

자율주행 트랙

자율주행 트랙

데이터베이스

목차 데이터베이스

1. 데이터베이스 기초
2. SQL 기초 I
3. SQL 기초 II
4. SQL 심화
5. Git

Git



효율적인 협업

자율주행 트랙

데이터베이스

Git



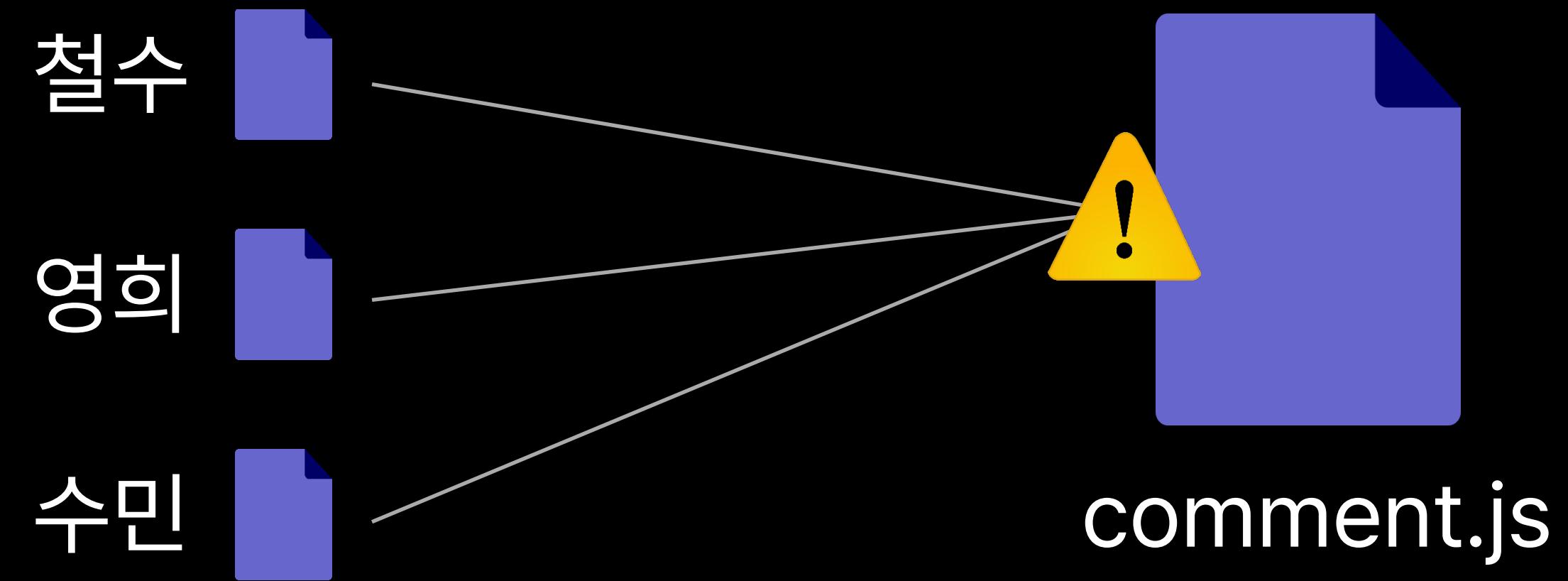


철수

comment.js 오류 사항 수정 했어.

영희

어? 내가 수정한 부분은 다 어디갔지?





comment.js



comment.js.bak



comment.js.bak2



comment.js.bak-180201



comment.js



Git의 특징

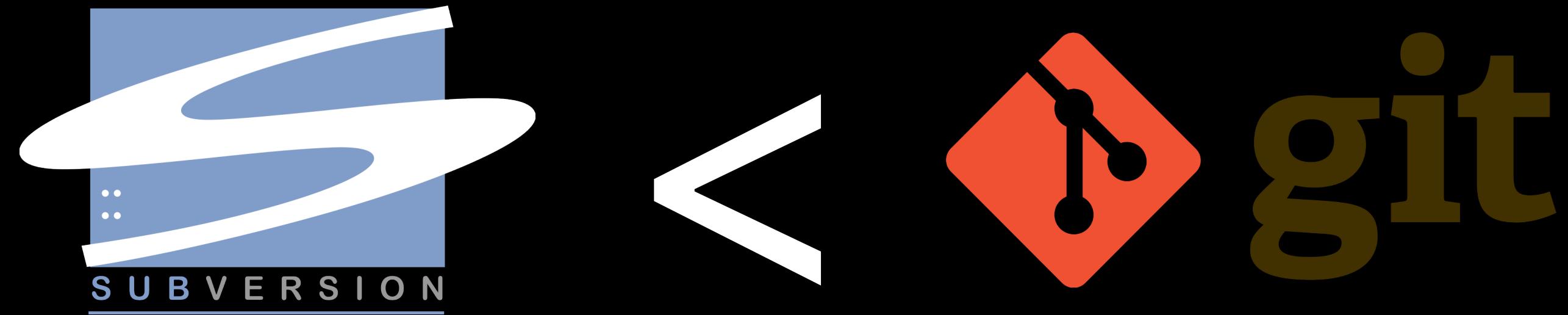
자율주행 트랙

데이터베이스

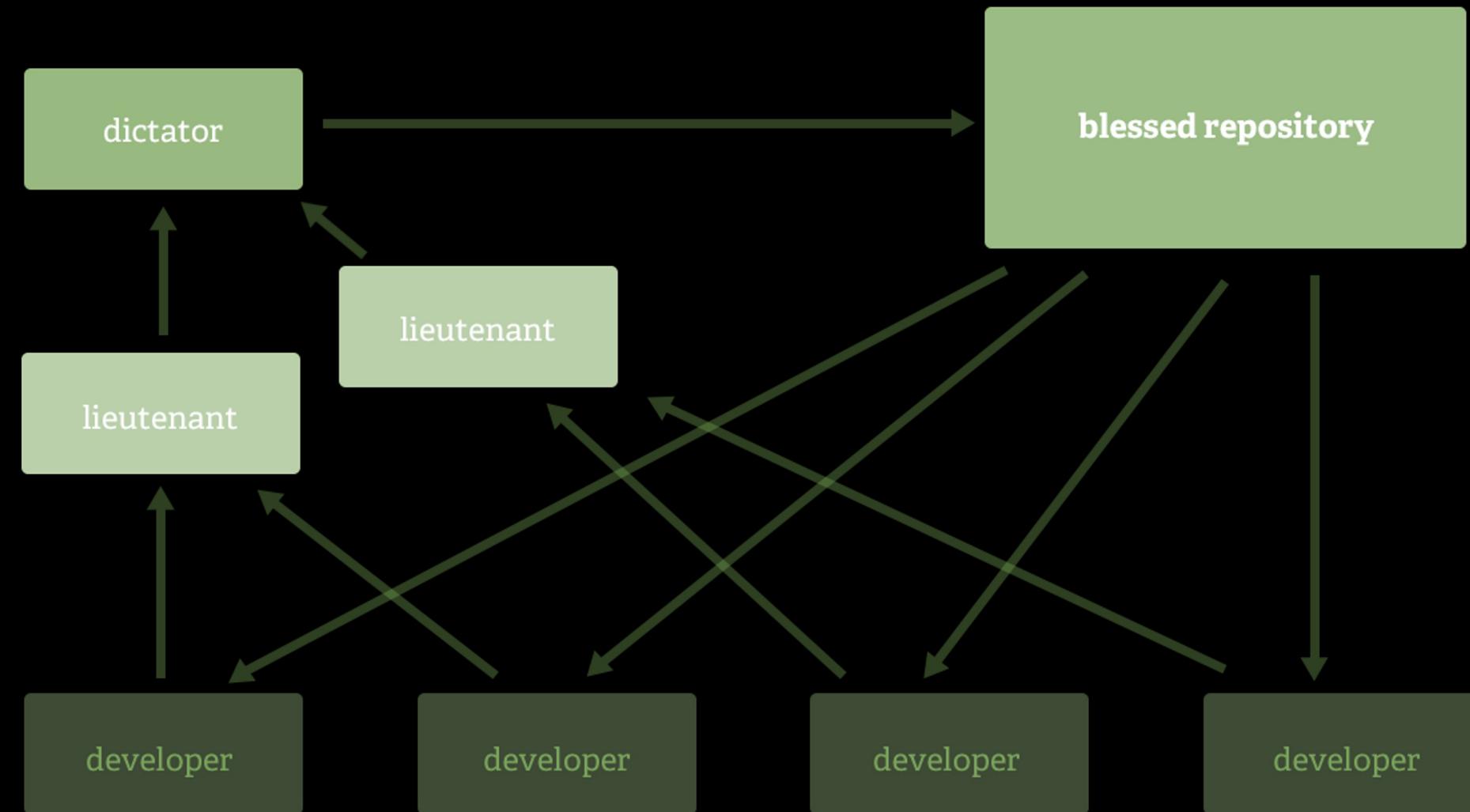
Git



1. 가지 치기와 병합



2. 가볍고 빠르다



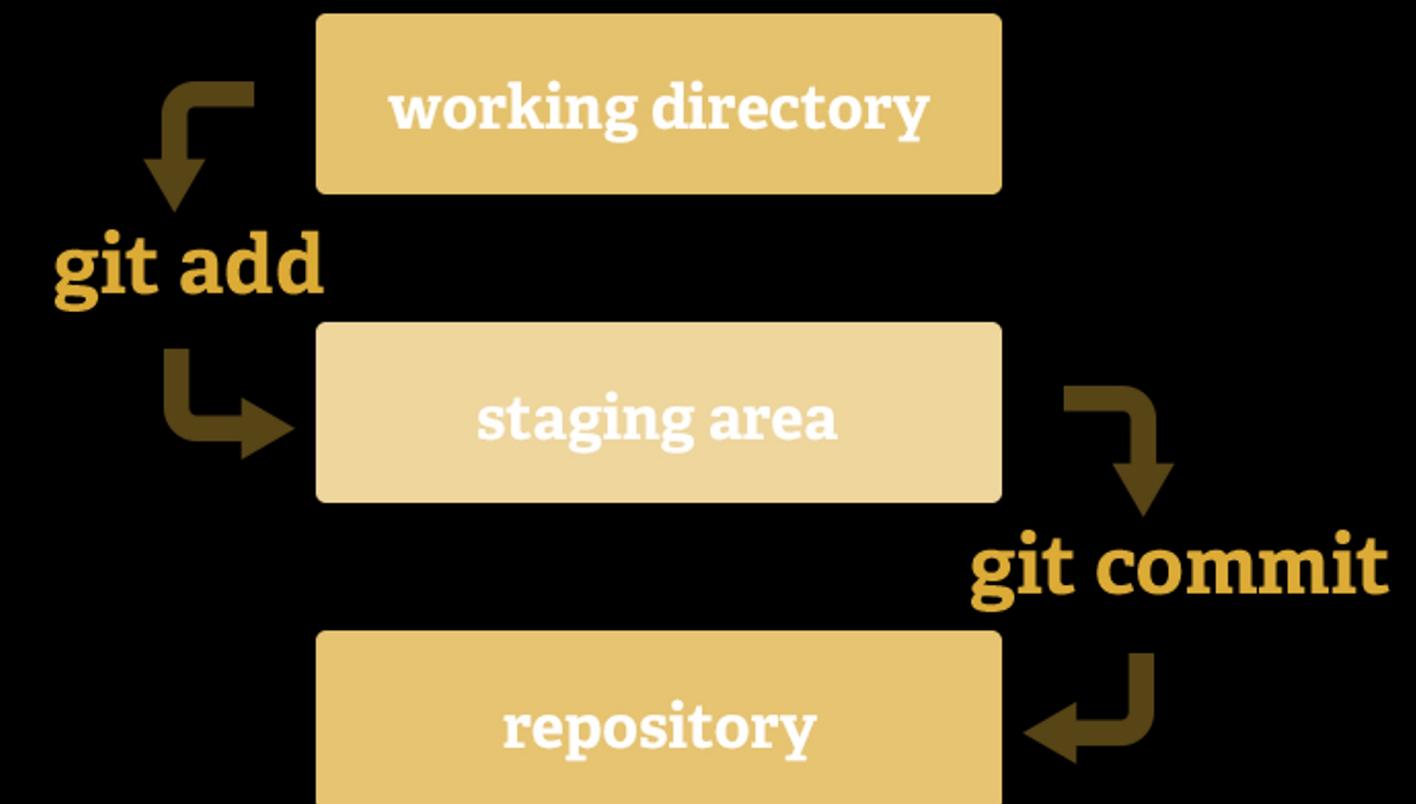
3. 분산 작업



The screenshot shows the GitHub desktop application interface. On the left, there's a sidebar with icons for History, Changes, Branches, and Settings. The main area displays a list of commits for the 'gitscm-next' repository. The commits are grouped by date: June 12, 2012, and June 11, 2012. A red box highlights the commit list for June 12, 2012.

Date	Author	Commit Message	SHA
Jun 12, 2012	Scott Chacon	Revert "refactor helpers to use content_tag/image_tag/link_to"	2575264
Jun 12, 2012	Scott Chacon	Merge pull request #148 from linquize/git-clean	7f760bb
Jun 12, 2012	Scott Chacon	Merge remote-tracking branch 'origin/pull/111/head'	a4fad60
Jun 12, 2012	Scott Chacon	Merge remote-tracking branch 'origin/pull/109/head'	64b93fa
Jun 11, 2012	Jason Smith	tidy up icon positioning	e3152a4
Jun 11, 2012	Jason Long	Merge pull request #147 from mmozuras/doc-controller-tests	0e42c69
Jun 11, 2012	Jason Long	Merge pull request #146 from mmozuras/gitignore	1e71eb1

4. 데이터 보장



5. 준비 영역 (Staging area)



6. 오픈 소스





- Linux 또는 MacOS 환경에서는 대부분 이미 Git이 설치되어 있습니다.
- 다음 페이지의 화면을 참고하여 Terminal 실행 후 Git 명령어를 실행해 보세요.
- Windows 환경의 경우 다음 단계로 넘어가 주세요.



Git 설치 - 1. 설치 여부 확인

자율주행 트랙

데이터베이스

Git

```
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
```

These are common Git commands used in various situations:

...

Terminal을 실행하고, `git` 입력 후 엔터를 눌러주세요.

(위와 같이 실행되지 않는다면 다음 단계에서 Git을 설치해 주세요.)



- 아래의 사이트 접속 후 본인에게 맞는 설치 파일을 다운로드 받습니다.
 - <https://git-scm.com/downloads>
 - 다운로드 된 설치 파일을 실행합니다.



- Linux 또는 MacOS: Terminal 실행
- Windows: 시작 메뉴 → Git → Git Bash 실행



Git 설치 – 4. 설치 확인

자율주행 트랙

데이터베이스

Git

```
$ git --version  
git version 2.39.3 (Apple Git-146)
```

Git이 정상적으로 설치되었나요? 위 명령어를 입력하여 Git 버전을 확인해 보세요.
(정확한 버전은 위와 달라도 상관 없습니다.)

```
$ git config --global user.name "elice"  
$ git config --global user.email gitaccount@elice.com
```

저장소에 코드를 반영할 때 등록될 사용자 정보를 설정합니다.

```
$ git config user.name "elice"  
$ git config user.email gitaccount@elice.com
```

프로젝트마다 다른 사용자 정보를 지정하고 싶으면,
저장소 생성 후 `--global` 옵션을 빼고 실행해 주세요.



Git 초기 설정 - 2. 설정 정보 확인

자율주행 트랙

데이터베이스

Git

```
$ git config --list  
credential.helper=osxkeychain  
user.name=elice  
user.email=gitaccount@elice.com  
...
```

위 명령을 실행하여 앞에서 설정한 내용을 확인해 보세요.



Git 저장소 생성

자율주행 트랙

데이터베이스

Git

```
$ git init
```

기존의 디렉토리를 git repository로 설정합니다.



```
$ git init  
Initialized empty Git repository in /Users/elice/git_test/.git/
```

Git을 사용할 프로젝트 폴더로 이동 후
위 명령어를 실행해 주세요.



Git 저장소 생성

자율주행 트랙

데이터베이스

Git

```
$ ls -al
drwxr-xr-x  3 elice  staff  96 11 11 21:45 .
drwxr-xr-x+ 5 elice  staff 160 11 11 21:45 ..
drwxr-xr-x 10 elice  staff 320 11 11 21:45 .git
```

프로젝트 디렉토리에 `.git` 디렉토리가 생성되며
저장소 생성이 완료되었습니다. 간단하죠?



새로운 파일 생성

자율주행 트랙

데이터베이스

Git

저장소 생성 완료 후,
새로운 `comment.js` 파일 작업을 완료하였습니다.
이 파일을 저장소에 어떻게 반영할 수 있을까요?

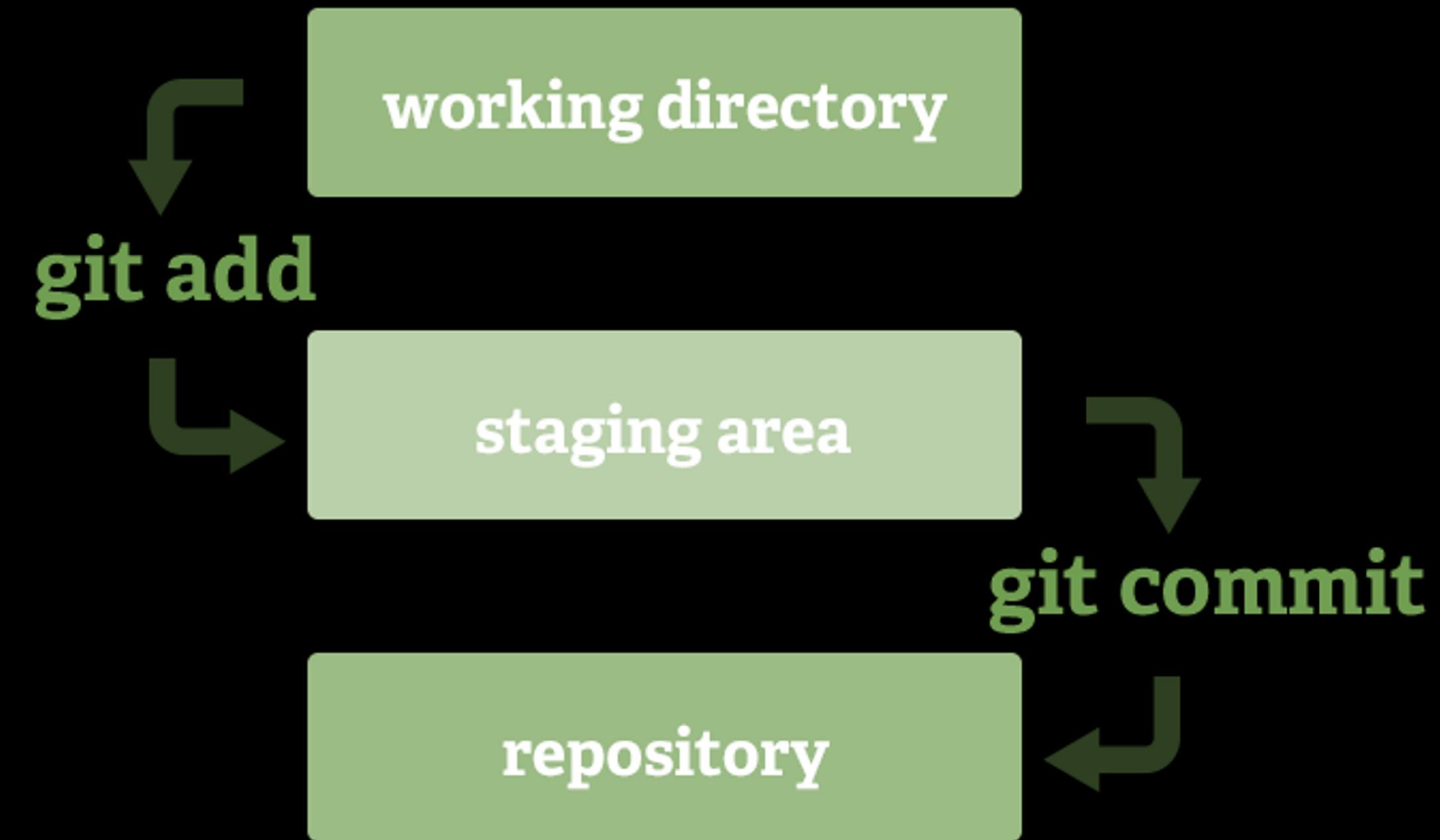


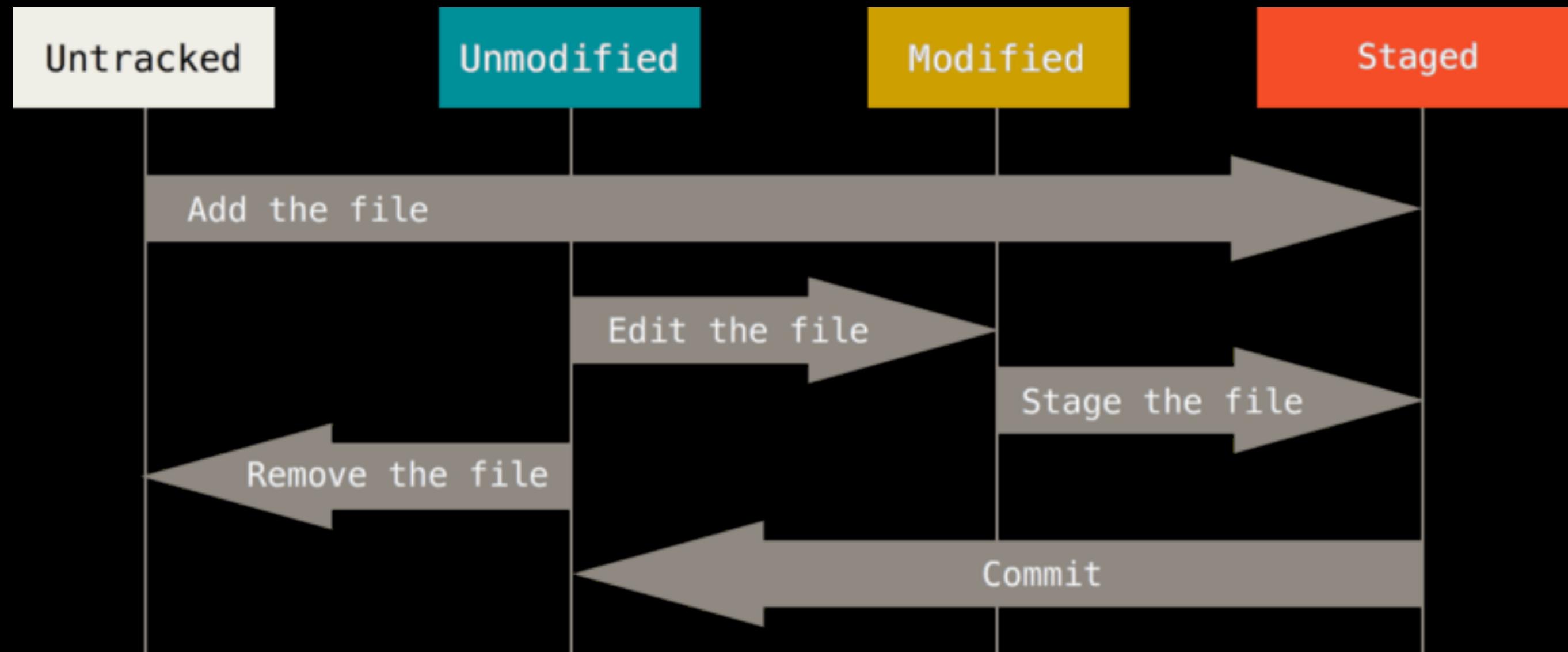
파일 영역의 라이프 사이클

자율주행 트랙

데이터베이스

Git







새로운 파일 생성 (1)

자율주행 트랙

데이터베이스

Git

```
$ git add comment.js
```

먼저, `comment.js` 파일을 **준비 영역**으로 보내야 합니다.

`git add` 명령어를 사용합니다.

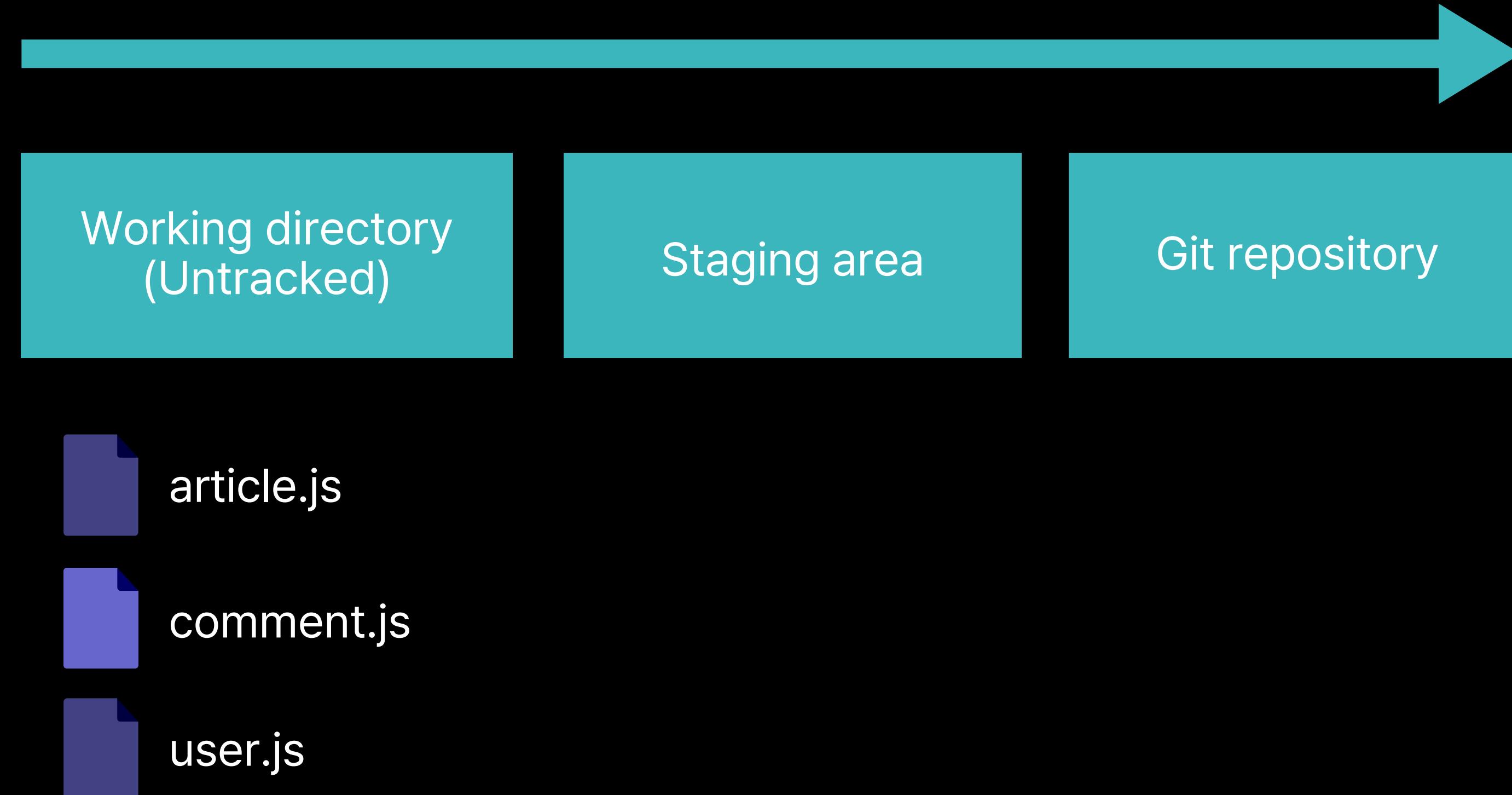


새로운 파일 생성 (1)

자율주행 트랙

데이터베이스

Git



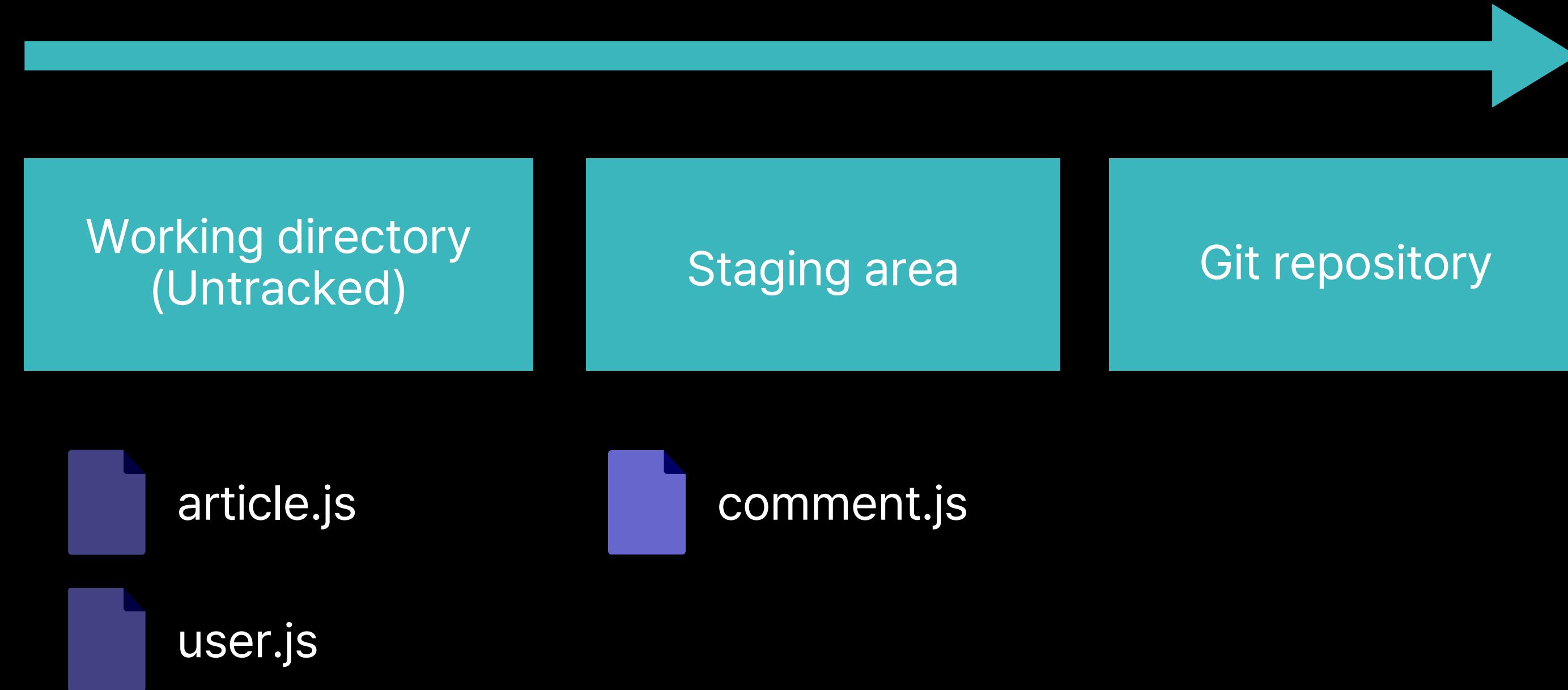


새로운 파일 생성 (1)

자율주행 트랙

데이터베이스

Git





새로운 파일 생성 (2)

자율주행 트랙

데이터베이스

Git

```
$ git add user.js
```

추가할 파일이 있다면 계속해서 더 추가할 수 있습니다.

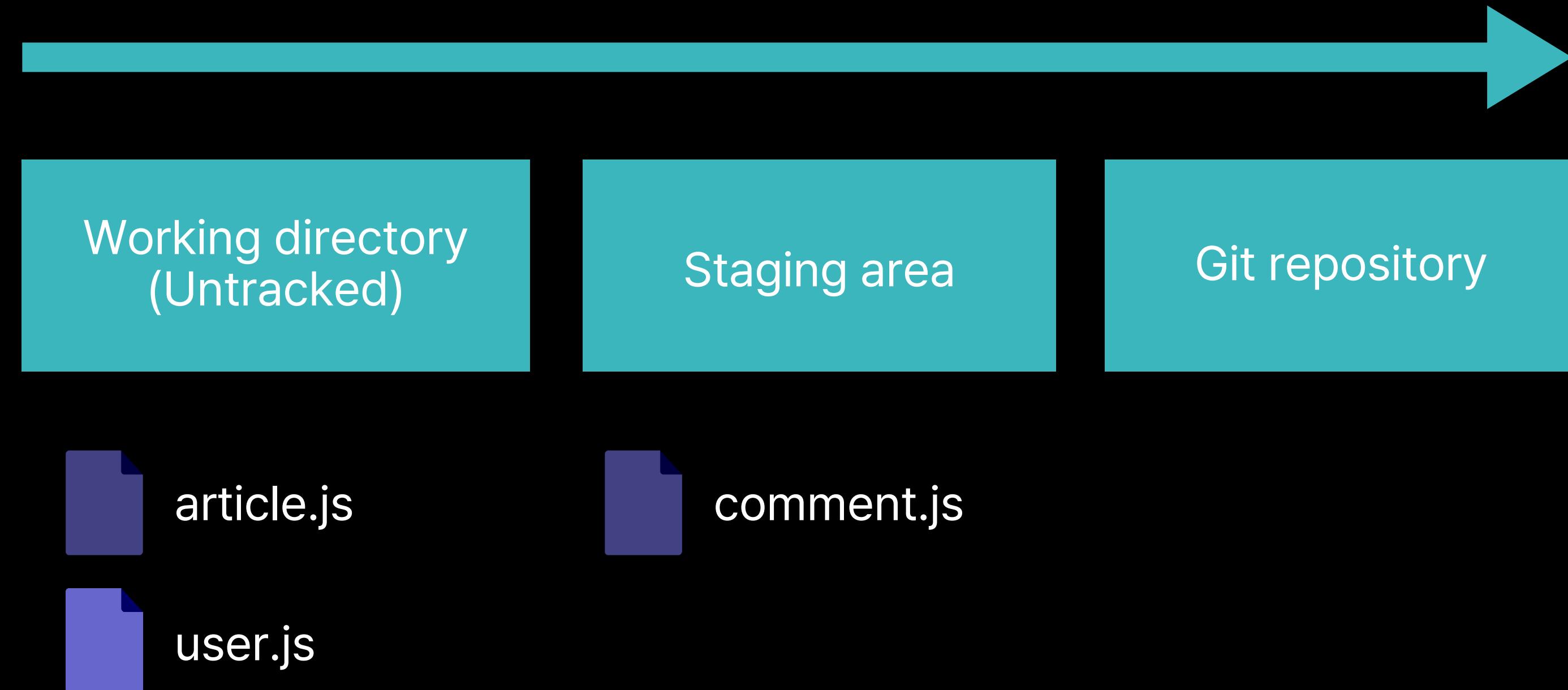


새로운 파일 생성 (2)

자율주행 트랙

데이터베이스

Git



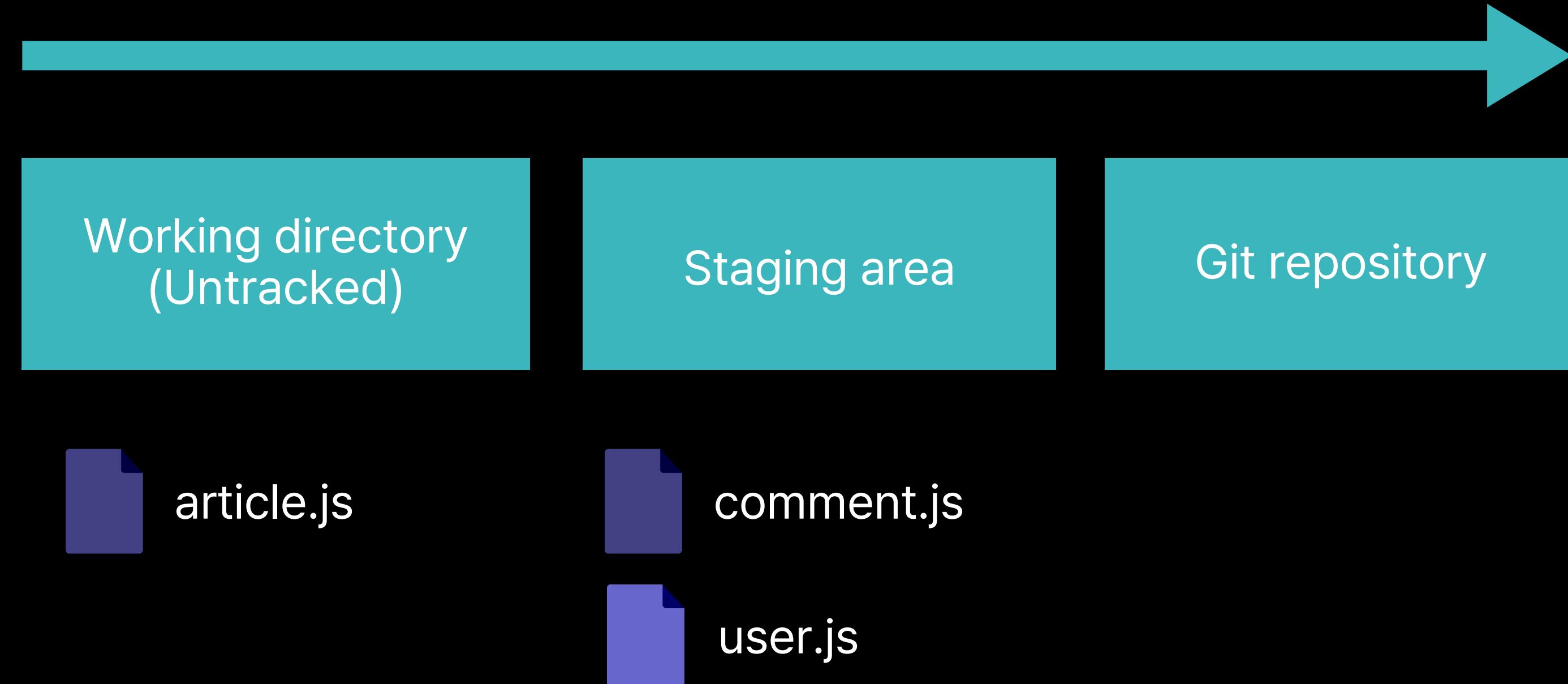


새로운 파일 생성 (2)

자율주행 트랙

데이터베이스

Git





새로운 파일 생성 (3)

자율주행 트랙

데이터베이스

Git

```
$ git add .
```

한 번에 추가할 파일이 너무 많다면
현재 폴더 내 전체 파일을 대상으로 지정할 수도 있습니다.



```
$ git reset <file_name>
```

위 명령을 이용하여 add 명령을 취소할 수 있습니다.



Staging 상태 확인

자율주행 트랙

데이터베이스

Git

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

    modified:   comment.js
    ...
```

‘git status’ 명령어로 staging area의 어떤 파일이 변경되었는지 등
파일의 상태를 확인할 수 있습니다.



‘comment.js’ 파일 작업을 staging 하였으므로,
이제 무엇을 수정하고 추가했는지 메시지를 남겨
저장소에 저장하는 작업을 진행합니다.



Git 저장소 반영 – commit

자율주행 트랙

데이터베이스

Git

```
$ git commit -m "modified comment.js"
master (root-commit) d78ade4] modified comment.js
 2 files changed, 4 insertions(+)
  create mode 100644 comment.js
  create mode 100644 user.js
```

준비 영역에 있는 파일들을 저장소에 반영하겠습니다.

생략 가능하지만 반영한 내용을 추후에 쉽게 알 수 있도록 **적절한 메세지를 넣어주세요.**

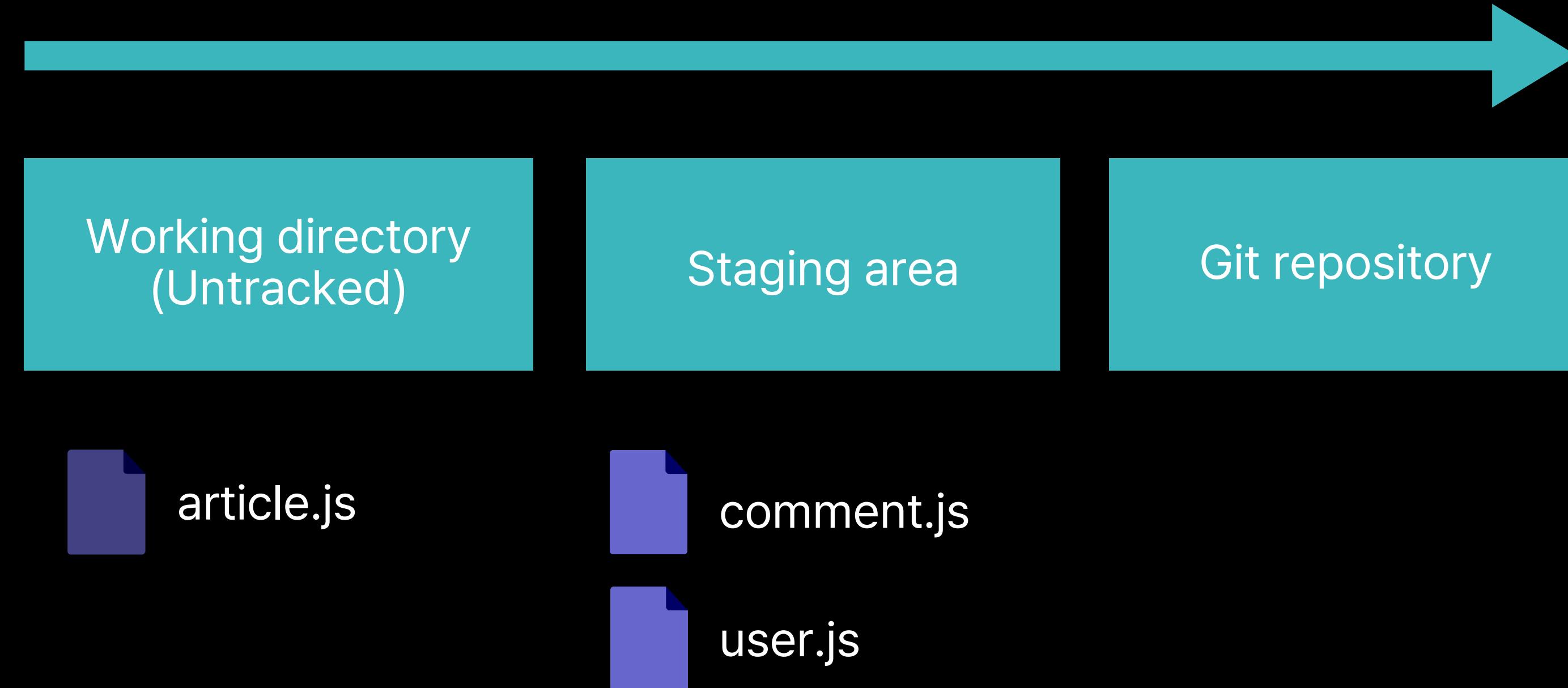


Git 저장소 반영 – commit

자율주행 트랙

데이터베이스

Git





Git 저장소 반영 – commit

자율주행 트랙

데이터베이스

Git





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

앞에서 적은 메세지에 오탏가 있거나
누락된 파일이 있을 경우 위 명령어로 정정할 수 있습니다.



Git 저장소 commit 내용 변경

자율주행 트랙

데이터베이스

Git

Initial commit

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
```

...

텍스트 편집기가 실행되고,
수정하고 싶은 부분을 수정 후 저장하면 그대로 반영됩니다.



Git 저장소 commit 기록 확인

자율주행 트랙

데이터베이스

Git

```
$ git log  
commit d78ade4c54fbfe333a466c78f4c2ca66d63d6053  
Author: Elicer <elice@elice.io>  
Date:   Tue Dec 11 22:23:13 2018 +0900  
  
modified comment.js
```

저장소 반영 내역이 궁금하다면
위 명령어를 사용해 보세요.

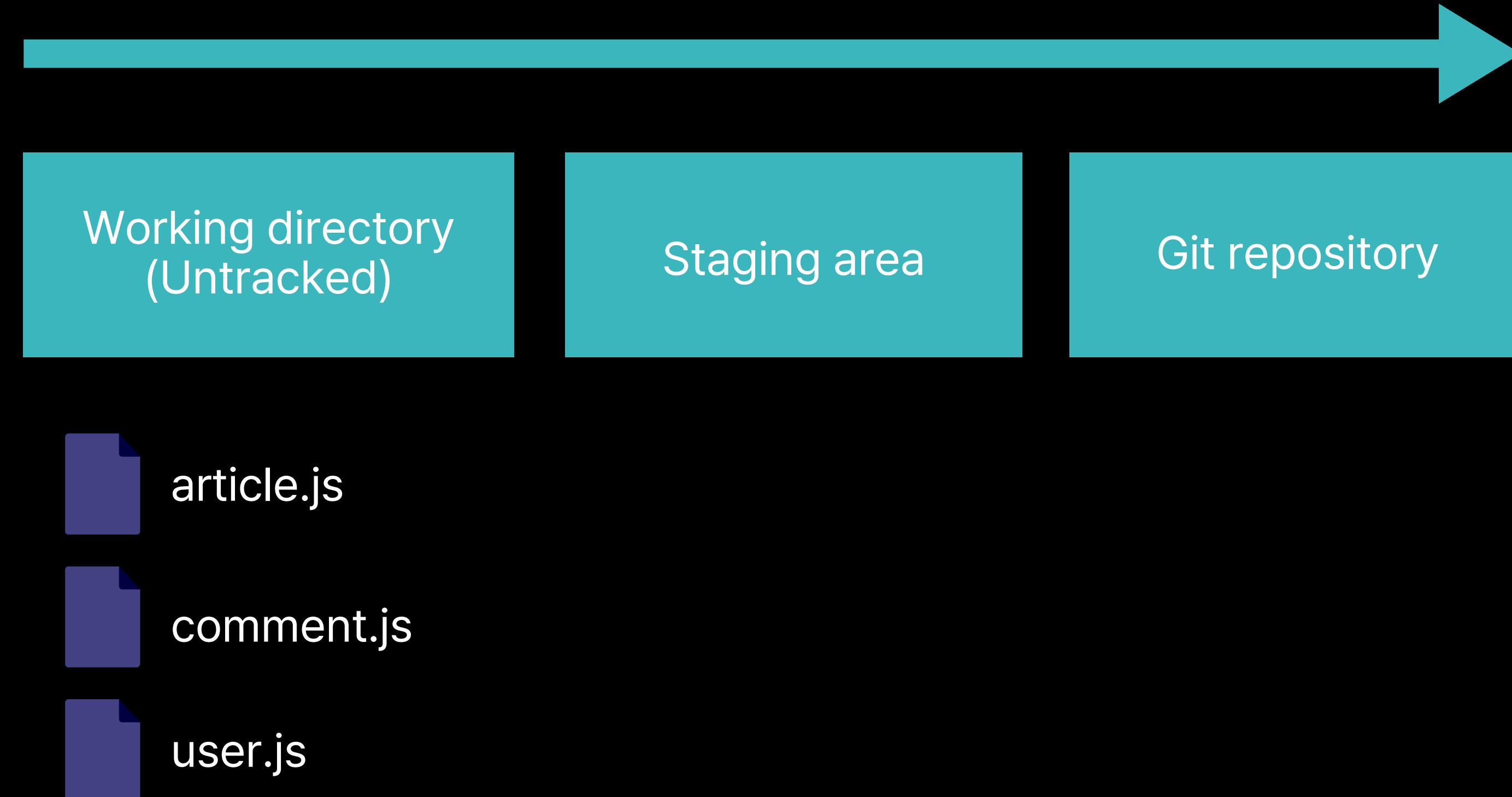


Git status

자율주행 트랙

데이터베이스

Git





```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    article.js
    comment.js
    user.js
nothing added to commit but untracked files present
(use "git add " to track )
```

Untracked files

add 명령어로 staging 되지 않은 & 한번도 commit 되지 않은 파일들

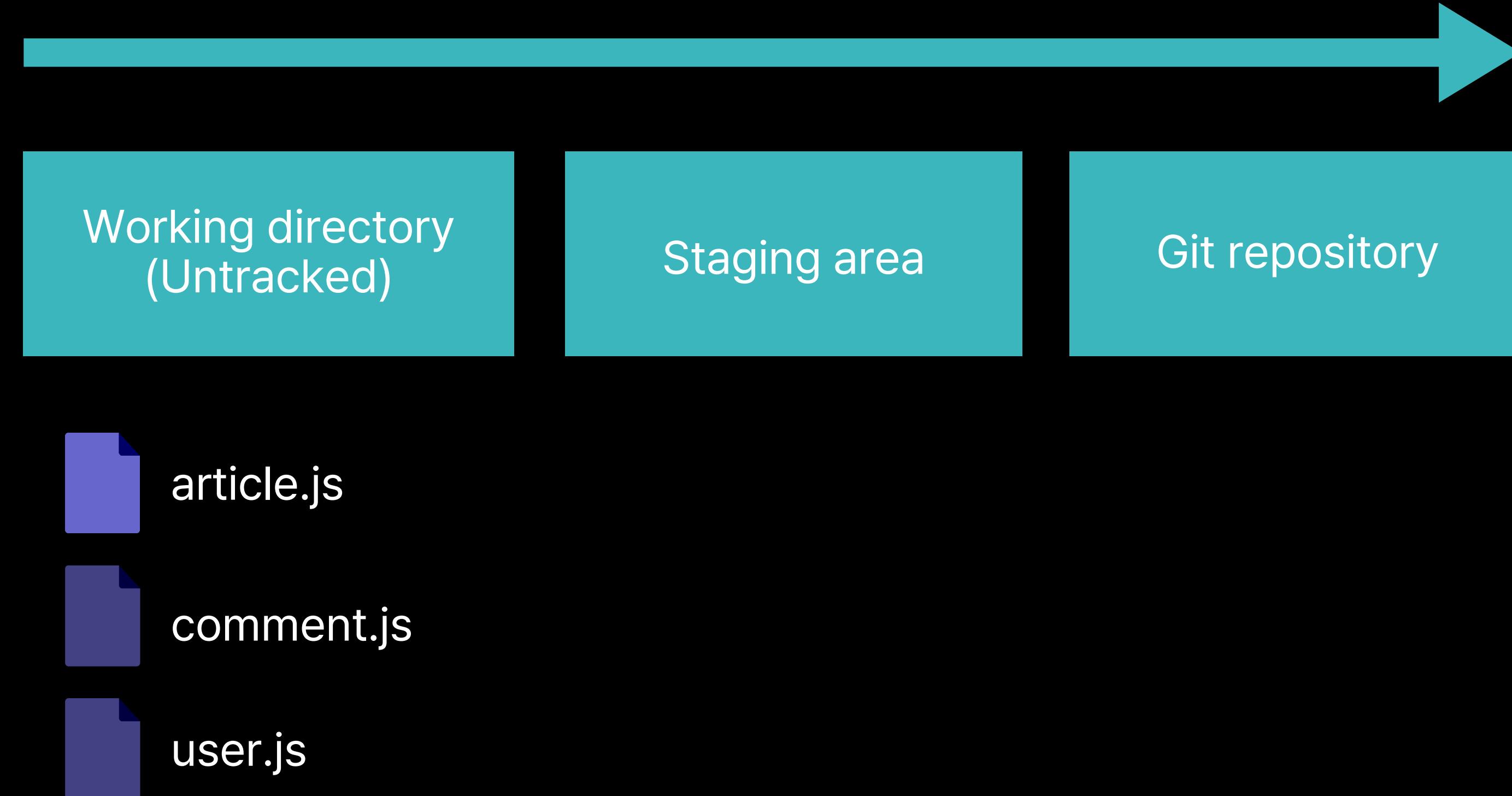


Git status

자율주행 트랙

데이터베이스

Git



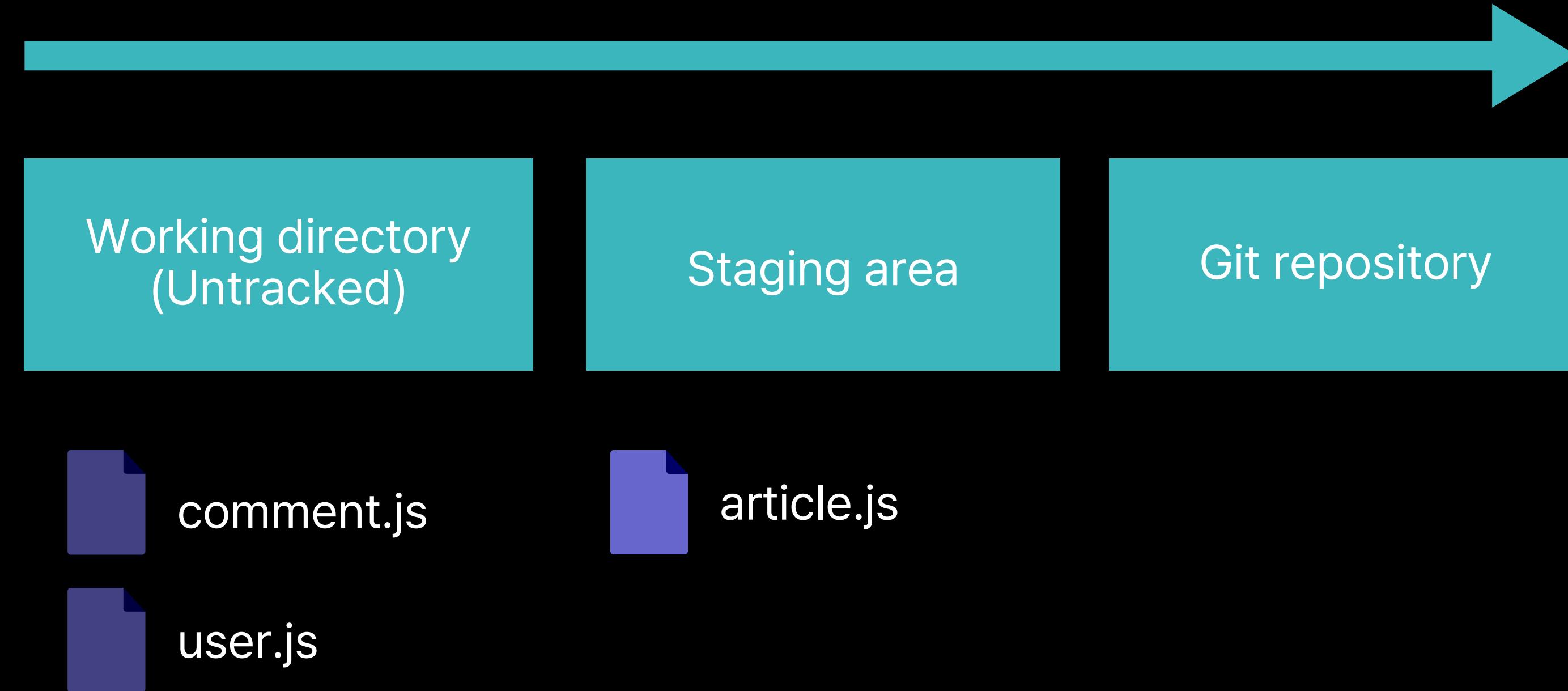


Git status

자율주행 트랙

데이터베이스

Git





Git status

자율주행 트랙

데이터베이스

Git

```
$ git status
On branch master

No commits yet

changes to be commited:
(use "git rm --cached <file>..." to unstage)
  new file : article.js

untracked files:
(use "git add <file>" to include in what will be committed)
  comment.js
  user.js
```

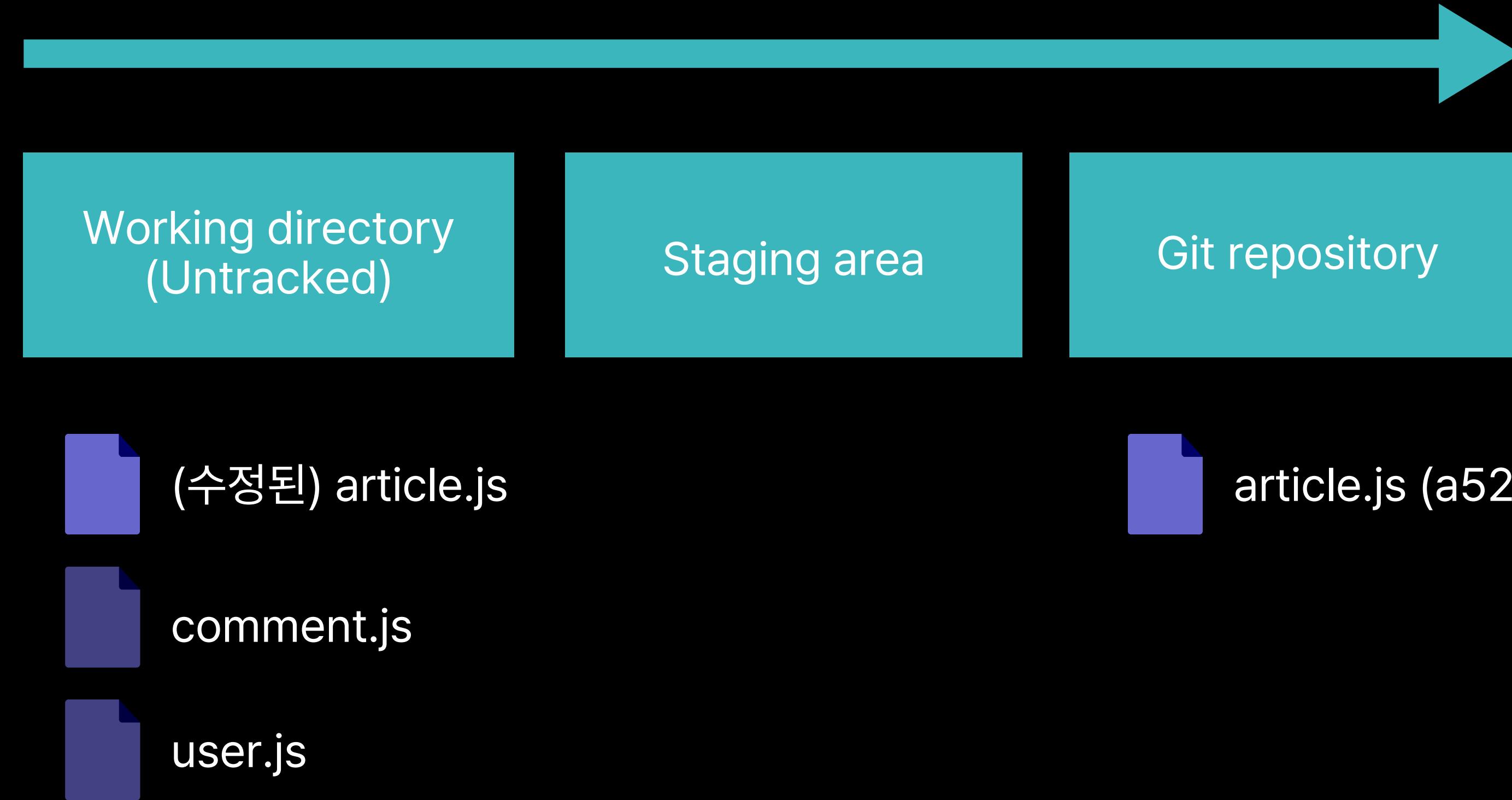


Git status

자율주행 트랙

데이터베이스

Git





Git status

자율주행 트랙

데이터베이스

Git

```
$ 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 ...)

    modified : article.js

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

modified

commit 된 파일 중 수정된 파일

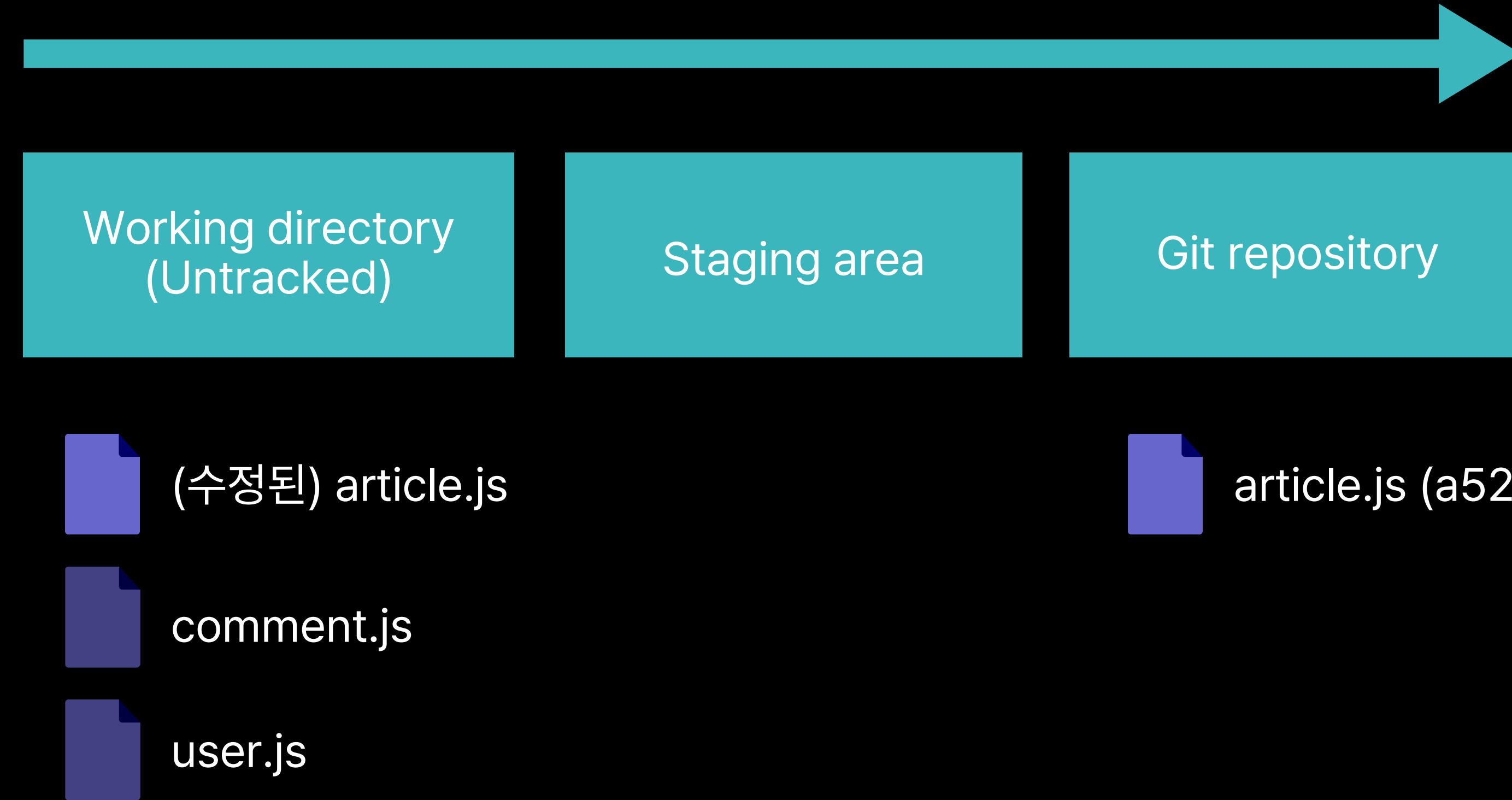


Git status

자율주행 트랙

데이터베이스

Git



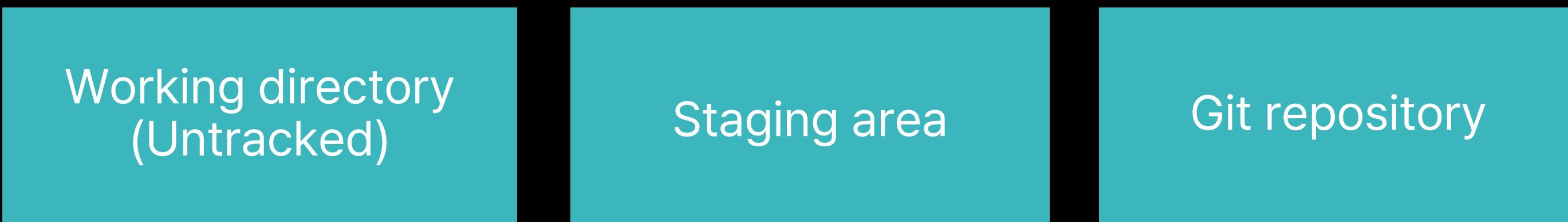


Git status

자율주행 트랙

데이터베이스

Git



comment.js



user.js



(수정된) article.js



article.js (a522b)



Git status

자율주행 트랙

데이터베이스

Git





Git status

자율주행 트랙

데이터베이스

Git

a522b

Commit size

Tree

Parent

Author

Committer

Init git

f1a14

Commit size

Tree

Parent : a522b

Author

Committer

modified article





Git Branch?

- 독립적으로 어떤 작업을 진행하기 위한 개념
- 각각의 Branch는 다른 Branch의 영향을 받지 않음

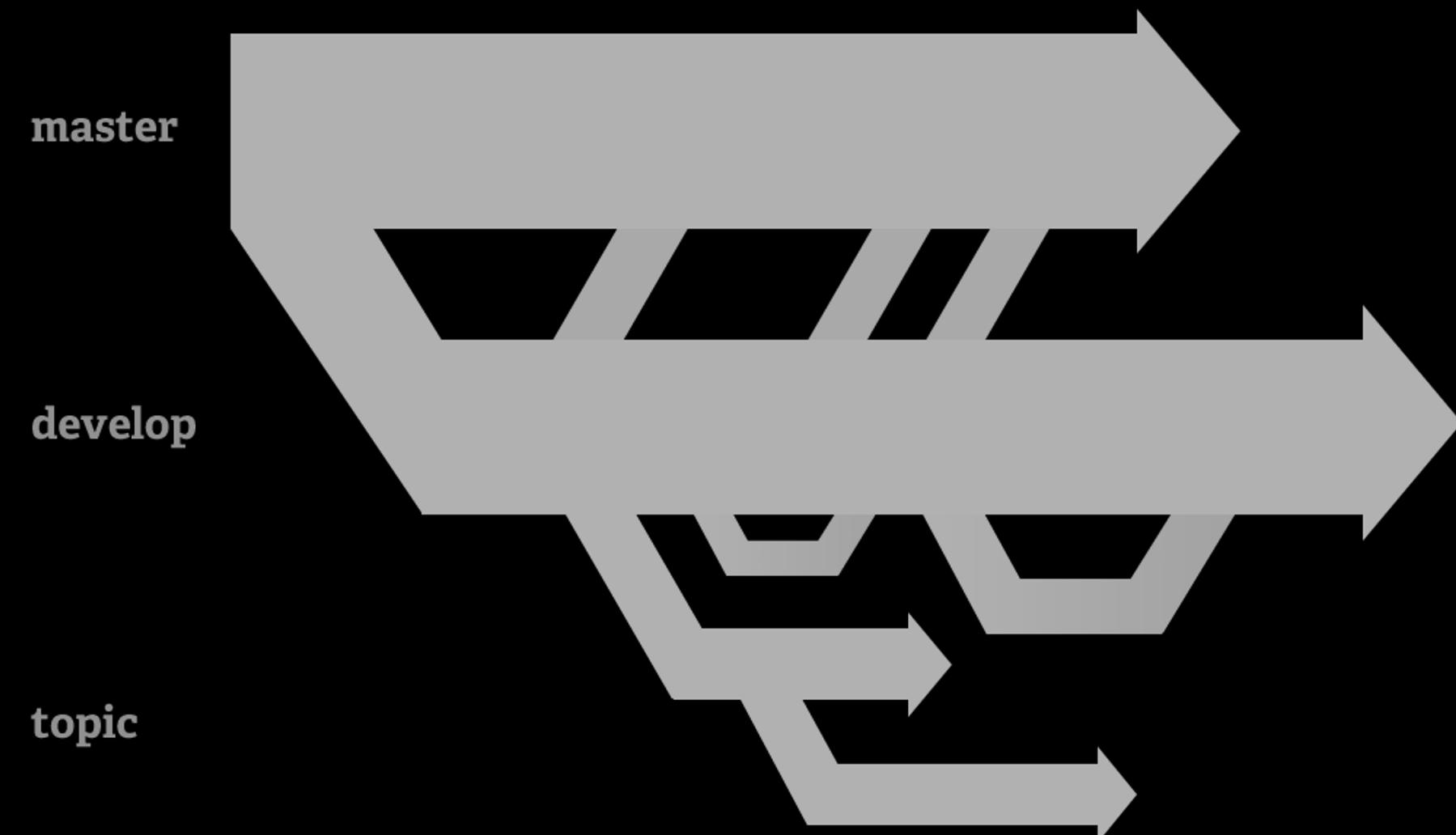


Git Branch

자율주행 트랙

데이터베이스

Git





Git Branch 종류

자율주행 트랙

데이터베이스

Git

메인 Branch

배포할 수 있는 수준의
안정적인 Branch

토픽 Branch

기능 추가나 버그 수정과 같은
단위 작업을 위한 Branch



Git Branch 생성

자율주행 트랙

데이터베이스

Git

```
$ git branch <branch_name>
```

Branch는 위 명령어로 생성할 수 있습니다.

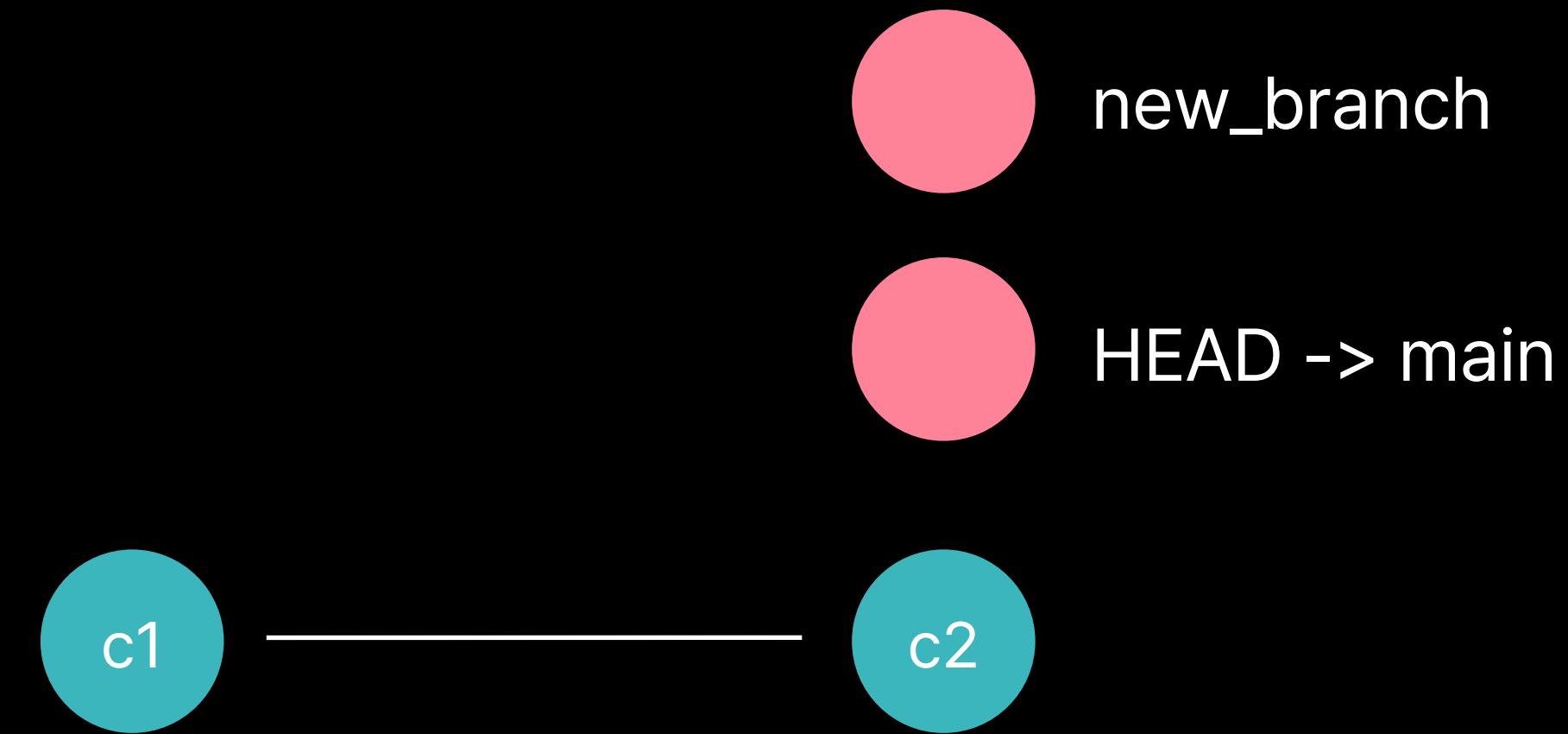


Git Branch 생성

자율주행 트랙

데이터베이스

Git





Git Branch 전환

자율주행 트랙

데이터베이스

Git

```
$ git branch  
  new_branch  
* main
```

현재의 Branch는 위 명령어를 통해 확인할 수 있습니다.



Git Branch 전환

자율주행 트랙

데이터베이스

Git

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

Branch 전환은 아래의 명령어를 통해 할 수 있습니다.

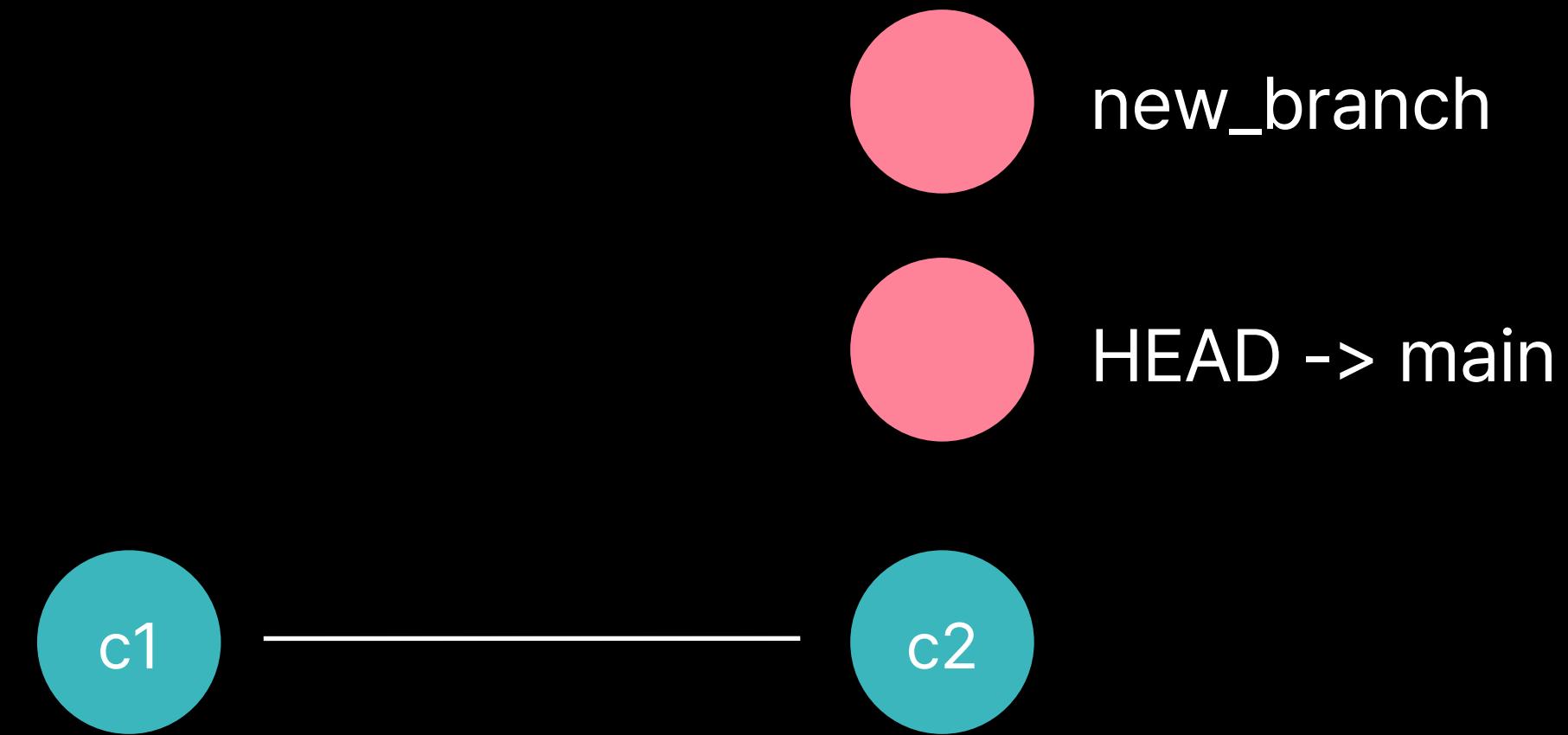


Git Branch 전환

자율주행 트랙

데이터베이스

Git



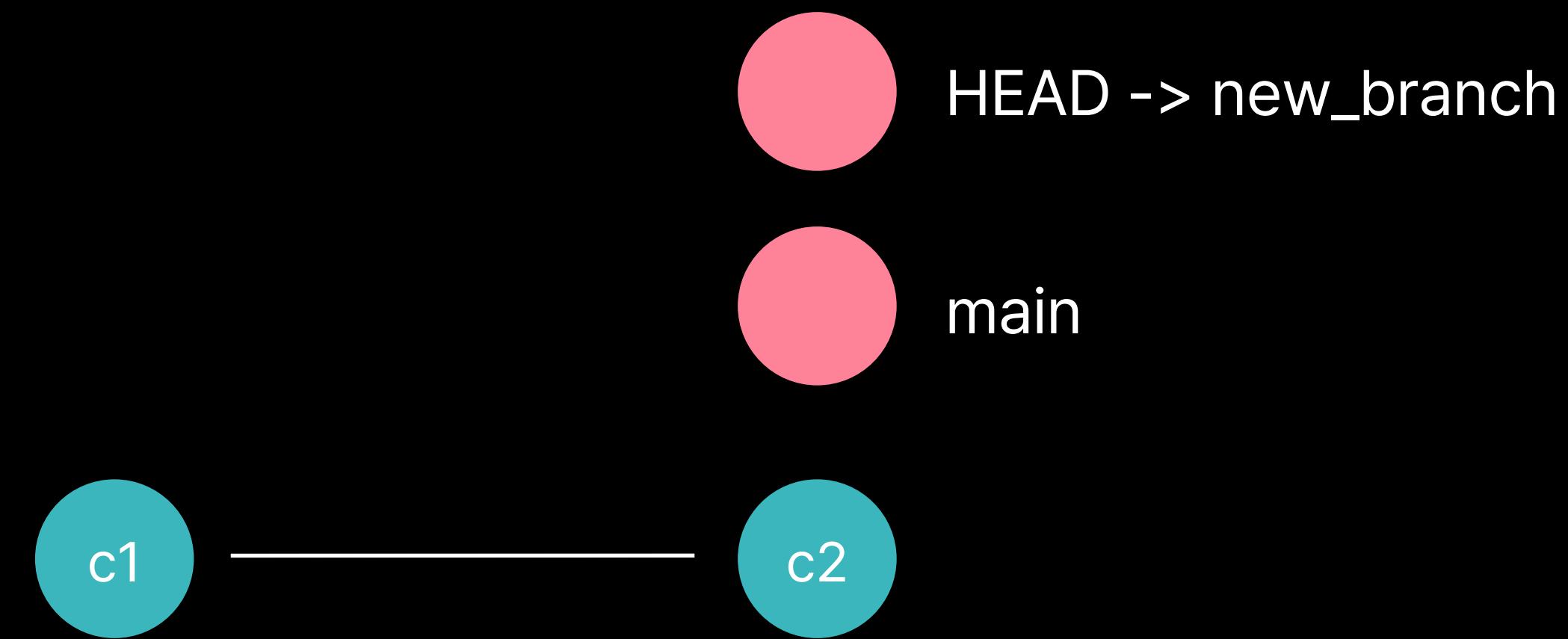


Git Branch 전환

자율주행 트랙

데이터베이스

Git





‘new_branch` Branch의 working directory에서
새로운 정보를 넣어 commit 해보겠습니다.
먼저 ‘new_branch`로 위치를 이동시켜볼까요?

```
$ git checkout new_branch
```

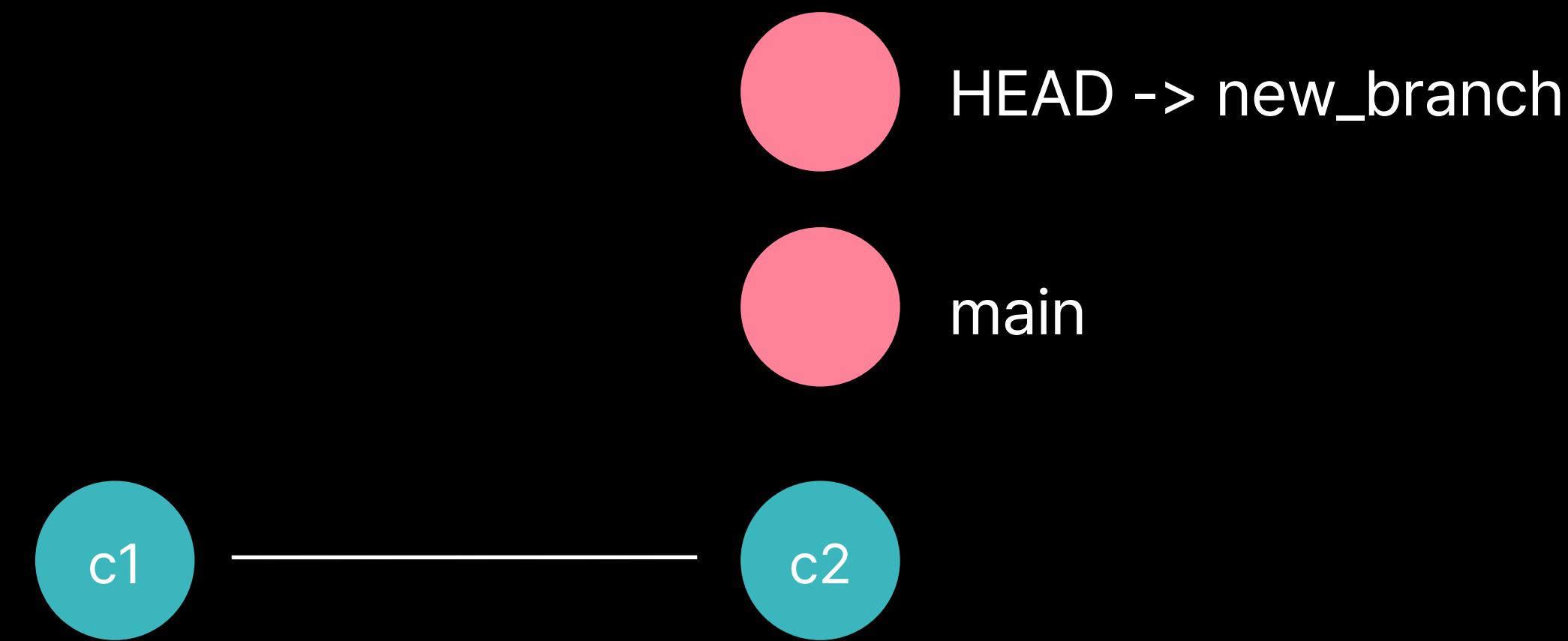


Fast Forward

자율주행 트랙

데이터베이스

Git



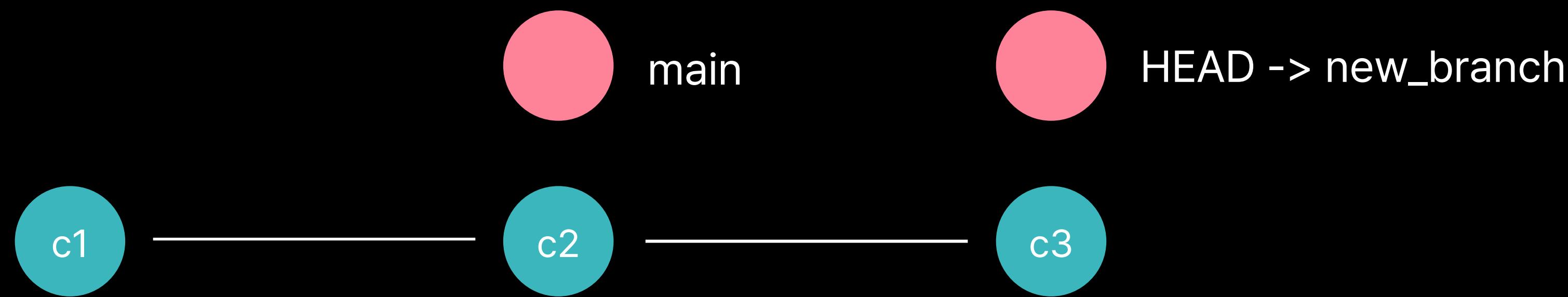


Fast Forward

자율주행 트랙

데이터베이스

Git





‘new_branch` Branch에서의 작업을 끝마치고,

‘main` Branch로 통합합니다.

= Merge



```
$ git checkout main
Switched to branch 'main'
$ git merge new_branch
Updating d78ade4..a63hec2
Fast-forward
  comment.js | 3 +--
  1 file changed, 2 insertions(+), 1 deletion(-)
```

먼저 `main` Branch로 이동하여
`new_branch` Branch를 병합합니다.

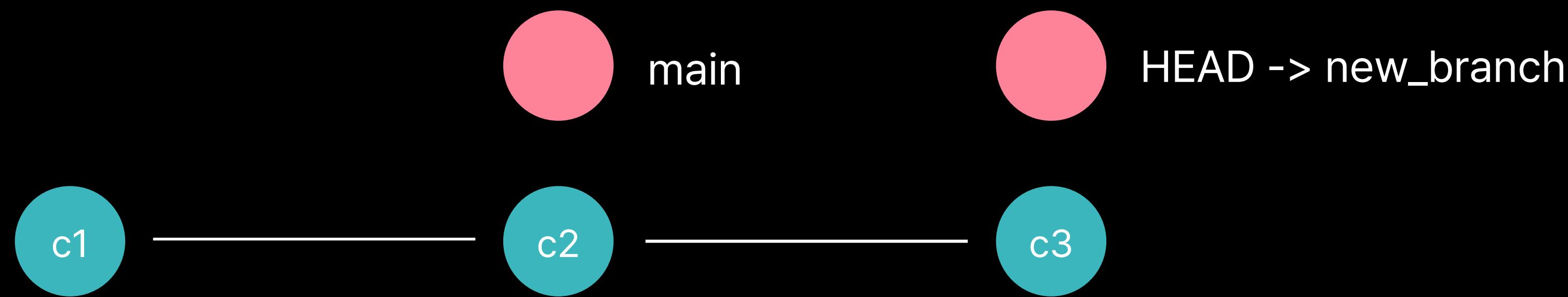


Fast Forward

자율주행 트랙

데이터베이스

Git



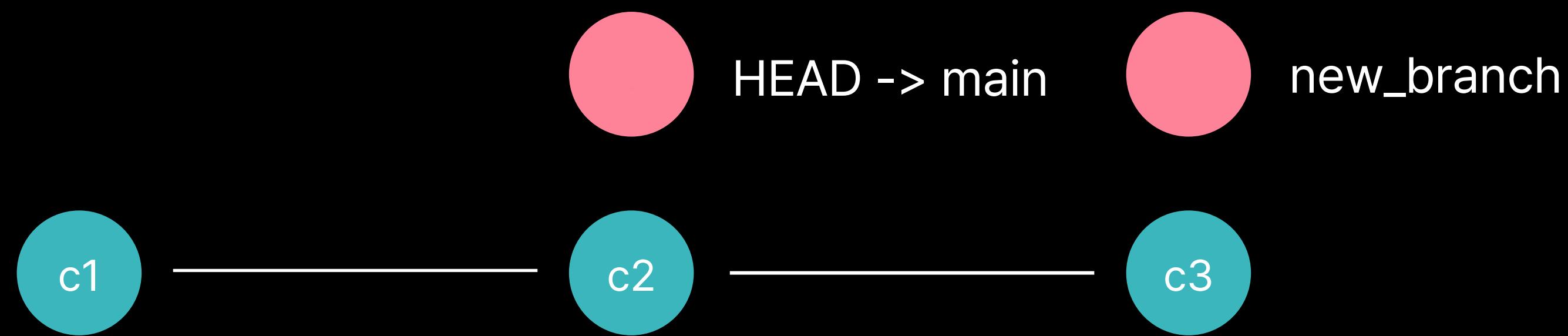


Fast Forward

자율주행 트랙

데이터베이스

Git



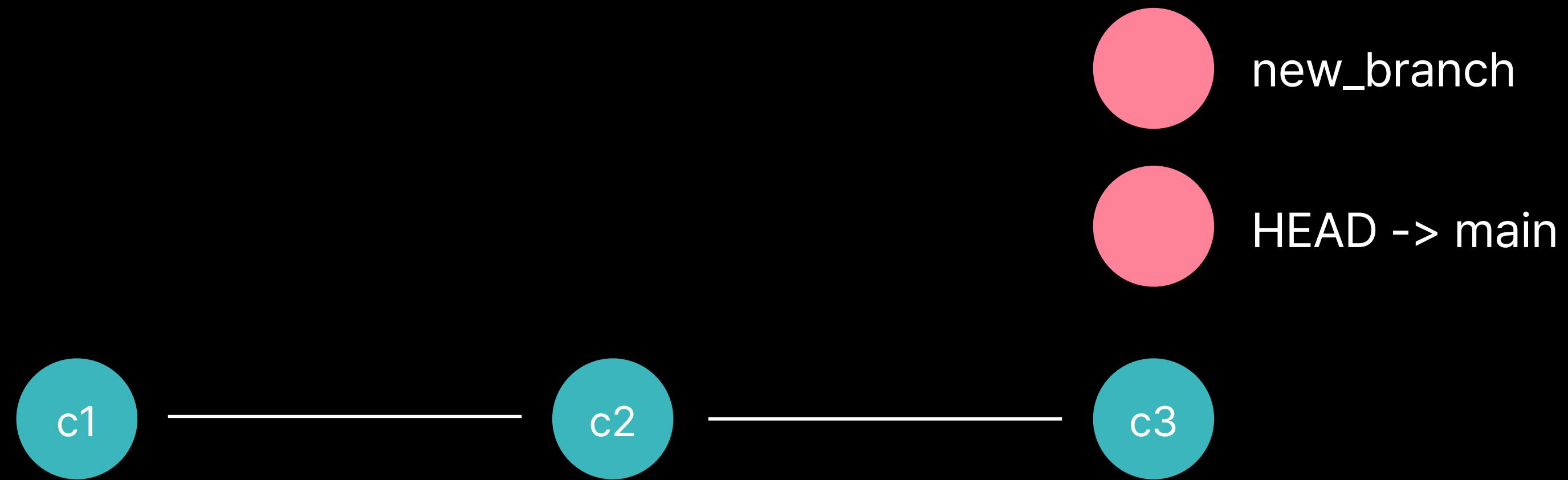


Fast Forward

자율주행 트랙

데이터베이스

Git





`new_branch` Branch의 내용이 `main` Branch에서
업데이트 된 내용이기 때문에 **곧바로** merge가 되는 것을 확인할 수 있습니다.

이렇게 merge가 곧바로 이루어지는 것을 **fast-forward**라고 부릅니다.



이번에는 각각의 Branch의 working directory에서
같은 파일의 내용을 다르게 수정해보겠습니다.

명심하세요! 각각의 Branch는 다른 Branch의
영향을 받지 않기 때문에, 여러 작업을 동시에 진행할 수 있습니다.

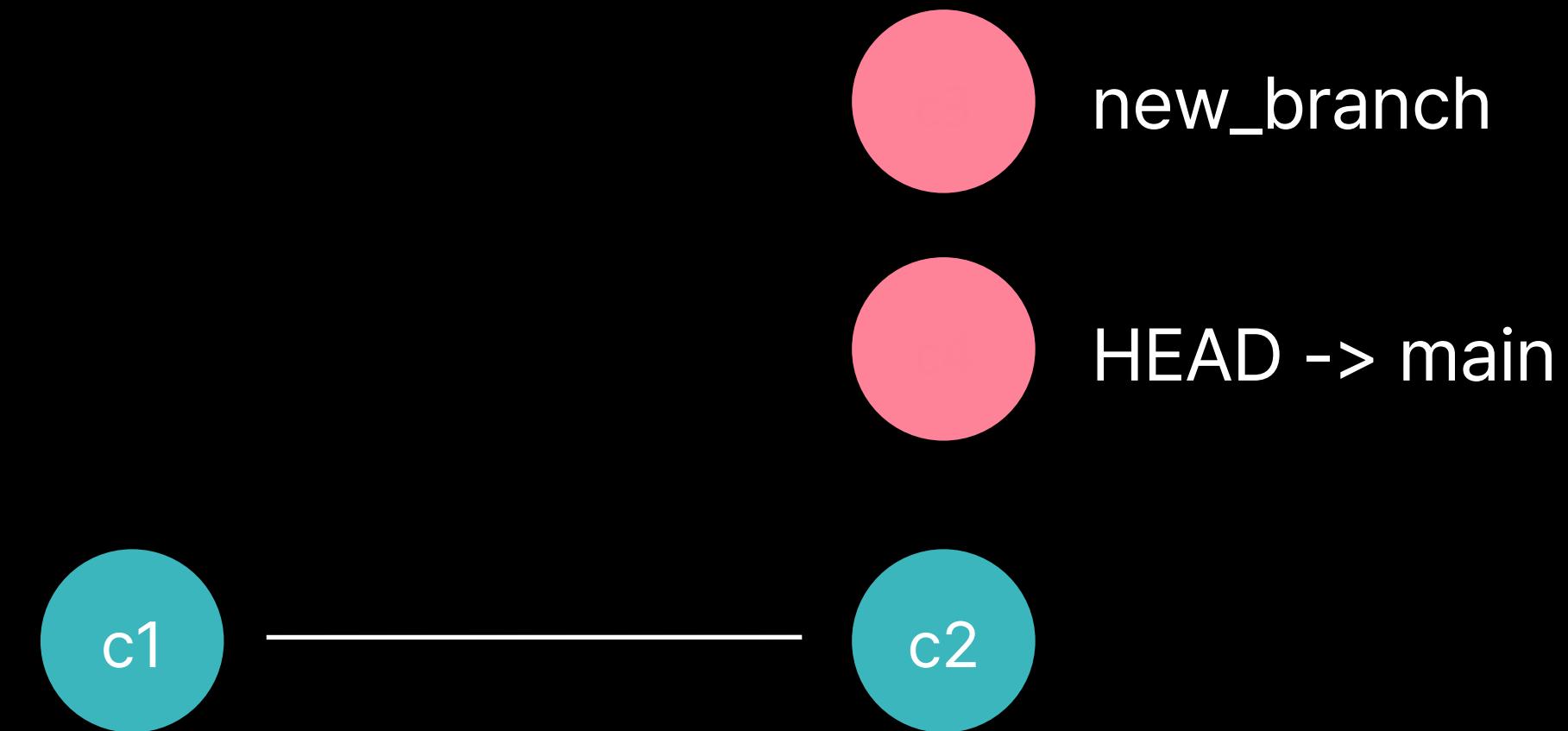


갈라지는 Branch

자율주행 트랙

데이터베이스

Git



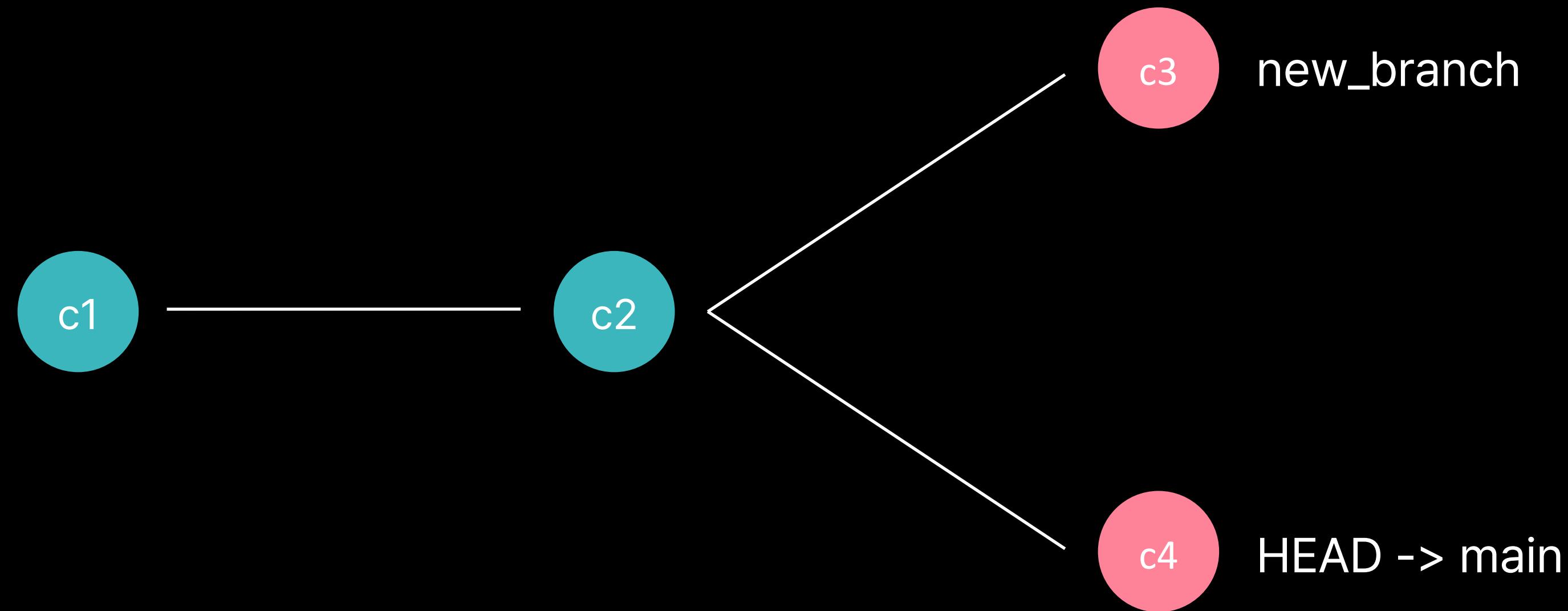


갈라지는 Branch

자율주행 트랙

데이터베이스

Git





```
$ git log --pretty=oneline --graph --all
* 0dd4b5e377ea70f50bce2bfe0c26630f5113318f (HEAD -> main) add main
| * a6a6ed1a0c25f8974c38023777a16a290c7a4fcf (new_branch) add new_branch
|/
* aeb77657d31d35ff87496c6025a1ebcc63f94c8b Initial Commit
```

`git log --pretty=oneline --graph --all`을 사용하면
commit graph를 예쁘게 확인할 수 있습니다.



`git checkout main`을 이용하여 main으로 checkout한 후
`git merge new_branch`로 merge 해보겠습니다.



```
$ git checkout main
switched to branch 'main'
$ git merge new_branch
Merge made by the 'ort' strategy.
 article.js | 1 +
 1 file changed, 1 insertion(+)
```

‘git checkout main’을 이용하여 main으로 checkout한 후
‘git merge new_branch’로 merge 해보겠습니다.

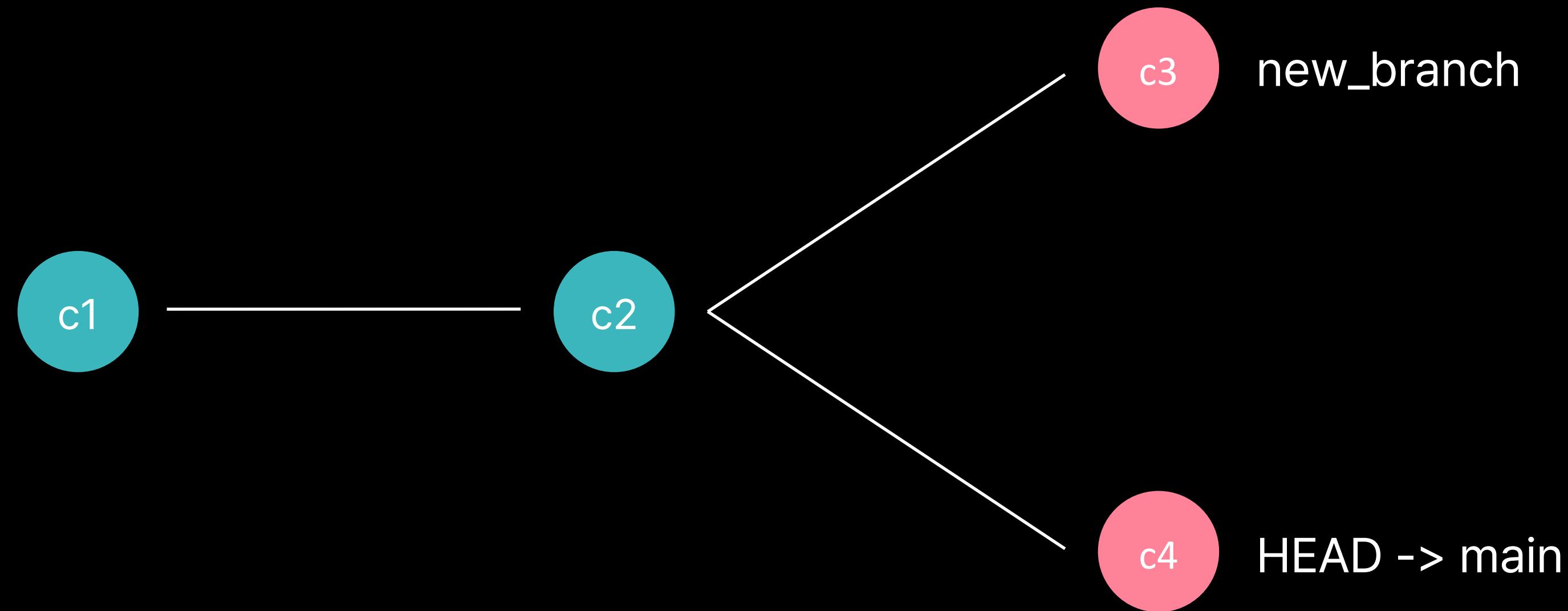


갈라지는 Branch

자율주행 트랙

데이터베이스

Git



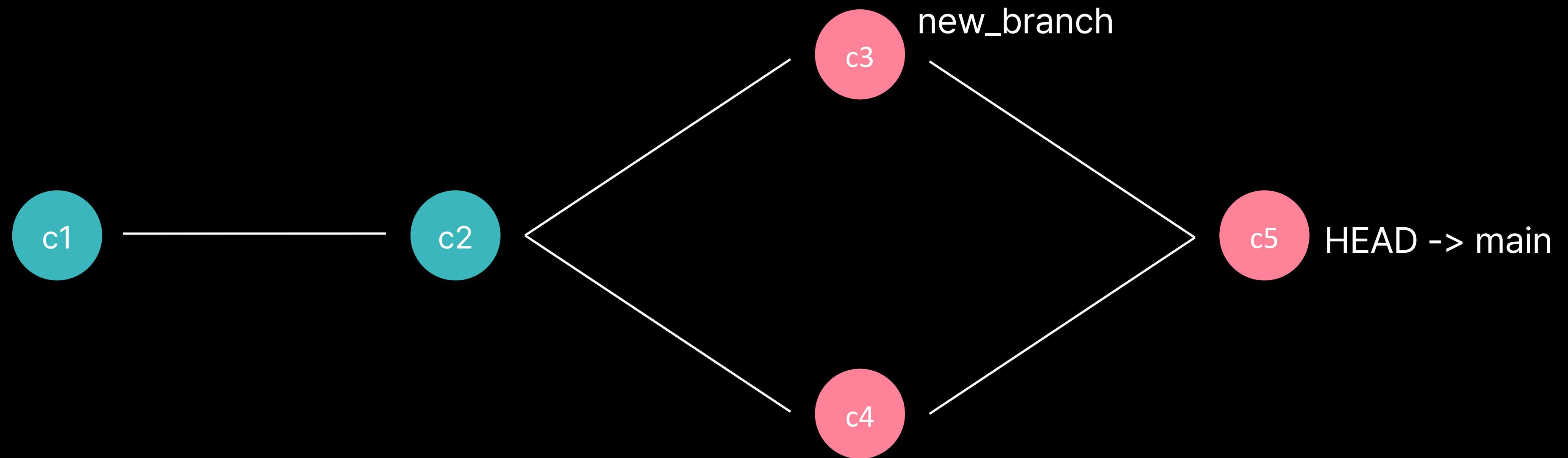


갈라지는 Branch

자율주행 트랙

데이터베이스

Git





Git Branch 삭제

자율주행 트랙

데이터베이스

Git

```
$ git branch --merged  
* main  
  new_branch
```

위 명령어로 Merge 된 Branch를 볼 수 있습니다.



Git Branch 삭제

자율주행 트랙

데이터베이스

Git

```
$ git branch -d new_branch  
Deleted branch new_branch (was 1454fda).
```

사용을 마친 branch는 `git branch -d <branch_name>`을
이용하여 삭제할 수 있습니다.



Merge Conflict

자율주행 트랙

데이터베이스

Git

Merge한 두 branch에서
같은 파일을 변경했을 때 충돌이 발생합니다.



Merge Conflict

자율주행 트랙

데이터베이스

Git

```
$ git merge new_branch  
Auto-merging article.js  
CONFLICT (content): Merge conflict in article.js  
Automatic merge failed; fix conflicts and then commit the result.
```

‘git merge new_branch’ 명령을 수행했을 때
위와 같이 충돌이 발생했습니다.

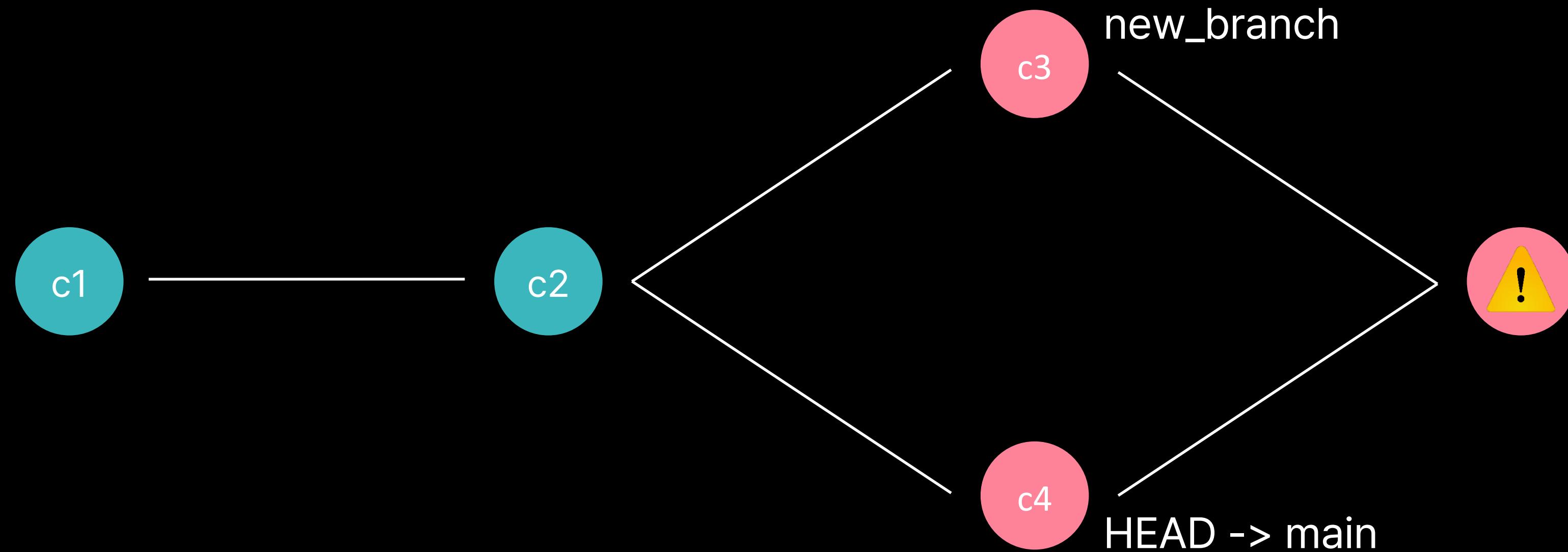


갈라지는 Branch

자율주행 트랙

데이터베이스

Git





Merge Conflict

자율주행 트랙

데이터베이스

Git

```
$ git status
On branch main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:  article.js

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

‘git status’ 명령어로
어느 파일에서 충돌이 발생했는지 확인합니다.



Merge 해결하기

자율주행 트랙

데이터베이스

Git

```
const i = "article";
<<<<< HEAD
const main = "main";
=====
const article = "hello";
>>>>> new_branch
```

충돌이 일어난 `article.js` 파일을 열어봅니다.



Merge 해결하기

자율주행 트랙

데이터베이스

Git

```
const i = "article";
const article = "hello";
```

수정 완료 후,

'<<<<<<', '=====', '>>>>>'가 포함된 행을 삭제해 줍니다.



Merge 해결하기

자율주행 트랙

데이터베이스

Git

```
$ git add .
$ git commit -m "merged new_branch"
```

이후 `git add`, `git commit` 과정을 거쳐
다시 merge 해줍니다.



Merge Conflict를 방지하려면?

자율주행 트랙

데이터베이스

Git

‘main’ branch의 변화를 지속적으로 가져와서,
충돌이 발생하는 부분을 최대한 제거합니다.



원격 저장소?

자율주행 트랙

데이터베이스

Git

인터넷이나 네트워크 어딘가에 있는 저장소



원격 저장소?

자율주행 트랙

데이터베이스

Git

TheAlgorithms / Python

Type ⌘ to search

Code Issues 33 Pull requests 214 Discussions Actions Projects Wiki Security Insights

Python Public

Sponsor Watch 5.9k Fork 43.7k Star 180k

master 9 Branches 0 Tags Go to file Add file Code

MaximSmolskiy and pre-commit-ci[bot] Fix some SIM114 per file igno... c599f6c · 3 days ago 3,347 Commits

.devcontainer Upgrade our Devcontainer to Python 3.12 on Debian book... 8 months ago

.github adding a proper fractions algorithm (#11224) 2 months ago

.vscode Create a Simultaneous Equation Solver Algorithm (#8773) last year

audio_filters Fix some ARG002 per file ignores (#11382) 2 weeks ago

backtracking Enable ruff RUF002 rule (#11377) 3 weeks ago

bit_manipulation Enable ruff RUF002 rule (#11377) 3 weeks ago

blockchain Ruff pandas vet (#10281) 7 months ago

boolean_algebra [pre-commit.ci] pre-commit autoupdate (#11322) 2 months ago

cellular_automata Enable ruff PLR5501 rule (#11332) 2 months ago

ciphers Enable ruff ICN001 rule (#11329) 2 months ago

compression Enable ruff RUF002 rule (#11377) 3 weeks ago

computer_vision [pre-commit.ci] pre-commit autoupdate (#11380) 3 weeks ago

conversions [pre-commit.ci] pre-commit autoupdate (#11322) 2 months ago

data_structures Fix some SIM114 per file ignores (#11395) 3 days ago

digital_image_processing Enable ruff RUF002 rule (#11377) 3 weeks ago

divide_and_conquer Enable ruff PLR5501 rule (#11332) 2 months ago

About All Algorithms implemented in Python the-algorithms.com/

python education algorithm practice interview sorting-algorithms learn algs algorithm-competitions sorts hacktoberfest algorithms-implemented community-driven searches

Readme MIT license Code of conduct Activity Custom properties 180k stars 5.9k watching 43.7k forks Report repository

Releases No releases published

Sponsor this project Liberapay.com/TheAlgorithms



Git 원격 저장소 받아오기

자율주행 트랙

데이터베이스

Git

```
$ git clone <repository_url>
```

원격 저장소에 있는 git repository를 복사하는 명령어

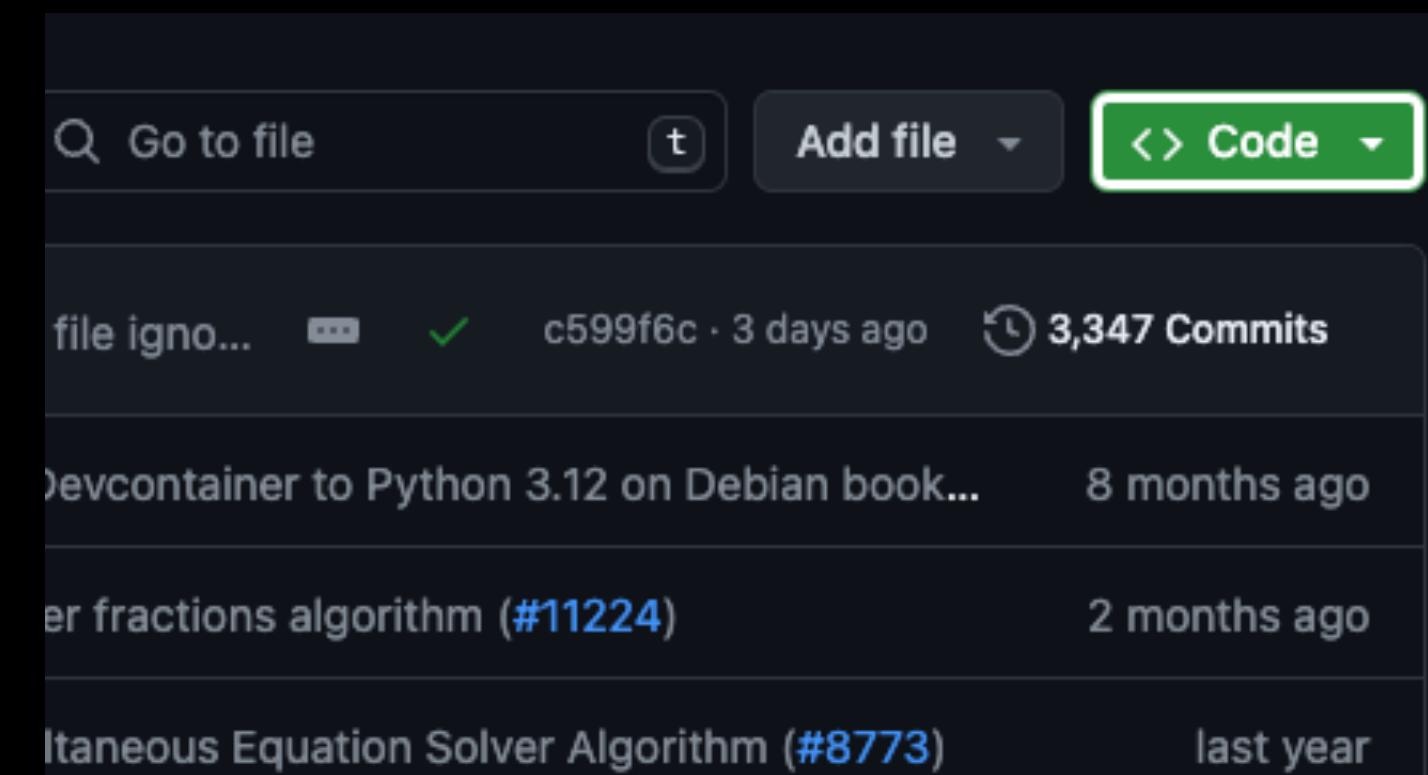


Git 원격 저장소 받아오기

자율주행 트랙

데이터베이스

Git



원격 저장소(GitHub, Gitlab 등)에서
원하는 Repository의 'Code' (혹은 'Clone') 버튼을 누릅니다.

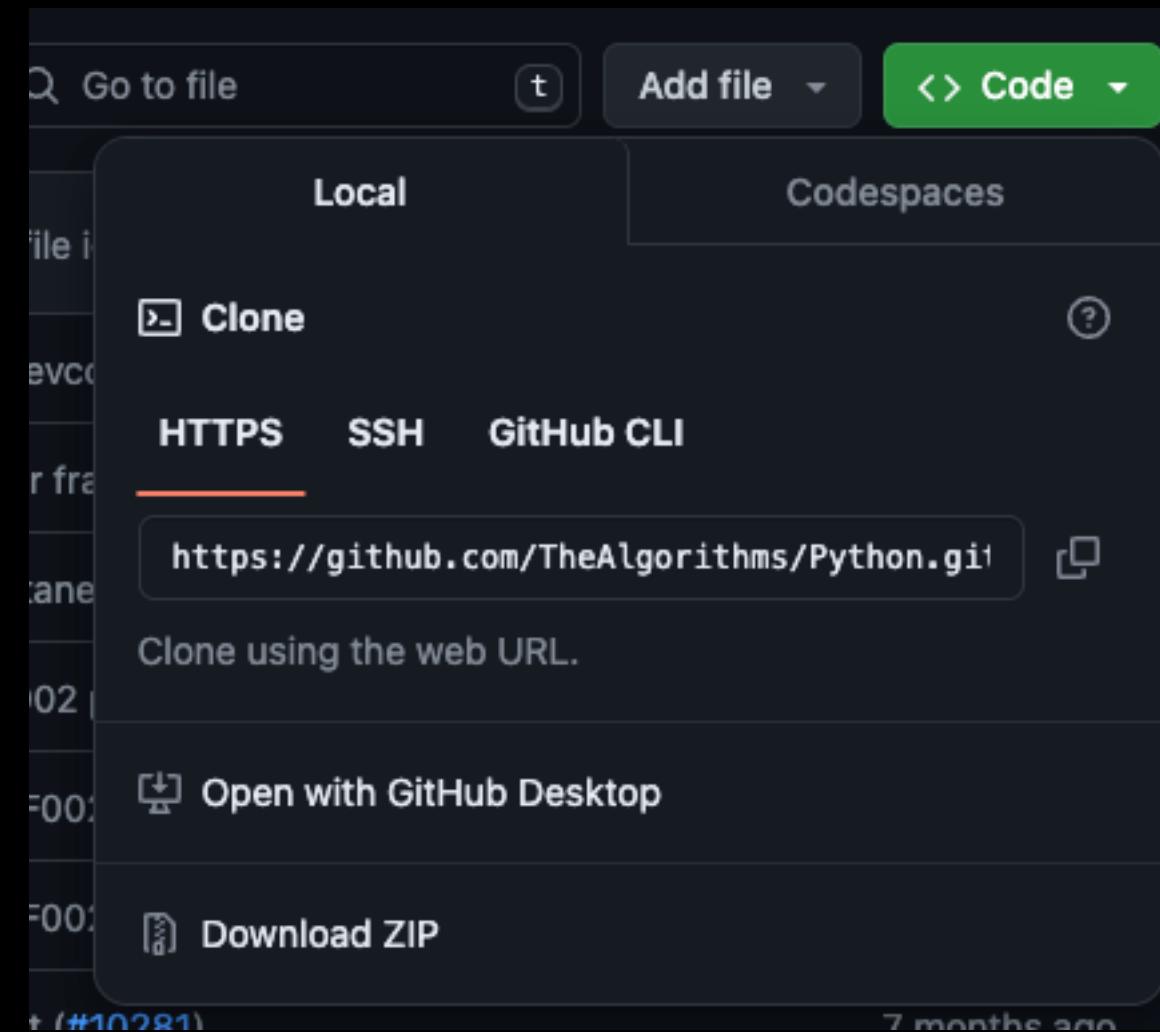


Git 원격 저장소 받아오기

자율주행 트랙

데이터베이스

Git



여러 옵션 중 **HTTPS 옵션**의 주소를 복사합니다.



Git 원격 저장소 받아오기

자율주행 트랙

데이터베이스

Git

```
$ git clone https://github.com/TheAlgorithms/Python.git
```

‘git clone’ 뒤에 복사한 원격 저장소의 URL을 넣어줍니다.



```
$ git remote add origin https://github.com/TheAlgorithms/Python.git
```

이미 생성된 로컬 저장소에 원격 저장소를 연결하려면
위 명령어를 사용합니다.



```
$ git remote add origin https://github.com/TheAlgorithms/Python.git
```

웹 호스트 서비스

그룹명

프로젝트명



원격 저장소 살펴보기

자율주행 트랙

데이터베이스

Git

```
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/.../git_test.git
  Push  URL: https://github.com/.../git_test.git
  HEAD branch: main
  Remote branch:
    main tracked
  Local branch configured for 'git pull':
    main merges with remote main
  Local ref configured for 'git push':
    main pushes to main (up to date)
```



Pull

원격 저장소에서 데이터 가져오기 + 병합(Merge)

Fetch

원격 저장소에서 데이터 가져오기



원격 저장소 갱신 - pull

자율주행 트랙

데이터베이스

Git

```
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 ...
Unpacking objects: 100% (3/3), done.
From https://github.com/.../git_test
  0daf5d2..c61952d  main -> origin/main
Updating 0daf5d2..c61952d
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

```
$ git log --all
commit f7b775d (HEAD->main, origin/main)
commit 725fc8b
commit fad4a9a
commit 1418700 init repository
```

원격 저장소에서 데이터를 가져와
로컬 데이터와 병합합니다.



원격 저장소 갱신 - fetch

자율주행 트랙

데이터베이스

Git

```
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 ...
Unpacking objects: 100% (3/3), done.
From https://github.com/.../git_test
  c61952d..932dd32  main -> origin/main
```

```
$ git log --all
commit f7b775d (HEAD->main)
commit 725fc8b (origin/main)
commit fad4a9a
commit 1418700 init repository
```

원격 저장소에서 데이터를 가져오지만, 병합하지는 않습니다.

진행중인 작업을 마무리하고 병합해 주어야 합니다.



원격 저장소 발행 - push

자율주행 트랙

데이터베이스

Git

```
$ git push origin main
```

로컬 저장소에서 작업한 내용을 원격 저장소에 반영합니다.

(협업 시, 다른 사람이 먼저 push한 상태에서는 push할 수 없습니다.
다른 사람이 작업한 것을 merge부터 해주세요.)



1. `git remote add origin` (또는 다른 원격 저장소 이름)으로 로컬 저장소와 **연결합니다.**
2. `git fetch` 또는 `git pull`을 이용하여 원격 저장소의 내용을 동기화합니다.
3. fetch를 실행한 경우 `git merge origin/main`로 병합을 완료해 줍니다.
4. `git push origin main`을 이용하여 변경된 사항을 원격 저장소에 **반영합니다.**



origin이란?

자율주행 트랙

데이터베이스

Git

내 컴퓨터에 저장되어 있는 저장소와
원격 저장소를 연결하기 위해서 아래와 같은 명령을 사용했습니다.

```
$ git remote add origin https://github.com/TheAlgorithms/Python.git
```

이는 원격 저장소의 단축 이름을 **origin**으로 지정한다는 의미입니다.



origin이 아닌 다른 이름으로 원격 저장소의 이름을
지정해 줄 수도 있습니다.

```
$ git remote add myproject https://github.com/TheAlgorithms/Python.git
```



origin이란?

자율주행 트랙

데이터베이스

Git

기본적으로 만들어진 원격 저장소의 이름은 origin이 default 값입니다.
때문에 clone으로 복사해온 저장소의 이름은 origin으로 통일되게 됩니다.



origin이란?

자율주행 트랙

데이터베이스

Git

```
$ git remote -v  
origin https://gitlab.com/group/project (fetch)  
origin https://gitlab.com/group/project (push)
```

`-v` 옵션을 사용하면
지정한 저장소의 이름과 주소를 함께 볼 수 있습니다.