Git을 사용한 버전 관리

Git 시작하기

/* elice */

수강 목표

Git저장소에 **작업 내용**을 **반영**할 수 있습니다.

Git저장소의 **세 가지 영역**을 알 수 있습니다.

Git저장소의 현재 **상태**를 **파악**할 수 있습니다.

목차

Git 파일 생성

새로운 파일 생성

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

파일 영역의 라이프 사이클

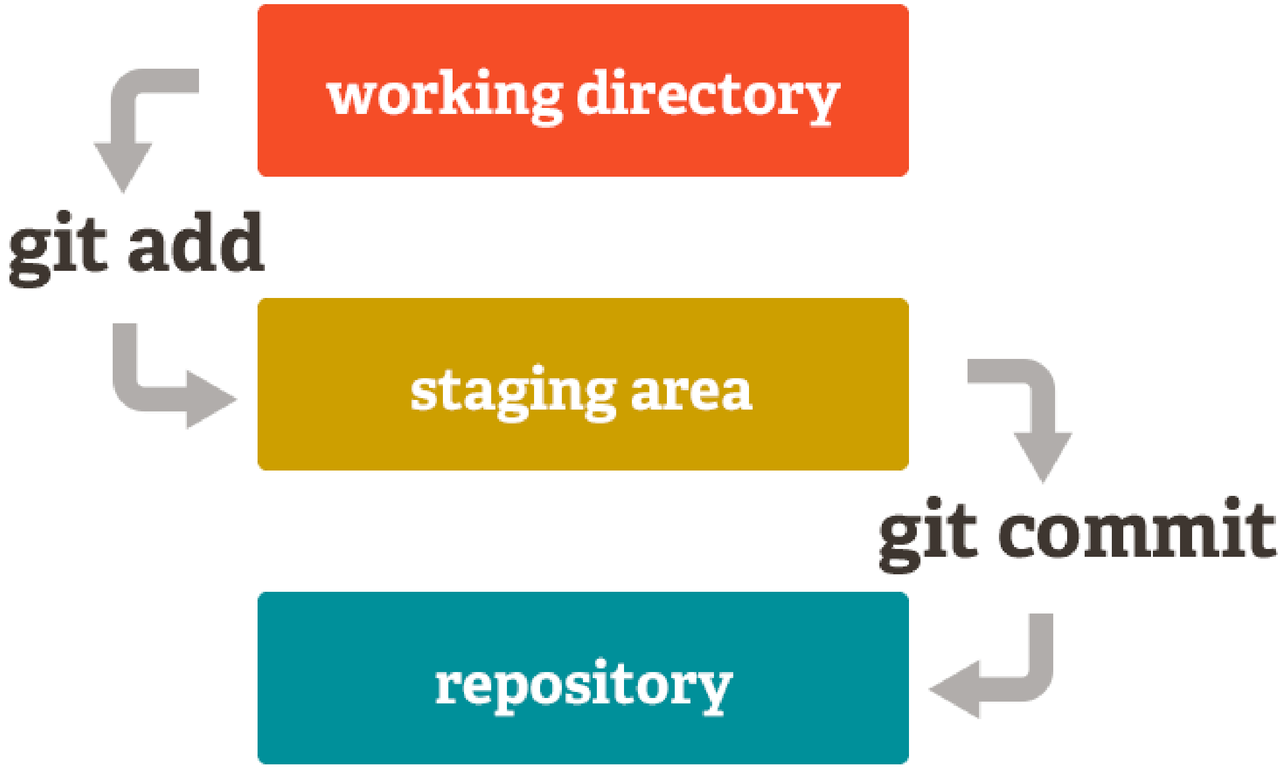


그림 git-scm.com

파일의 상태 라이프 사이클

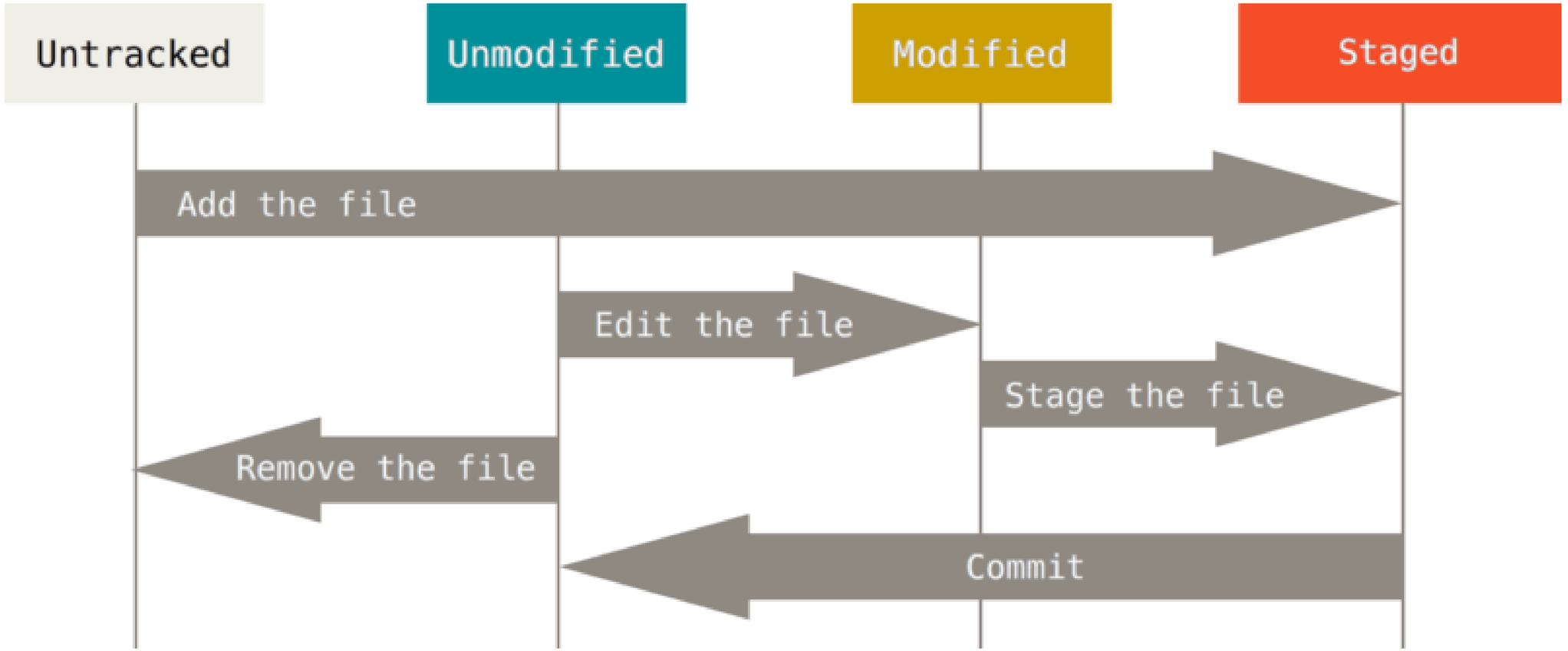


그림 git-scm.com

새로운 파일 생성

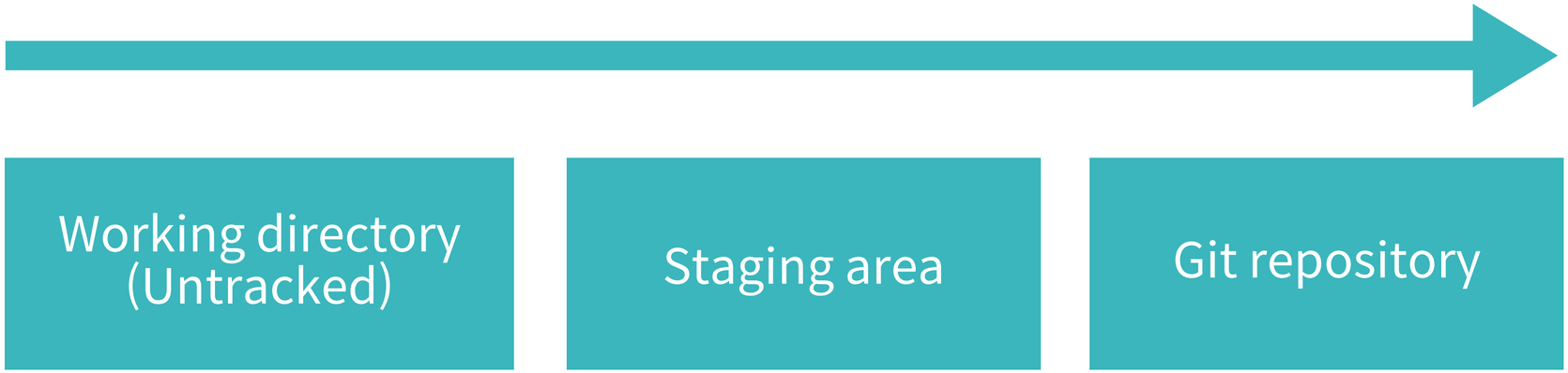
먼저, `comment.js` 파일을

준비영역으로 보내야합니다

`git add` 명령어를 사용합니다

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

세 가지 영역

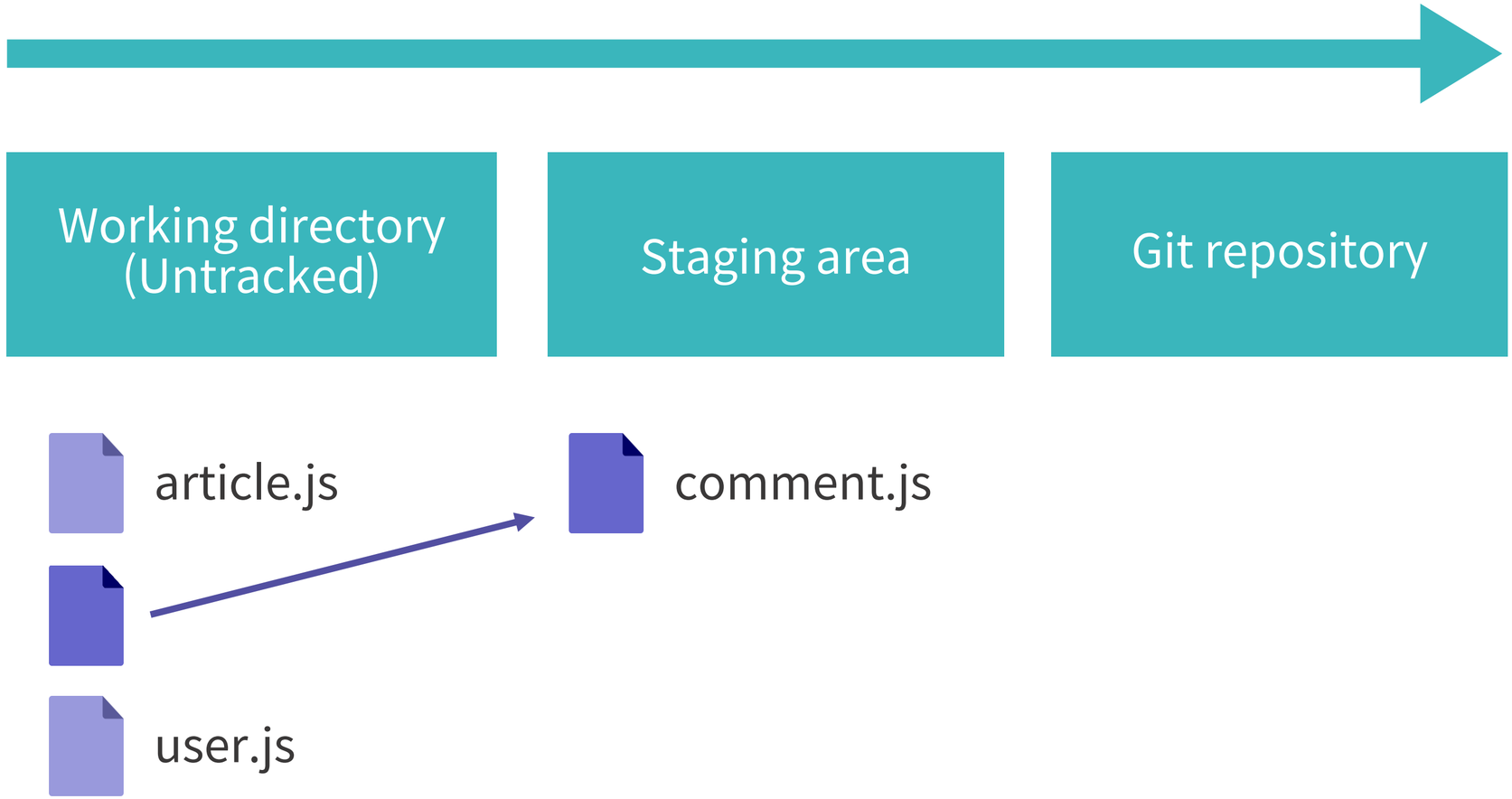


article.js

comment.js

user.js

세 가지 영역

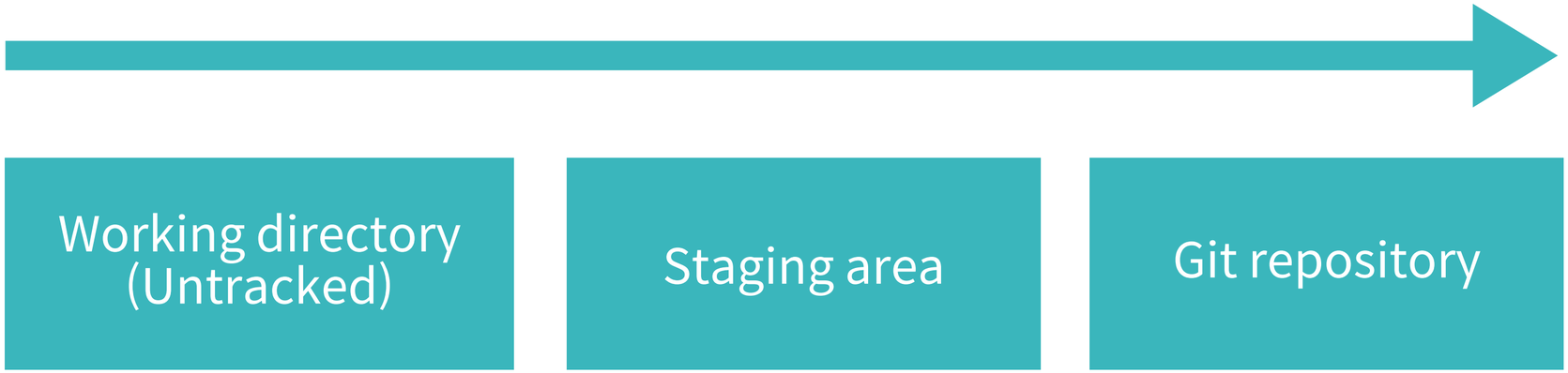

Working directory
(Untracked)
Staging area
Git repository
article.js
comment.js
user.js

새로운 파일 생성 (1)

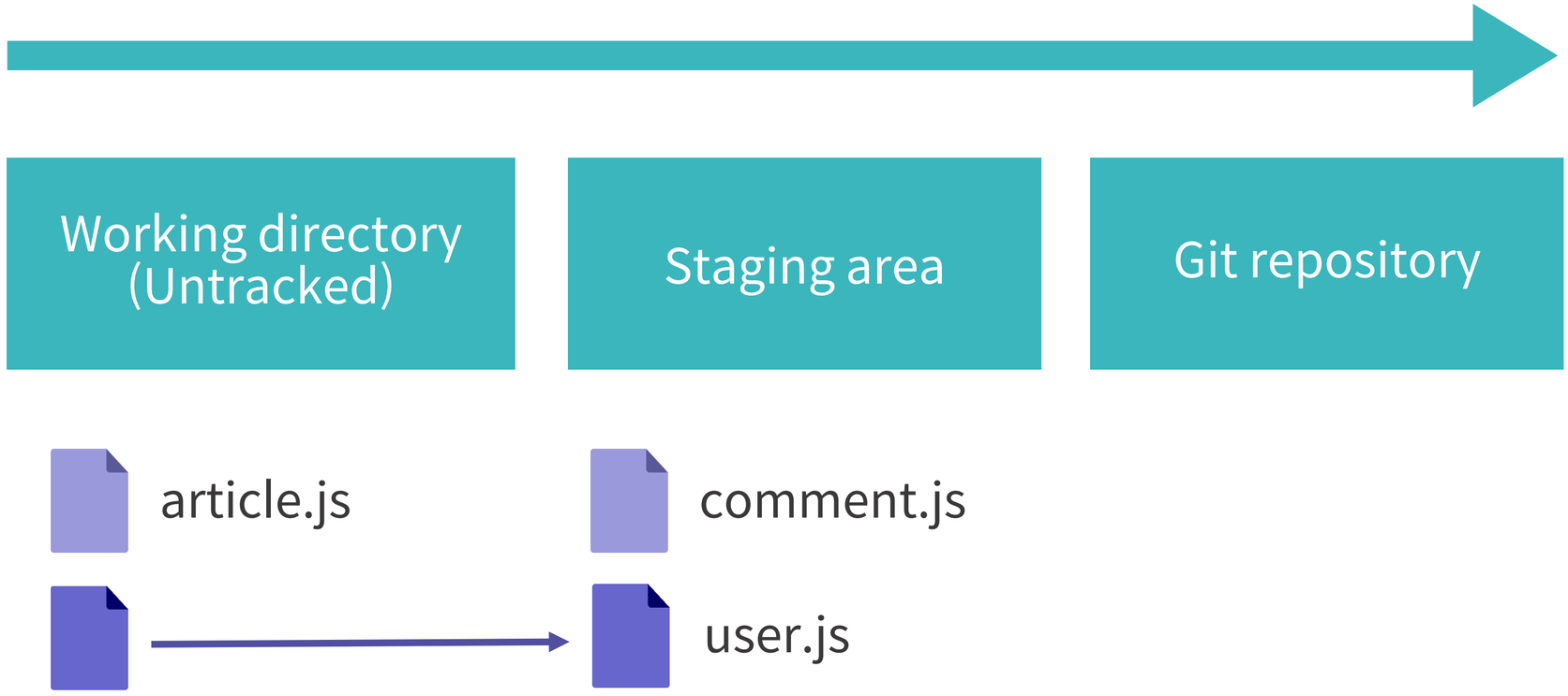
추가할 파일이 있다면 계속해서 더 추가할 수 있습니다

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

세 가지 영역


Working directory
(Untracked)
Staging area
Git repository
article.js
comment.js
user.js

세 가지 영역


Working directory
(Untracked)
Staging area
Git repository
article.js
comment.js
user.js

새로운 파일 생성 (2)

한 번에 추가할 파일이 너무 많다면
현재 폴더를 대상으로 지정할 수도 있습니다

```
$ git add .
```

Staging 상태 확인

`git status` 명령어로 Staging area의
어떤 파일이 변경되었는지 등의
파일의 상태를 확인 할 수 있습니다

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

      modified:   comment.js
```

Git 저장소 반영

Git 저장소 반영

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

Git 저장소 반영

```
git commit
```

.git 저장소 내에 staging 파일 저장

파일 영역의 라이프 사이클

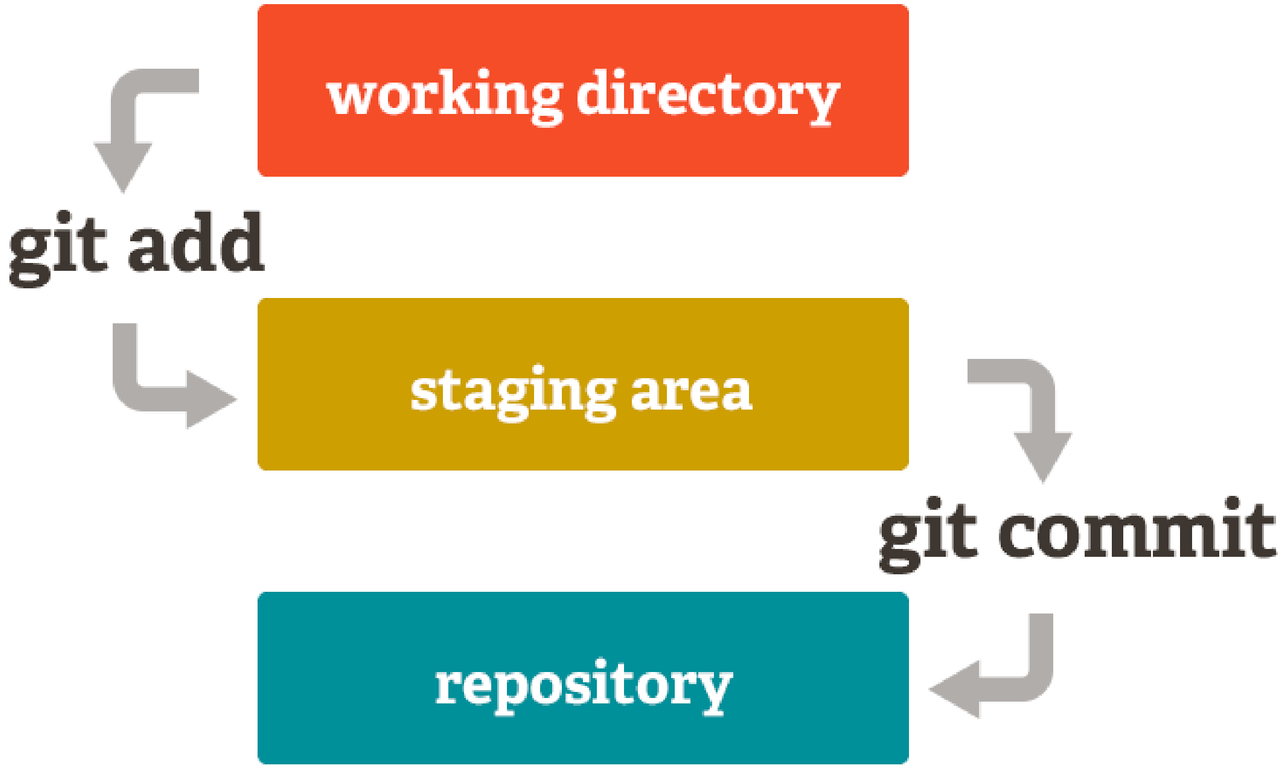


그림 git-scm.com

파일 상태의 라이프 사이클

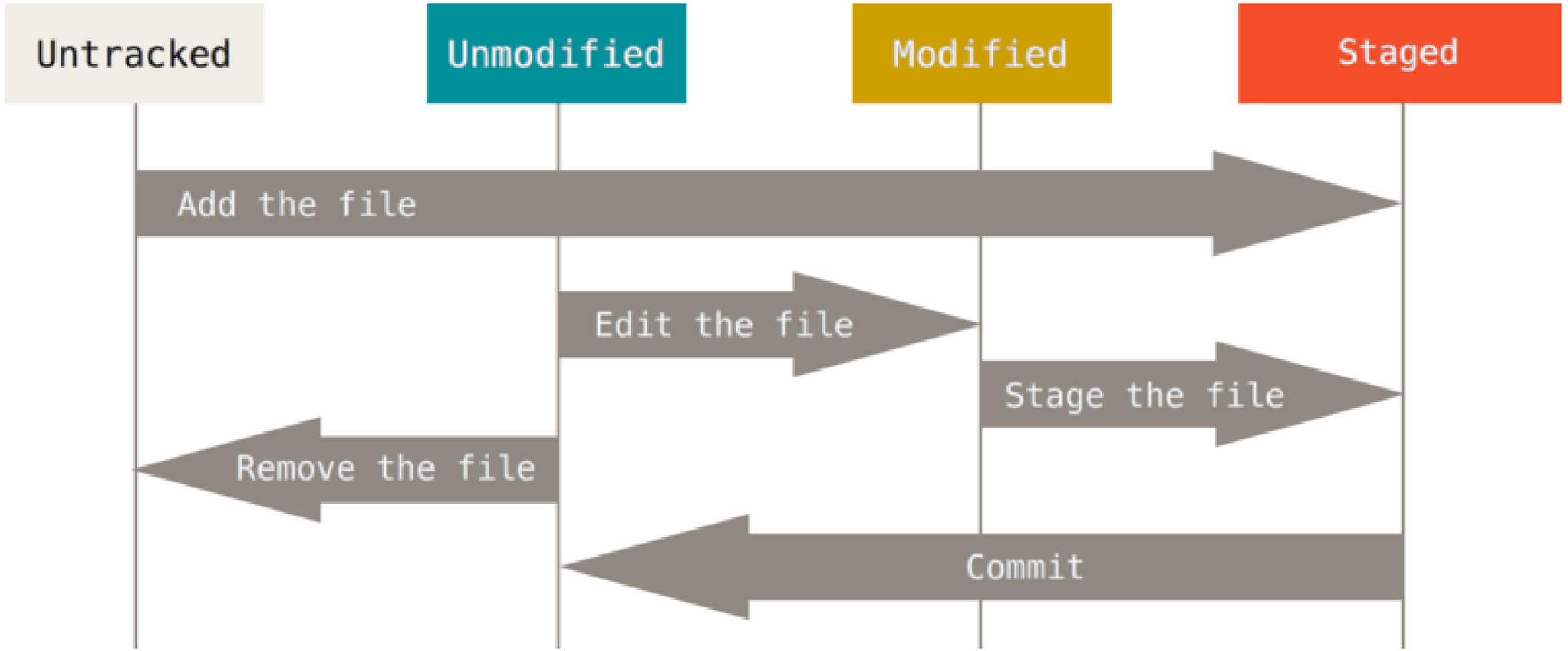


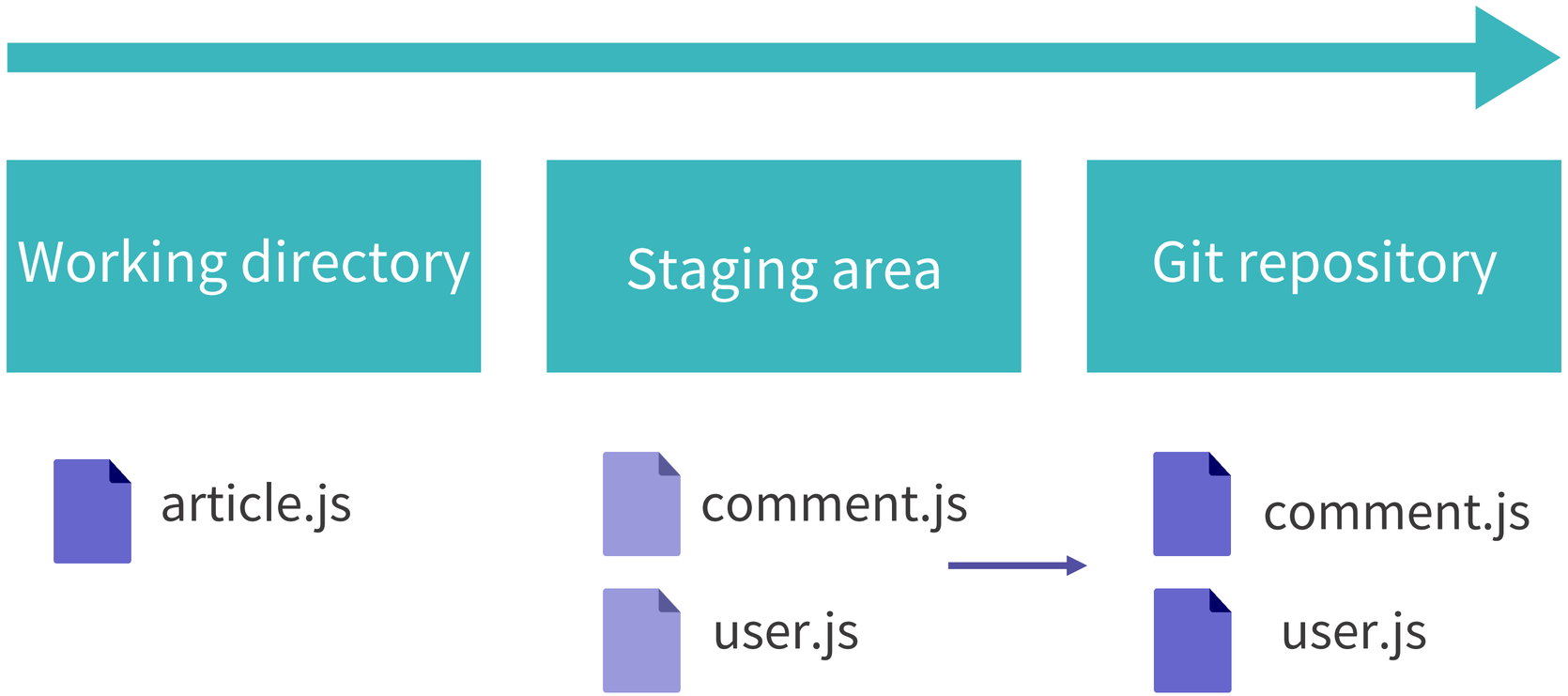
그림 git-scm.com

저장소 반영

준비 영역에 있는 파일들을 저장소에 반영하겠습니다.
생략 가능하지만 반영한 내용을 추후에 쉽게 알 수 있도록
적절한 메세지를 넣어주세요.

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

세 가지 영역


Working directory
Staging area
Git repository
article.js
comment.js
user.js
comment.js
user.js

저장소 반영 내용 변경

앞에서 적은 메세지에 오타가 있거나
누락된 파일이 있을 경우 걱정하지 마세요.

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

저장소 반영 내용 변경

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

```
Initial commit

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

저장소 반영 내역

저장소 반영 내역이 궁금하시다면
아래 명령어를 사용해보세요.

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

    Initial commit
```

Git history

Commit 된 내용은 다음과 같습니다.

d78ad

```
Commit size
Tree
Parent
Author
Commiter

Initial commit
```

Git 관리 상태 확인

파일 상태 확인

Git 저장소까지 저장을 완료하였습니다.

파일들의 상태와 history를 볼 수 있는

`git status`, `git log` 에 대해 자세히 살펴보겠습니다.

Git 관리 상태 확인

`git status`

Staging file들의 상태 확인

`git log`

.git repository에 존재하는 history 확인

파일 영역의 라이프 사이클

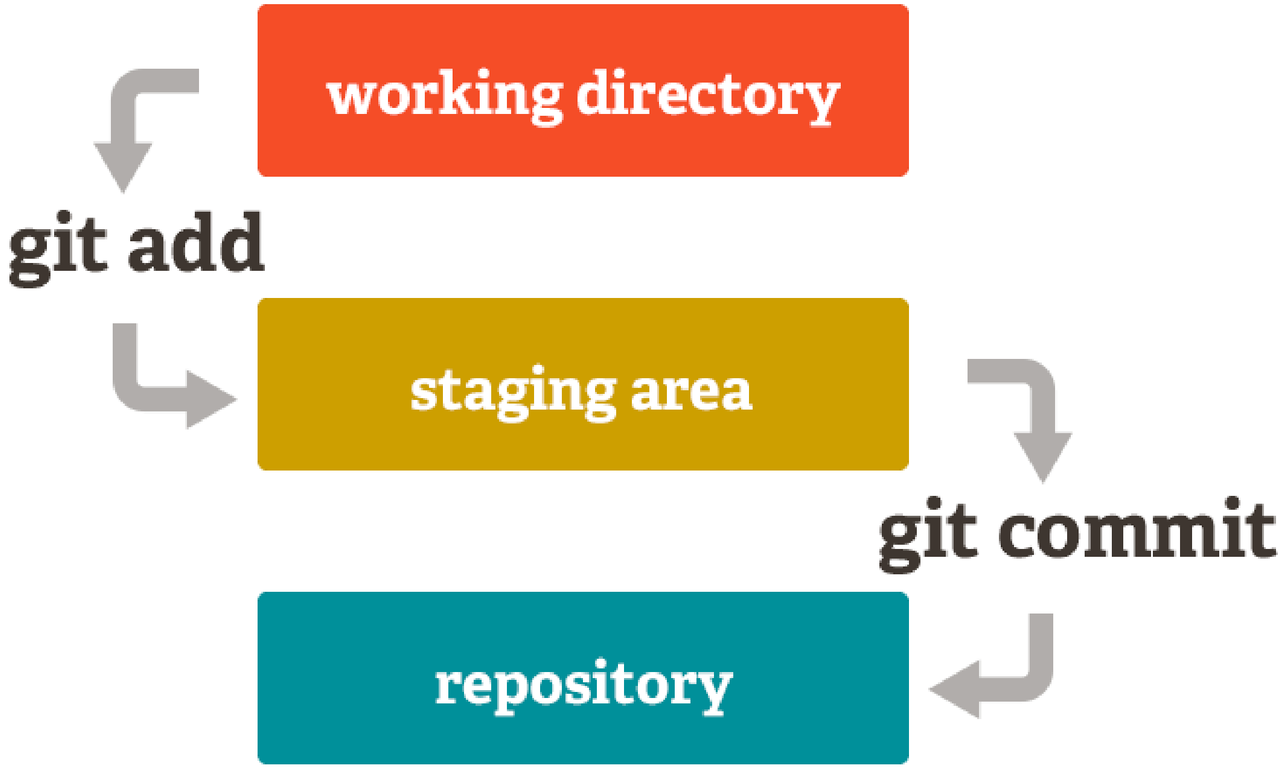


그림 git-scm.com

파일 상태의 라이프 사이클

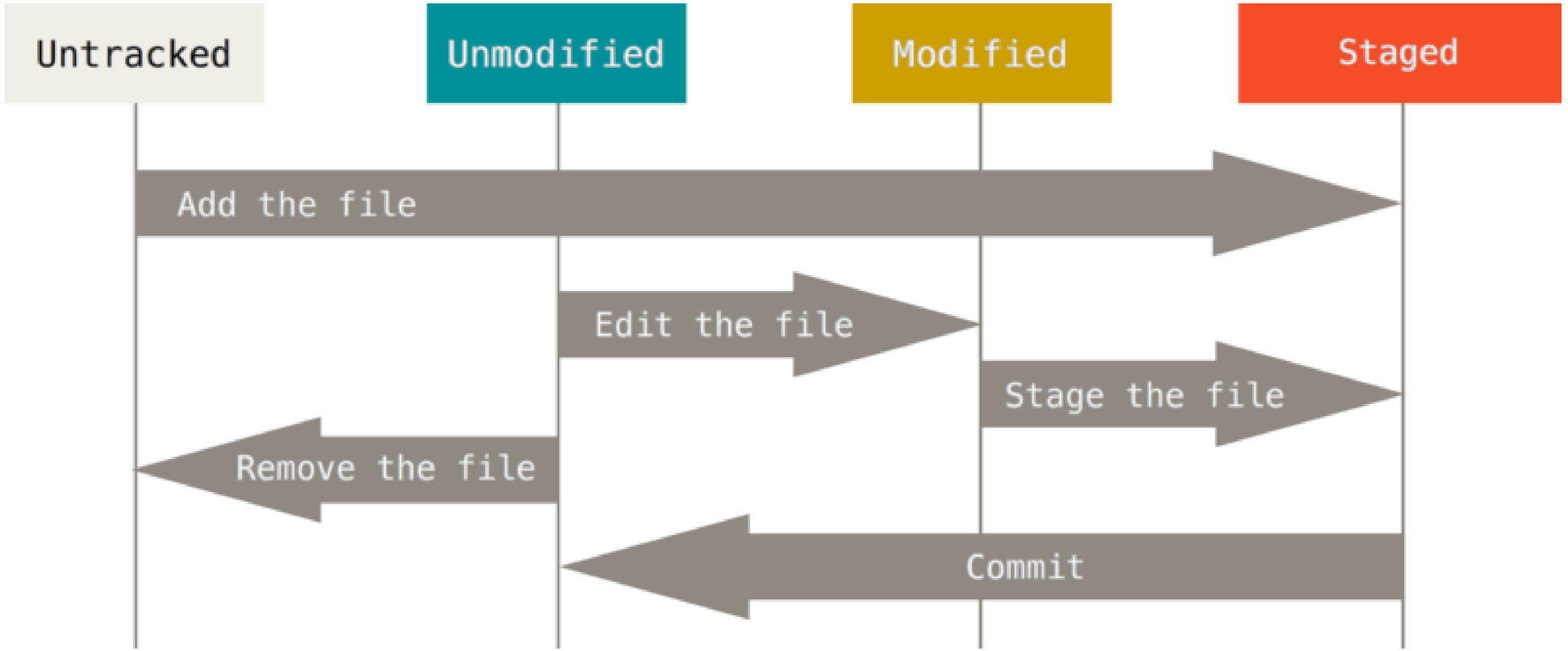


그림 git-scm.com

세 가지 영역

Working directory

Staging area

Git repository

comment.js

article.js

user.js

status

Untracked : add로 staging되지 않은 파일 들

```
  $ git status
  On branch master

  No commits yet

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

세 가지 영역

Working directory

Staging area

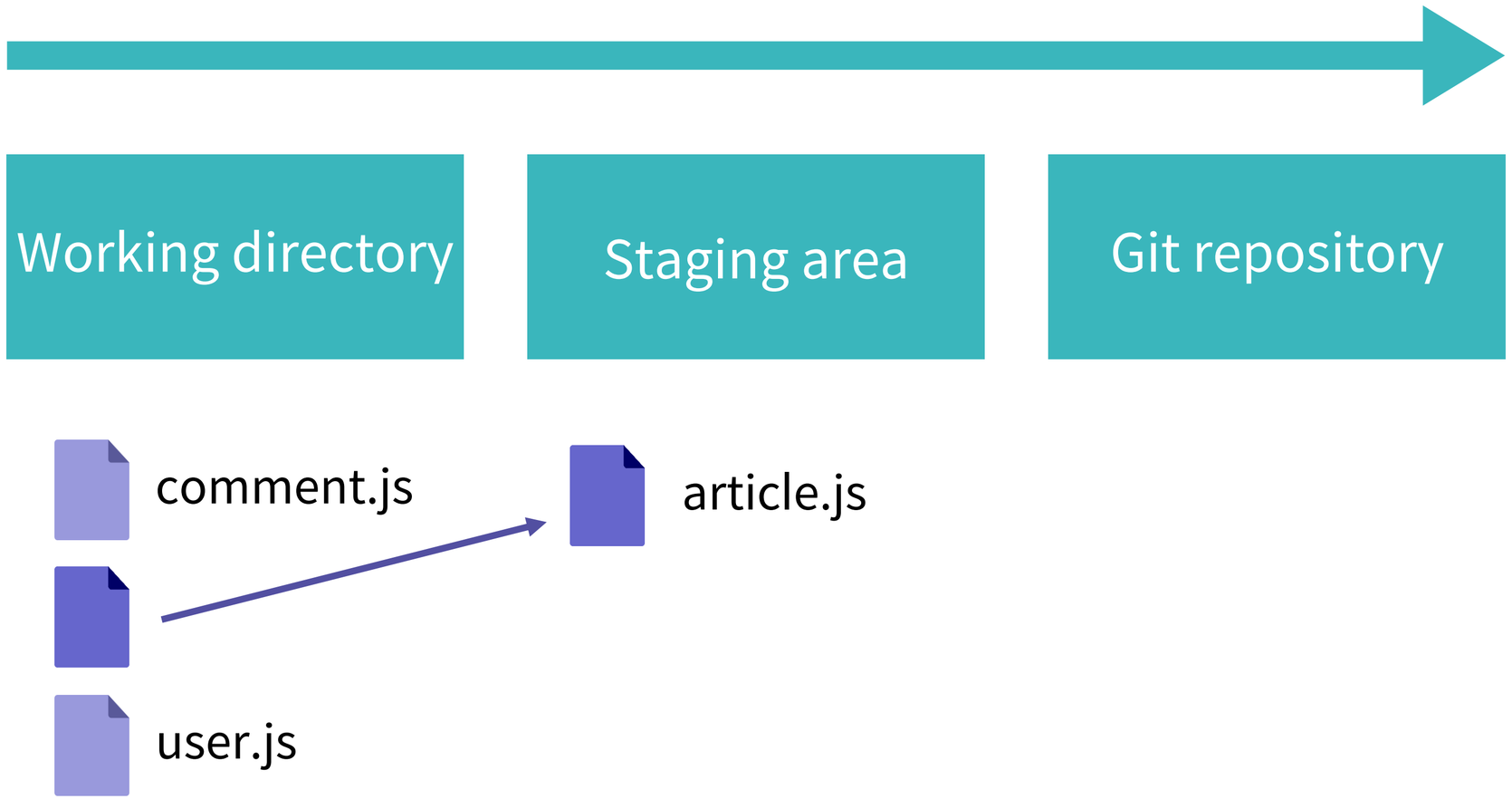
Git repository

comment.js

article.js

user.js

세 가지 영역


Working directory
Staging area
Git repository
comment.js
article.js
user.js

status

```
$ 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 commited)
            comment.js
            user.js
```

status

`git reset <file_name>`

위의 명령을 이용하여 add 명령을 취소할 수 있습니다.

이제 commit된 파일을 수정할 경우 git이 파일을 어떤 상태로 인식하는지 확인해 보겠습니다.

`article.js`를 commit 해주세요

세 가지 영역

Working directory

Staging area

Git repository

comment.js

user.js

(수정된)article.js

article.js (a522b)

status

modified : commit 된 파일 중 수정된 파일이 있을 경우

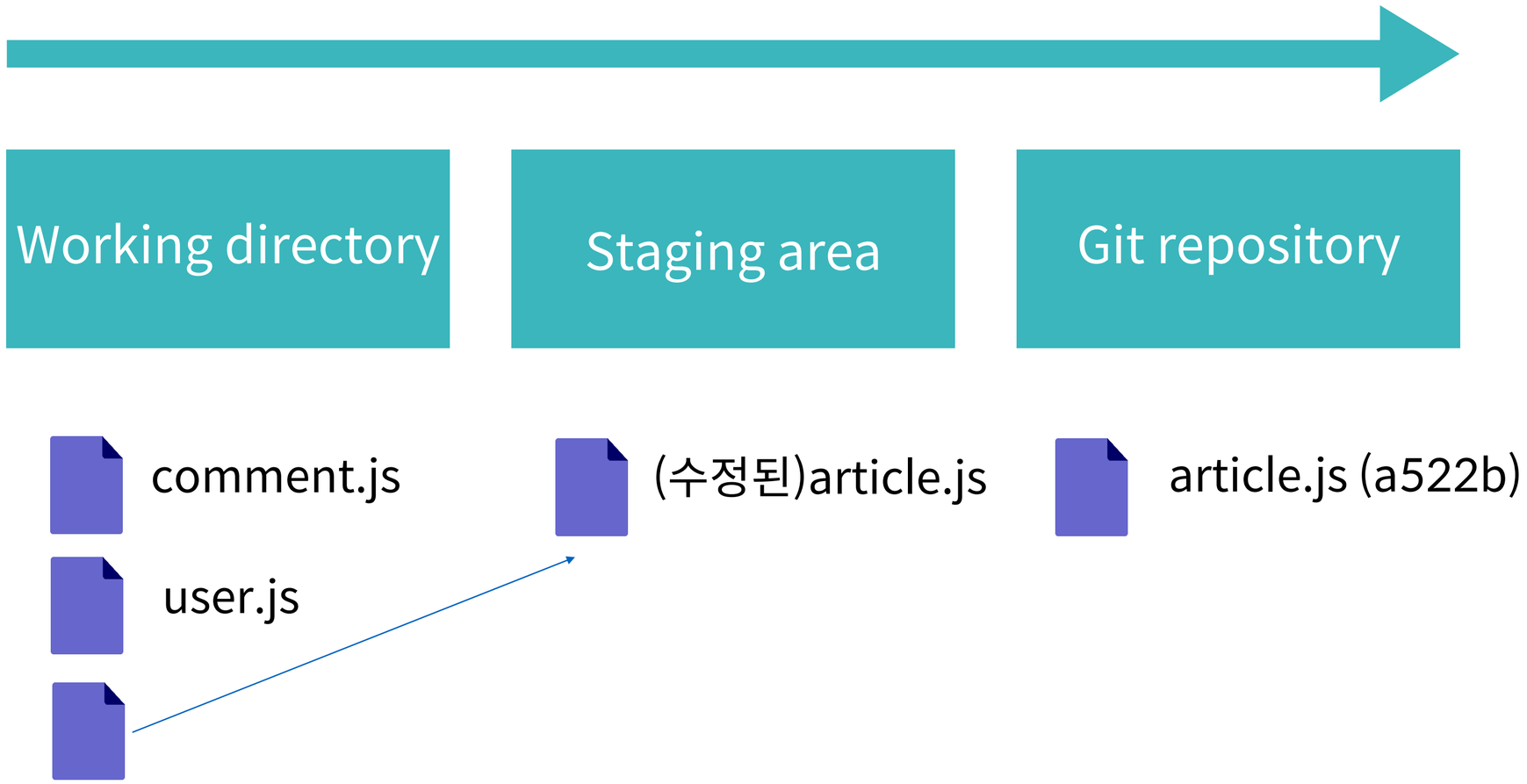
```
$ git status
On branch master
changes not staged for commit:
(use “git add<file>..” to update what will be commited)
(use “git checkout -- <file>..” to discard changes in working ..)

            modified : article.js

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

status

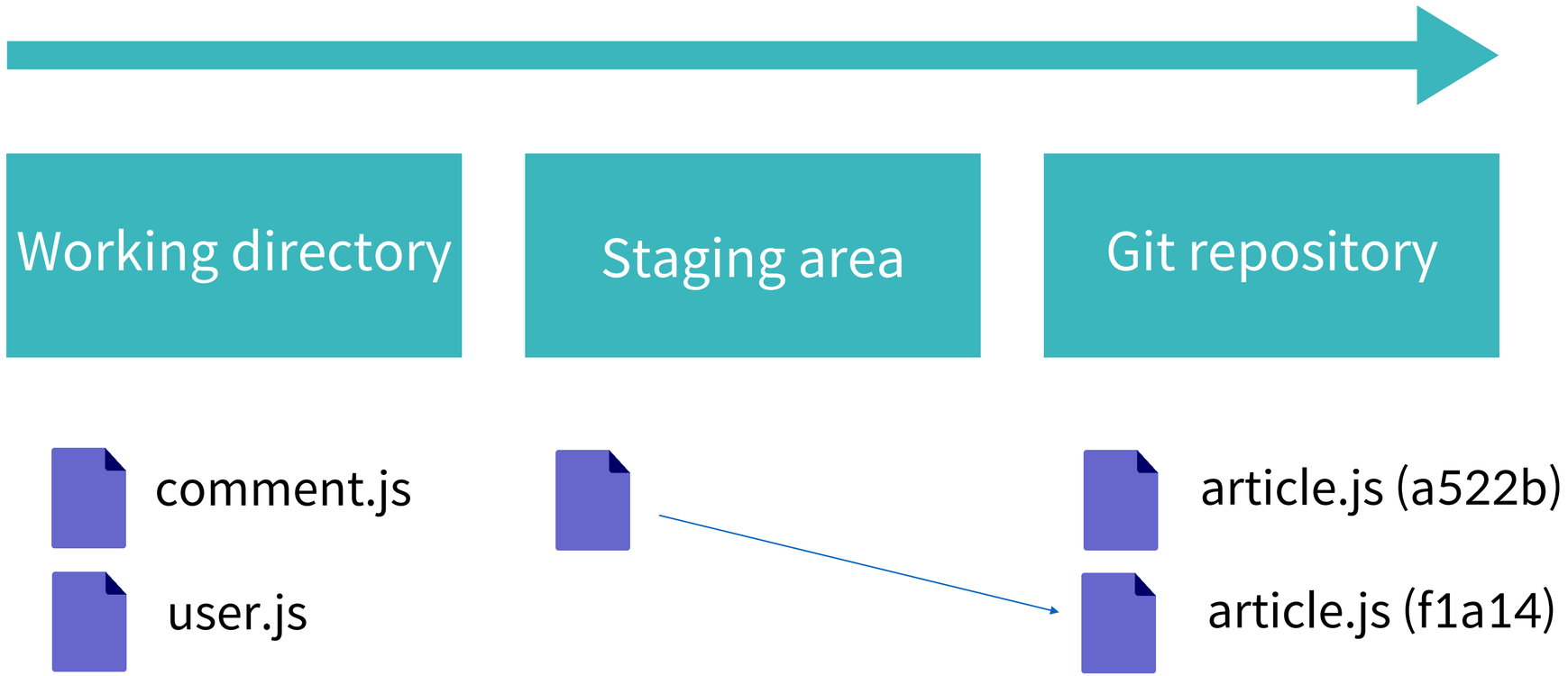
`git diff` : commit 된 파일 중 변경된 사항을 비교할때

```
$ git diff
diff --git a/article.js b/article.js
index e87ce30..a752541 100644
--- a/article.js
+++ b/article.js
@@ -1 +1 @@
-#let
+#trysome
```

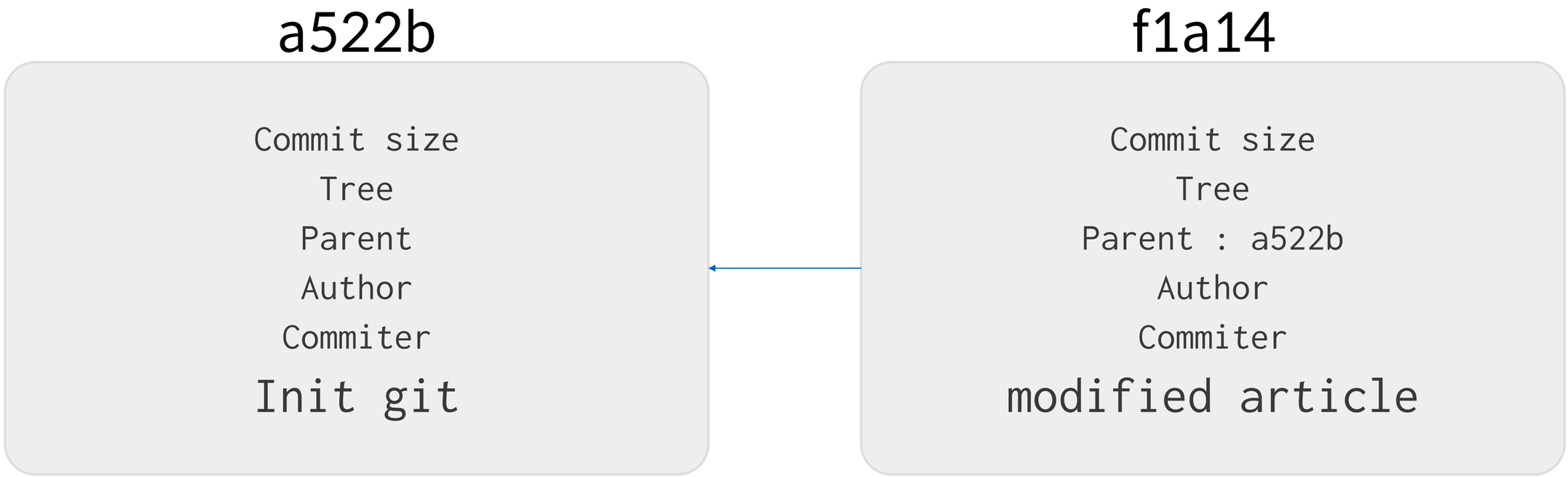
세 가지 영역


Working directory
Staging area
Git repository
comment.js
user.js
(수정된)article.js
article.js (a522b)

세 가지 영역


Working directory
Staging area
Git repository
comment.js
user.js
article.js (a522b)
article.js (f1a14)

Git history


a522b
Commit size
Tree
Parent
Author
Commiter
Init git
f1a14
Commit size
Tree
Parent : a522b
Author
Commiter
modified article

log

저장소 반영 내역을 확인하기 위하여 log 명령어를 사용합니다

```
$ git log
commit f1a14ec47cf07fcf915e161e30f4a7b7b6752fec
Author: unknown <unknown@unknown.me>
Date:   Tue Jul 30 14:52:43 2019 +0900

    modified article

commit a522b90e409b5f957e41c1e3e0a0206869a84696
Author: unknown <unknown@unknown.me>
Date:   Tue Jul 30 14:17:04 2019 +0900

    init git
```

대표적인 log 옵션들

```
git log -p -2
```

-p, --patch : 각 commit 의 수정 결과를 보여주는 diff와 같은 역할을 수행합니다

-n : 상위 n개의 commit만 보여줍니다

대표적인 log 옵션들

```
git log --stat
```

--stat : 어떤 파일이 commit에서 수정되고 변경되었는지,
파일 내 라인이 추가되거나 삭제되었는지 확인

```
git log --pretty=oneline
```

--pretty=oneline : 각 commit을 한 줄로 출력

대표적인 log 옵션들

```
git log --graph
```

--graph: commit간의 연결된 관계를
아스키 그래프로 출력한다.

```
git log -S function_name
```

-S : 코드에서 추가되거나 제거된 내용 중 특정 텍스트
(위에서는 funtion_name)가 포함되어 있는지 검사


/*elice*/
contact@elice.io