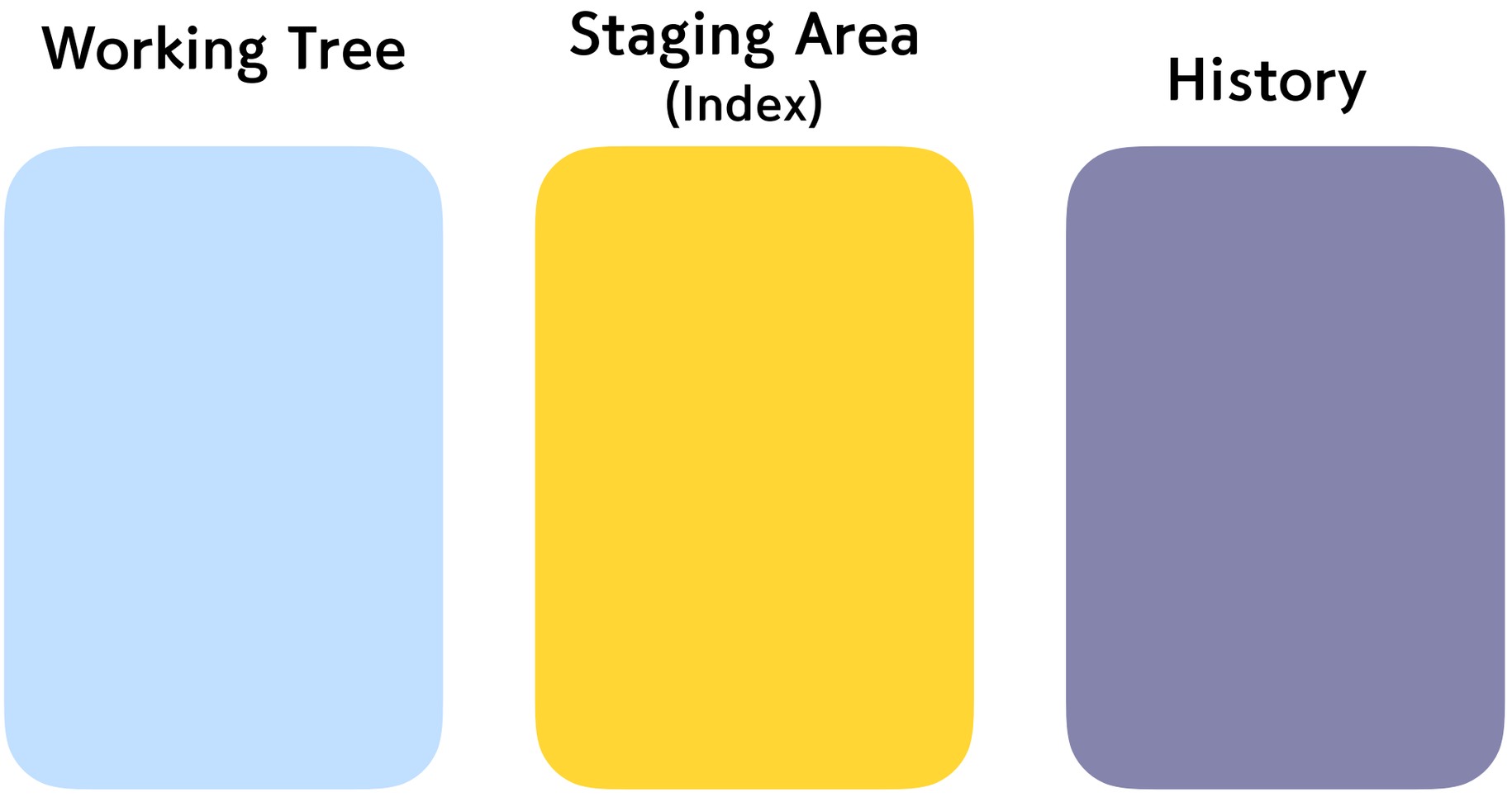
저장소

Working Tree

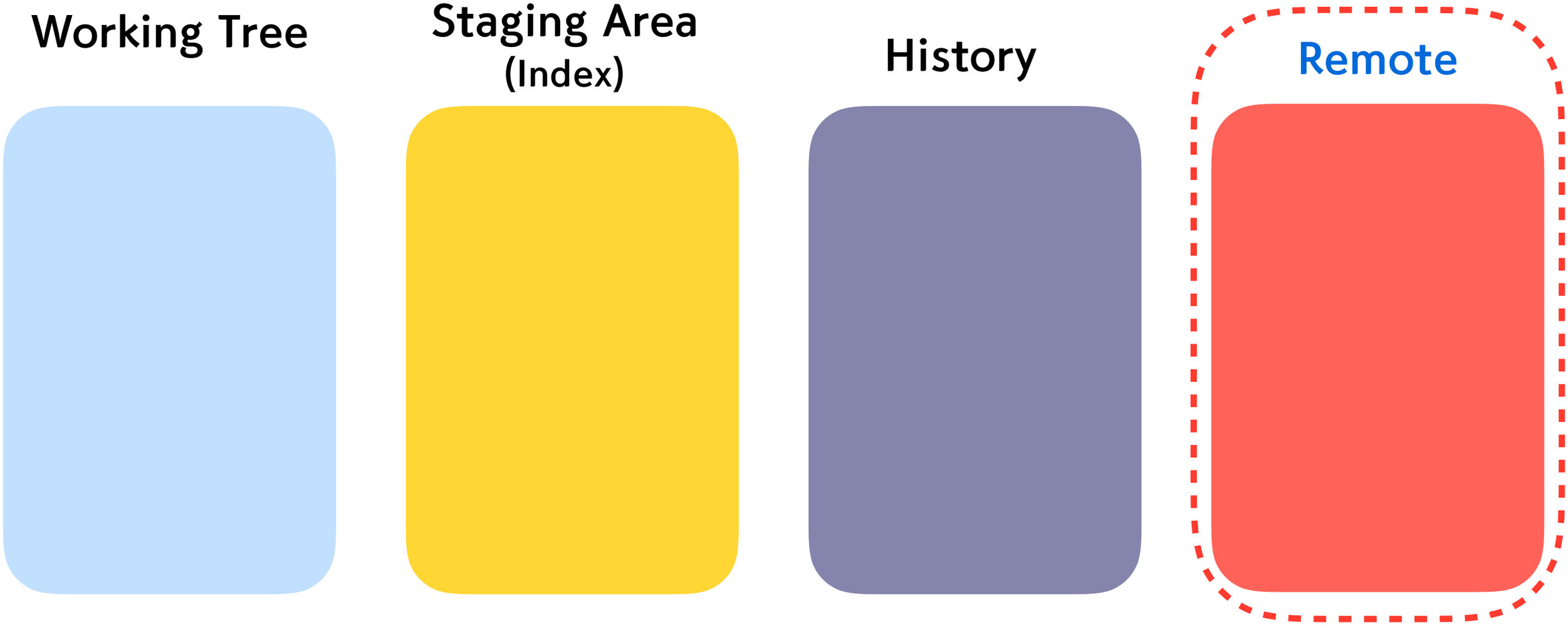
git init

git init

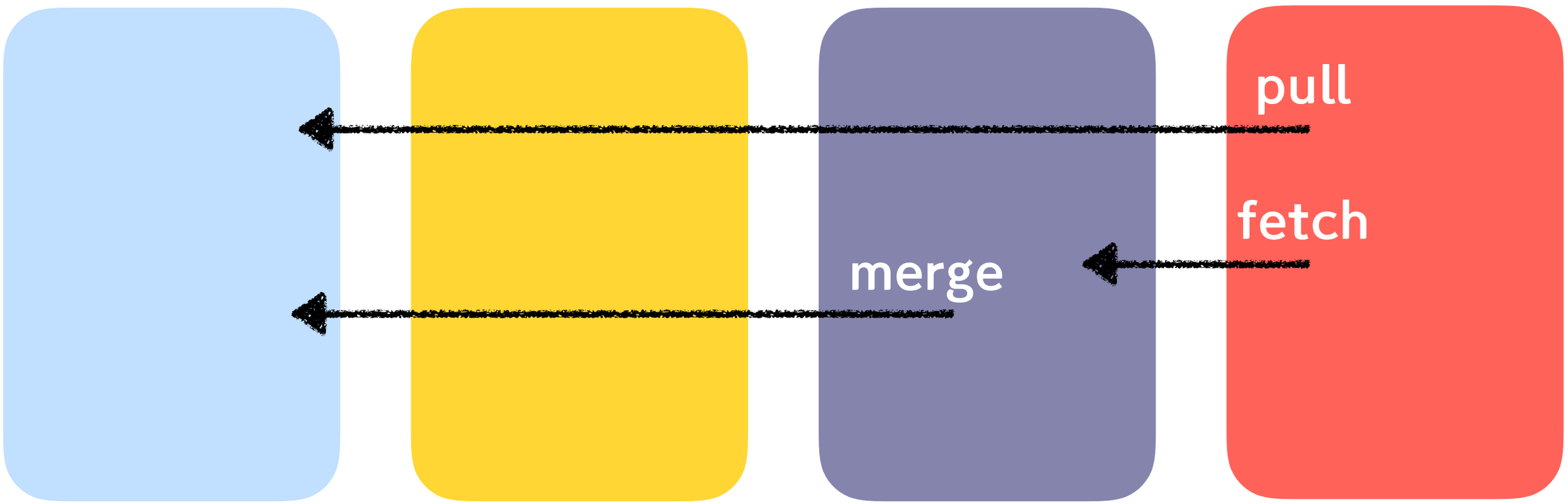
원격



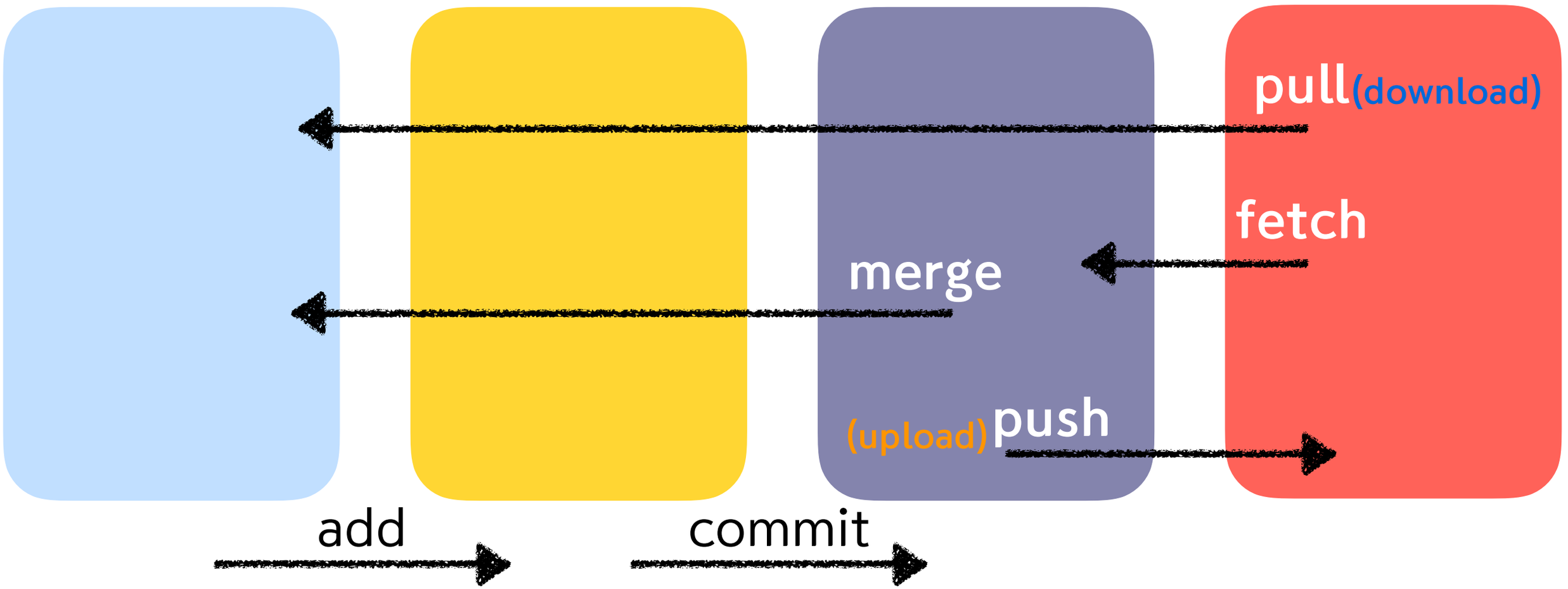
원격



원격


Working Tree
Staging Area
(Index)
History
Remote
pull
fetch
merge

로컬/원격


Working Tree
Staging Area
(Index)
History
Remote
pull(download)
fetch
merge
(upload) push
add
commit

설정

```
git config —-list

git config —-global user.name “name”

git config —-global user.email ”email”

git config —-global —- list

git help <verb>
```

https://gitlab.com

원격

연결

```
git remote add origin <url>
```

```
git push -u origin master
```

--set-upstream

remote

remote list 확인

```
git remote
```

remote(ex. upstream) 삭제

```
git remote remove upstream
```

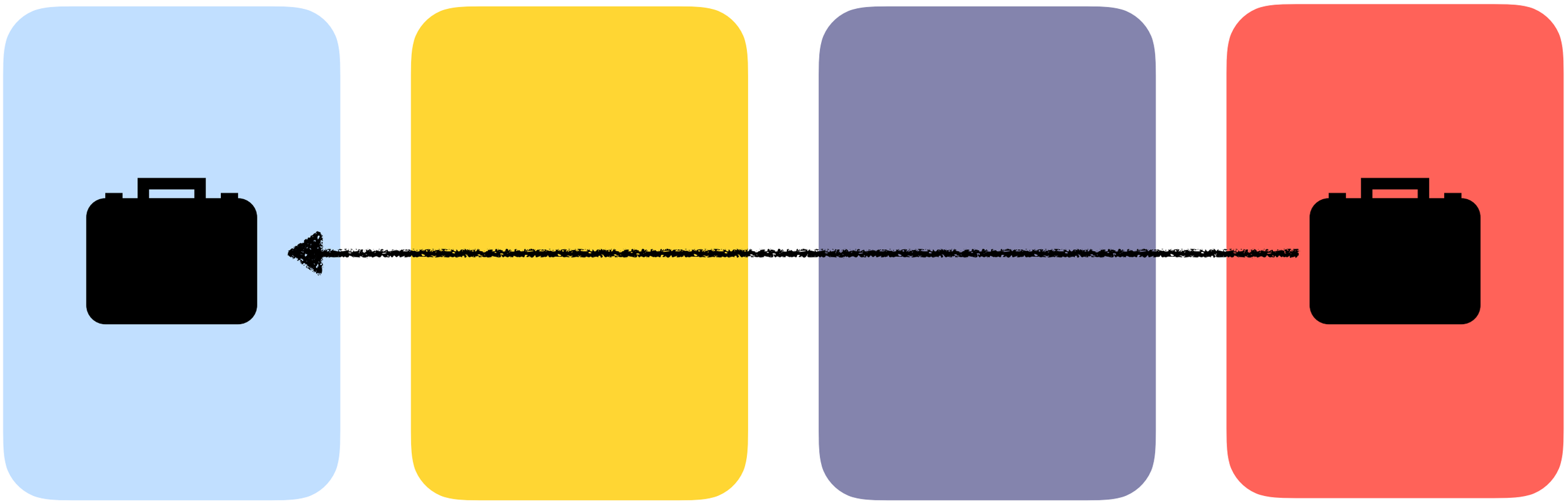
remote branch 확인(로컬과 연결된 remote branch만 보여짐)

```
git branch -r
```

remote branch(ex. issue_1)삭제

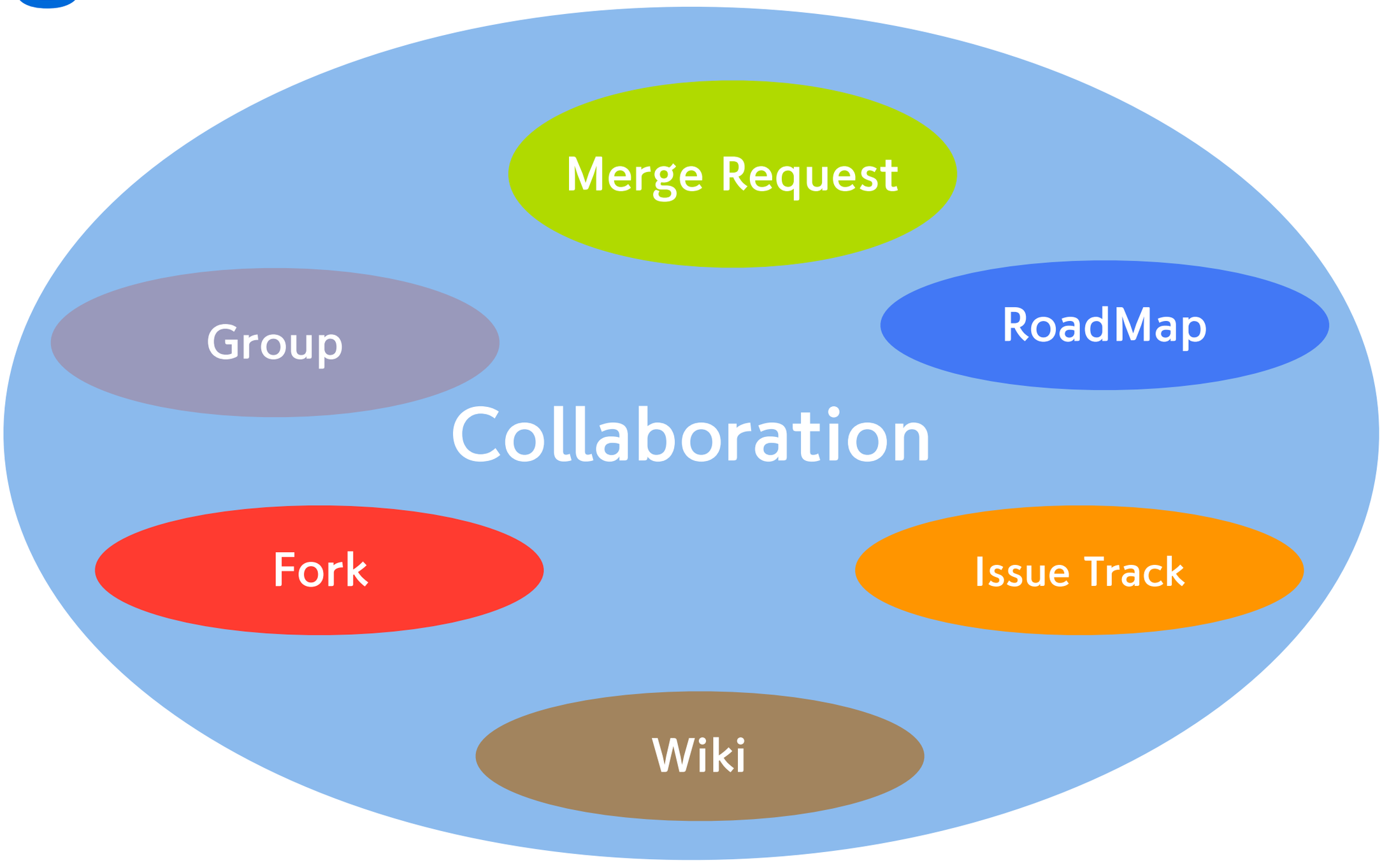
```
git push origin —-delete issue_1
git push origin :issue_1
```

clone




```
git clone <url>
```

gitlab


Merge Request
Group
RoadMap
Collaboration
Fork
Issue Track
Wiki


GitLab

```
remote:
remote: A default branch (e.g. master) does not exist for example/project.git
remote: Ask the project Owner or Maintainer to create a default branch.
remote:
remote:     https://gitlab.com/example/project/project_members
remote:
To gitlab.com:example/project.git
 ! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'git@gitlab.com:example/project.git'
```

```
remote:
remote: A default branch (e.g. master) does not exist for example/project.git
remote: Ask the project Owner or Maintainer to create a default branch.
remote:
remote:     https://gitlab.com/example/project/project_members
remote:
To gitlab.com:example/project.git
 ! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'git@gitlab.com:example/project.git'
```

Project -> Settings -> Repository -> Protected Branches

Protected branch (1)	Last commit	Allowed to merge	Allowed to push	
master default	f25b571e 13 minutes ago	Maintainers	Maintainers	Unprotect

Project members permissions

NOTE: **Note:** In GitLab 11.0, the Master role was renamed to Maintainer.

The following table depicts the various user permission levels in a project.

Action	Guest	Reporter	Developer	Maintainer	Owner
Download project	✓ (*1*)	✓	✓	✓	✓
Leave comments	✓ (*1*)	✓	✓	✓	✓
View Insights charts **[ULTIMATE]**	✓	✓	✓	✓	✓
View approved/blacklisted licenses **[ULTIMATE]**	✓	✓	✓	✓	✓
View license management reports **[ULTIMATE]**	✓ (*1*)	✓	✓	✓	✓
View Security reports **[ULTIMATE]**	✓ (*1*)	✓	✓	✓	✓
View project code	✓ (*1*)	✓	✓	✓	✓
Pull project code	✓ (*1*)	✓	✓	✓	✓
View GitLab Pages protected by access control	✓	✓	✓	✓	✓
View wiki pages	✓ (*1*)	✓	✓	✓	✓
See a list of jobs	✓ (*3*)	✓	✓	✓	✓

https://gitlab.com/help/user/permissions

alias

```
git config —-global alias.co checkout
git config —-global alias.br branch
```

```
alias git=g
```

.gitignore

https://www.gitignore.io

```
logs/
build/
DerivedData/

## Various settings
*.pbxuser
!default.pbxuser
*.mode1v3

## Other
*.moved-aside
*.xcuserstate
```

git links

git 설치 : 공식사이트

git 명령어 정리 : 다양한 명령어가 깔끔하게 정리되어있어요

누구나 쉽게 이해할 수 있는 git 입문 : 원숭이가 그림으로 쉽게 사용법을 설명해줘요

SVN 능력자를 위한 git 개념 가이드 : SVN과 git의 개념 차이를 그림으로 멋지게 설명했어요

★Pro git 영상 설명(영문) : 천천히, 그리고 완벽하게 git의 기본개념과 사용법을 설명합니다

.git 디렉토리 이해하기(영문) : .git 디렉토리에 대해 좀 더 알고 싶다면 읽어보아요

git links

- github 마크다운 가이드 : 공식 가이드(영문)
- 마크다운 작성법 by.허니몬
- 좋은 git 커밋 메시지를 작성하기 위한 8가지 약속 by. Uno Kim
- Oh My ZSH : 터미널 꾸미기

git 미션

- markdown resume 만들기
- TIL(Today I Learned) 정리하기
- Leran Git Branching 게임 정복하기

관련 링크

이직초보 어느 개발자의 이력서 만들기 by.구인본

[번역] 2017년 개발자 이력서 작성 가이드 by.마르코

github 개발자 이력서 by.Kuu

6개월간의 TIL 회고(꾸준히 하면 좋은 일이 생긴다) by.이현주

command line for mac

ls : 디렉토리 하위 리스트 확인

ls -a : 디렉토리 하위 리스트 확인(숨김 파일 포함)

cd : 특정 디렉토리로 이동

cd .. : 상위 디렉토리로 이동

pwd : 현재 디렉토리 확인

open . : 현재 디렉토리 열기

mkdir : 새로운 디렉토리 생성

touch : 새로운 파일 생성

vi : 파일 편집기 실행(해당 파일명이 없다면 파일 생성)

  i : 편집기 실행했을때 편집모드 실행(insert)

  w : 편집기 수정사항 저장

  q : 편집기 모드 종료

rm : 파일 삭제

clear : command 화면 삭제

미정의

친절한 git 강의

감사합니다

ninevincentg@gmail.com