getting started with git

김덕홍 연구원

2015-03-27

실습에 사용 될 sample code

http://goo.gl/w0zk7w

Branch

Branch 는 왜 필요할까?

새로운 기능 추가 작업중

58a7de3 - A기능 사용자 입력 버그 픽스

5bb77b1 - A기능 사용자 화면

e351322 - A기능 서버 로직 구현

6b53ccc - A기능을 위한 DB 스키마 추가

33842b6 - v2.1.0 Release

그러던 어느날....

v2.1.0 Release 버전에
버그가 있네요 고쳐주세요

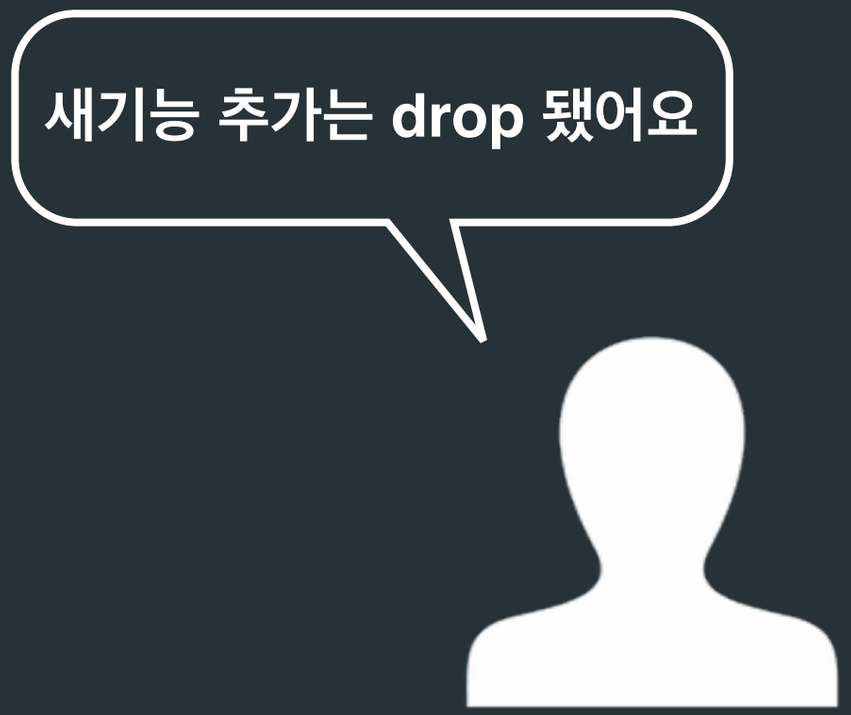
테스터

ㅇㅋㅇㅋ

개발자

이미 새로운 기능 추가를 위해 수정된 파일들

또는 이미 수정 중이던 작업을 되돌리게 되었을 때



기획자

개발자

어떤 경우에도 원본은
항상 유지 되고 있어야 한다.

SVN 에서 Branch

Directory Copy

Trunk

Feature

Git 용어 설명

branch

원래 코드와는 상관없이 독립적으로 개발을 진행해 나가는 것

branch name

특정 커밋을 가리키는 alias name (.git/refs/)

HEAD

특정 커밋 혹은 branch를 가리키는 레퍼런스 (.git/HEAD)

<Mini Project>
ToolCon 2015 official site 구축

toolcon 사이트를 만들기 위한 이슈들

- 컨퍼런스 사이트에 맞게 READMD.md 를 수정 한다.
- 티저 사이트를 만든다.
- 강사와 세션이 확정되면 티저 사이트에 업데이트 한다.
- 컨퍼런스가 끝나면 발표자료 링크를 사이트에 업데이트 한다.

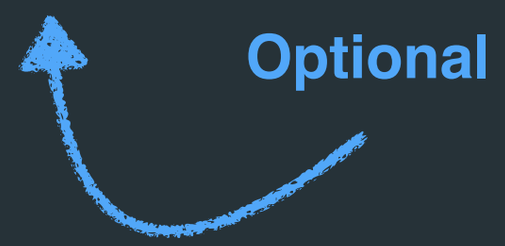
Git branch command

branch 목록 확인

```
$ git branch
```

branch 만들기

```
$ git branch (branch_name) [base pointer]
```



branch 삭제하기

```
$ git branch -D (branch_name)
```

git branch 실습

```
~/git-repo $  git branch

~/git-repo $  cat .git/refs/heads/master

~/git-repo $  git branch readme //readme 수정을 위한 branch

~/git-repo $  git branch

~/git-repo $  cat .git/refs/heads/readme
```

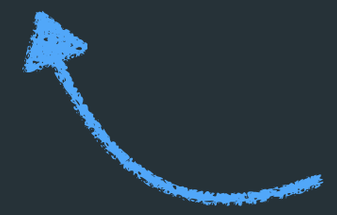
Git checkout command

지정한 branch 로 변경

```
$ git checkout (branch_name)
```

새로운 branch를 만들고 새로 만들어진 branch 로 변경

```
$ git checkout -b (branch_name) [base pointer]
```



git checkout 실습

```
~/git-repo $  git checkout -b teaser //티저 사이트를 위한 branch

~/git-repo $  cat .git/refs/heads/teaser

~/git-repo $  git branch

~/git-repo $  cat .git/HEAD

~/git-repo $  git checkout readme

~/git-repo $  cat .git/HEAD

~/git-repo $  git checkout master

~/git-repo $  cat .git/HEAD
```

branch 를 변경하면 일어나는 일들

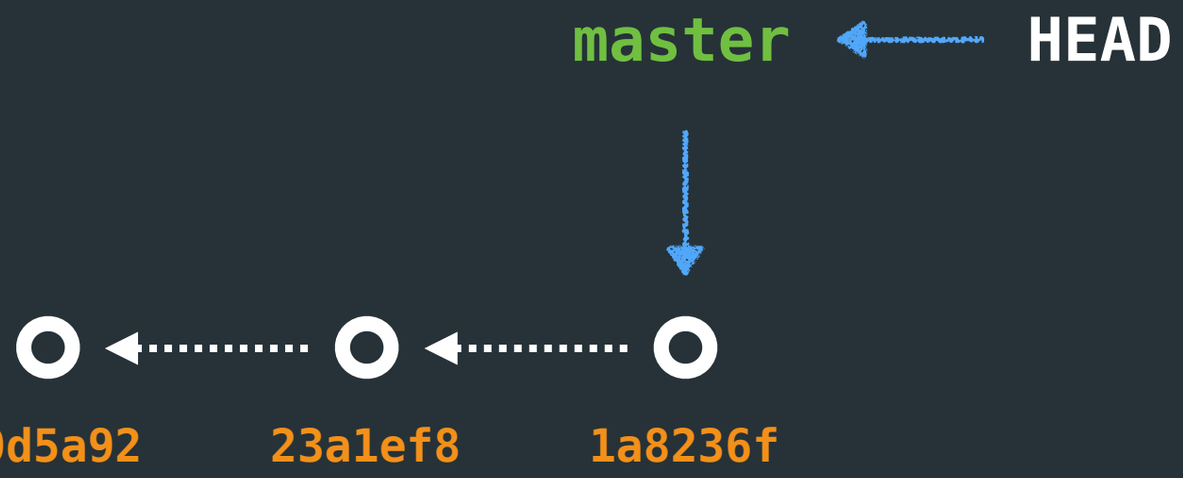
- index 와 working directory의 파일들이 변경된 branch를 기준으로 switching
- HEAD 가 가리키는 branch 가 변경됨

branch 를 만드는 기준은?

- 정해진 기준이 있는 것은 아님
- 운영 및 개발에 필요한 branch 를 결정
- topic 별 branch 를 만들어 작업하는 것을 권장

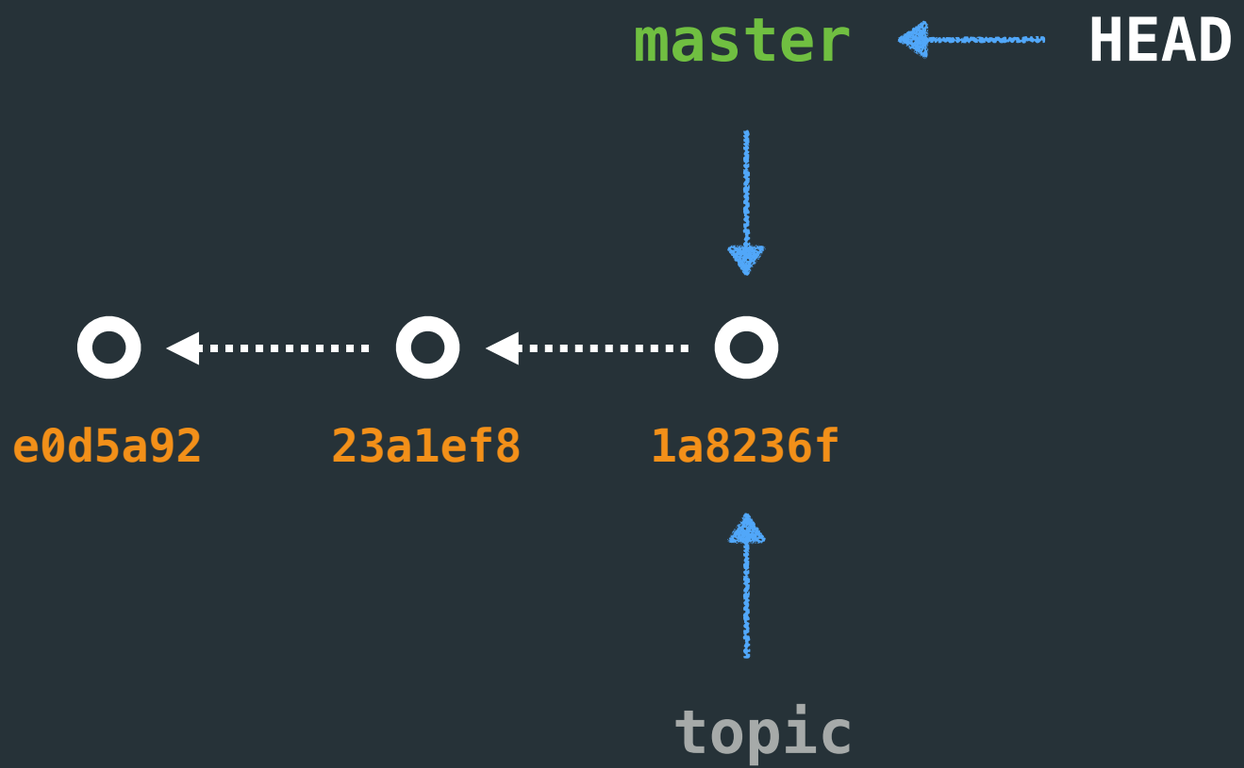
branch workflow

~/git-repo $ **git checkout master**



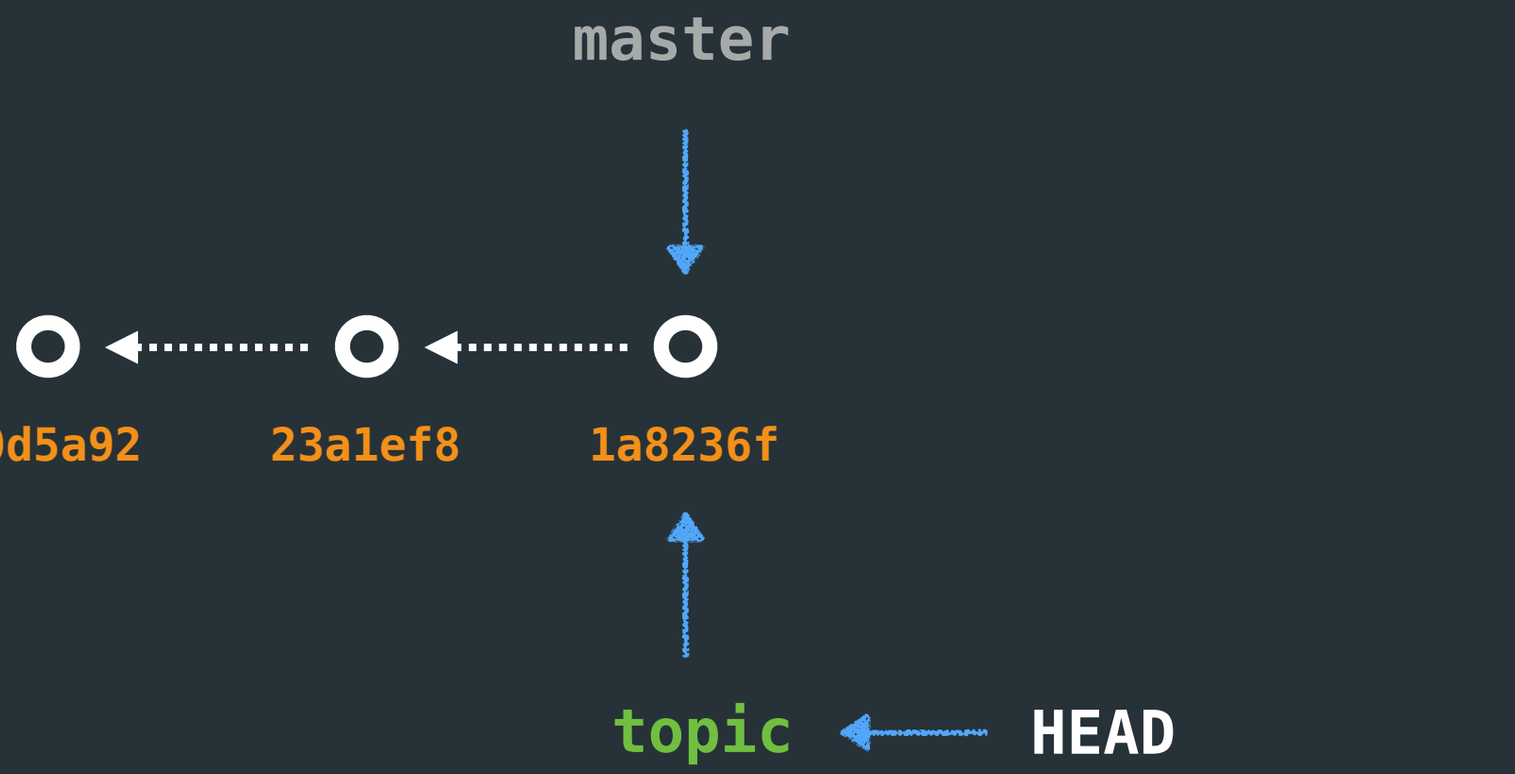
branch workflow

~/git-repo $ **git branch topic**



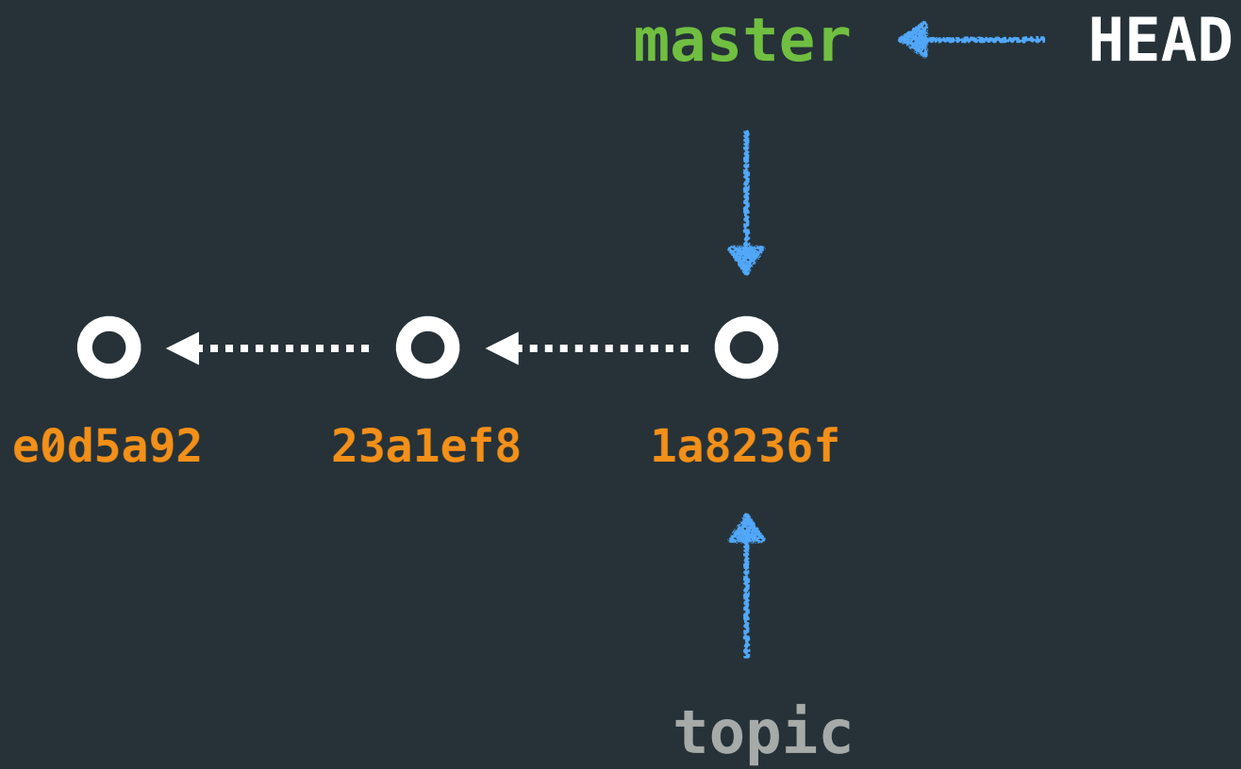
branch workflow

~/git-repo $ **git checkout topic**



branch workflow

~/git-repo $ **git checkout master**



Merge

Git merge command

merge 하기

```
$ git merge (target_branch_name)
```

주의할 점

코드가 반영될 branch 에서 merge 를 수행해야 한다.

git merge 실습 시나리오

- 컨퍼런스 사이트에 맞게 READMD.md 를 수정하는 이슈를 해결
- readme branch 에서 작업 후 master branch 로 merge
- merge 된 이후 readme branch 는 삭제

```
## ToolCon 2015
This is official site for ToolCon 2015

## LICENSE
MIT License

## AUTHOR
ToolCon 2015 organization
```

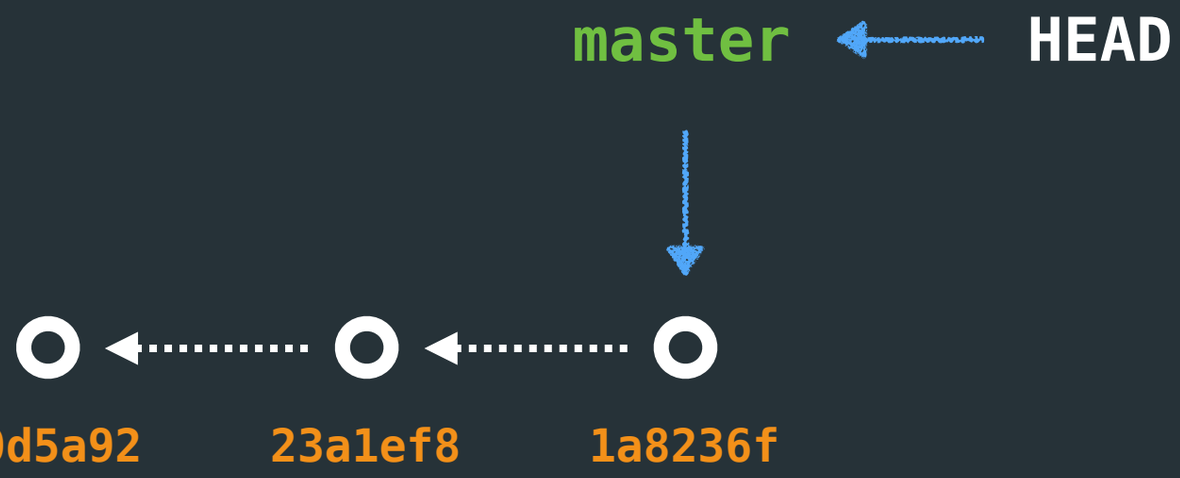
git merge 실습

```
~/git-repo $  git checkout readme

~/git-repo $  vi README.md

~/git-repo $  git add README.md

~/git-repo $  git commit

~/git-repo $  git checkout master

~/git-repo $  git merge readme

~/git-repo $  git branch -D readme
```

fast-forward merge

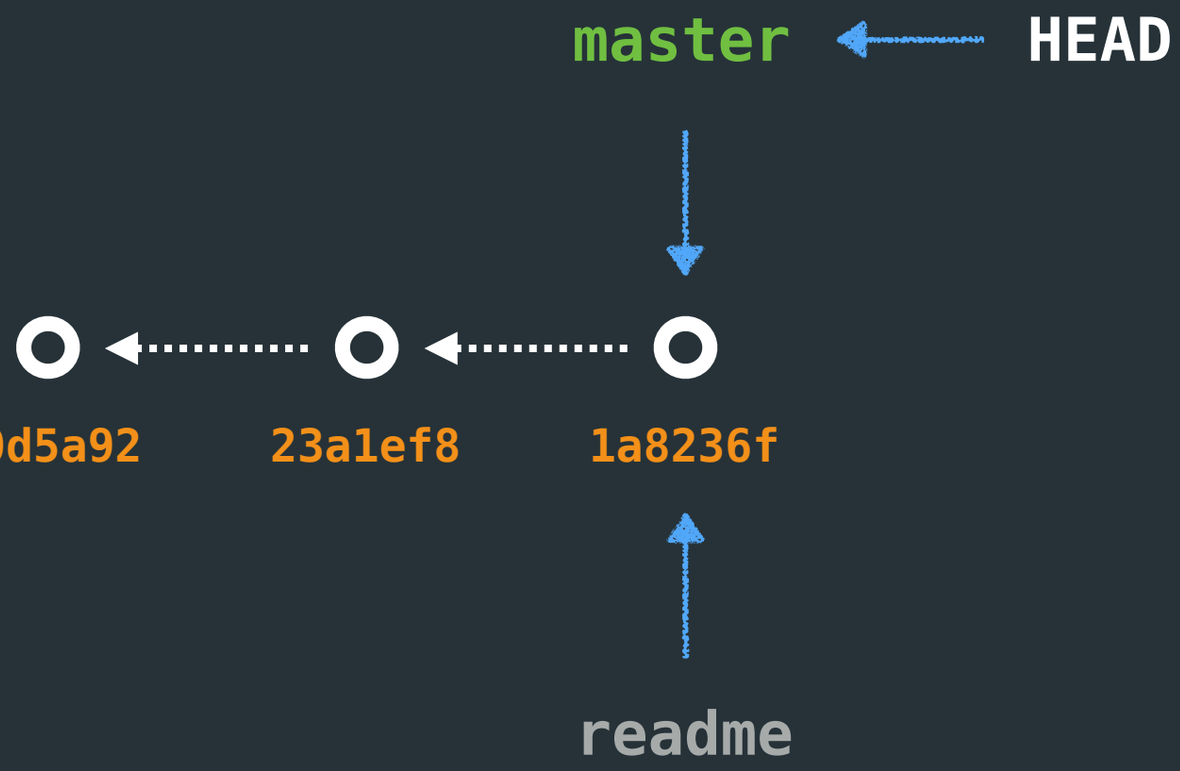
“현재 branch(수정된 코드가 반영될 branch)의 커밋이
변경이 일어난 branch (merge 를 하려는 branch)의
base commit 과 동일한 경우”

fast-forward merge workflow



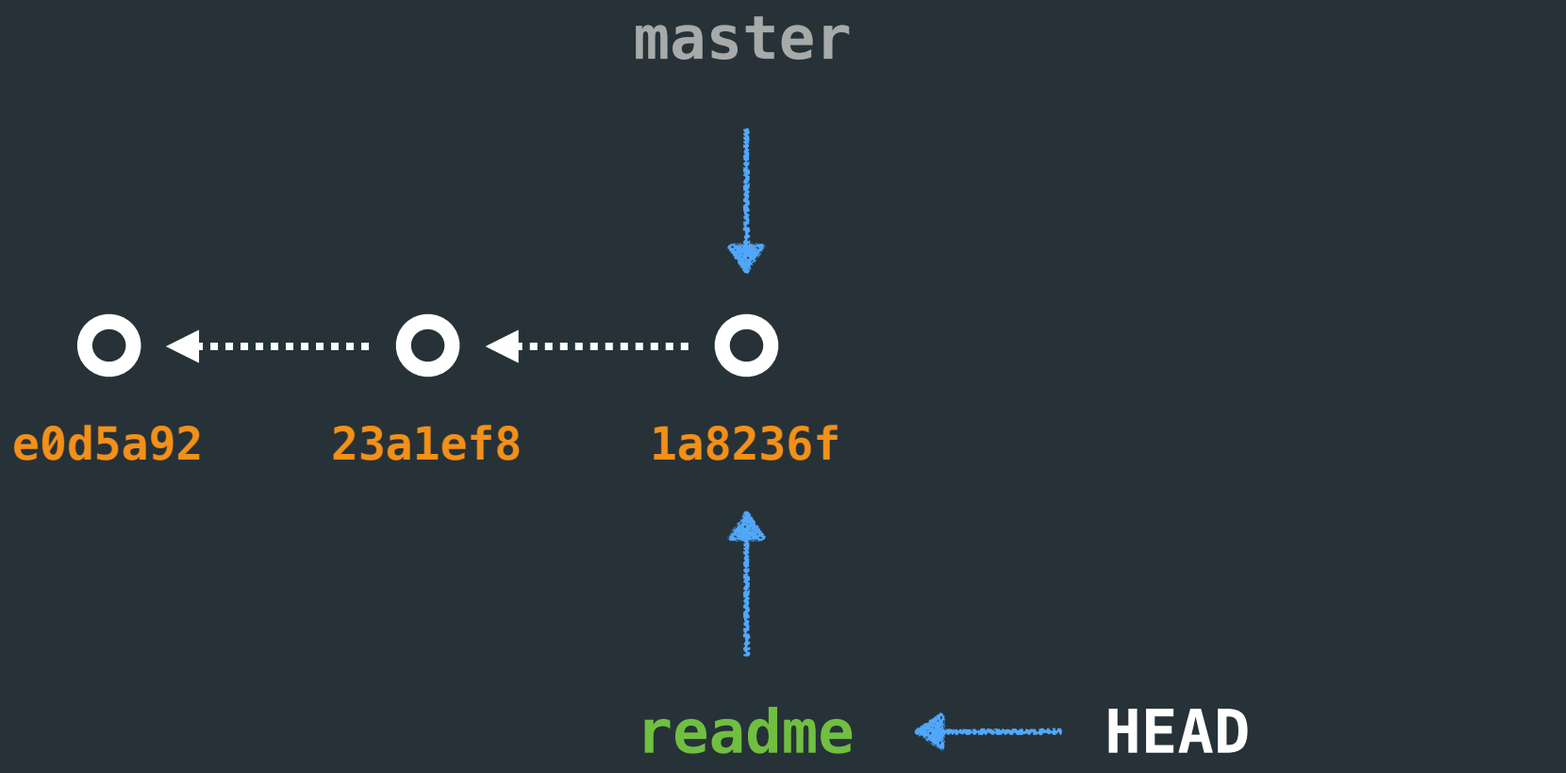
fast-forward merge workflow

~/git-repo $ **git branch readme**



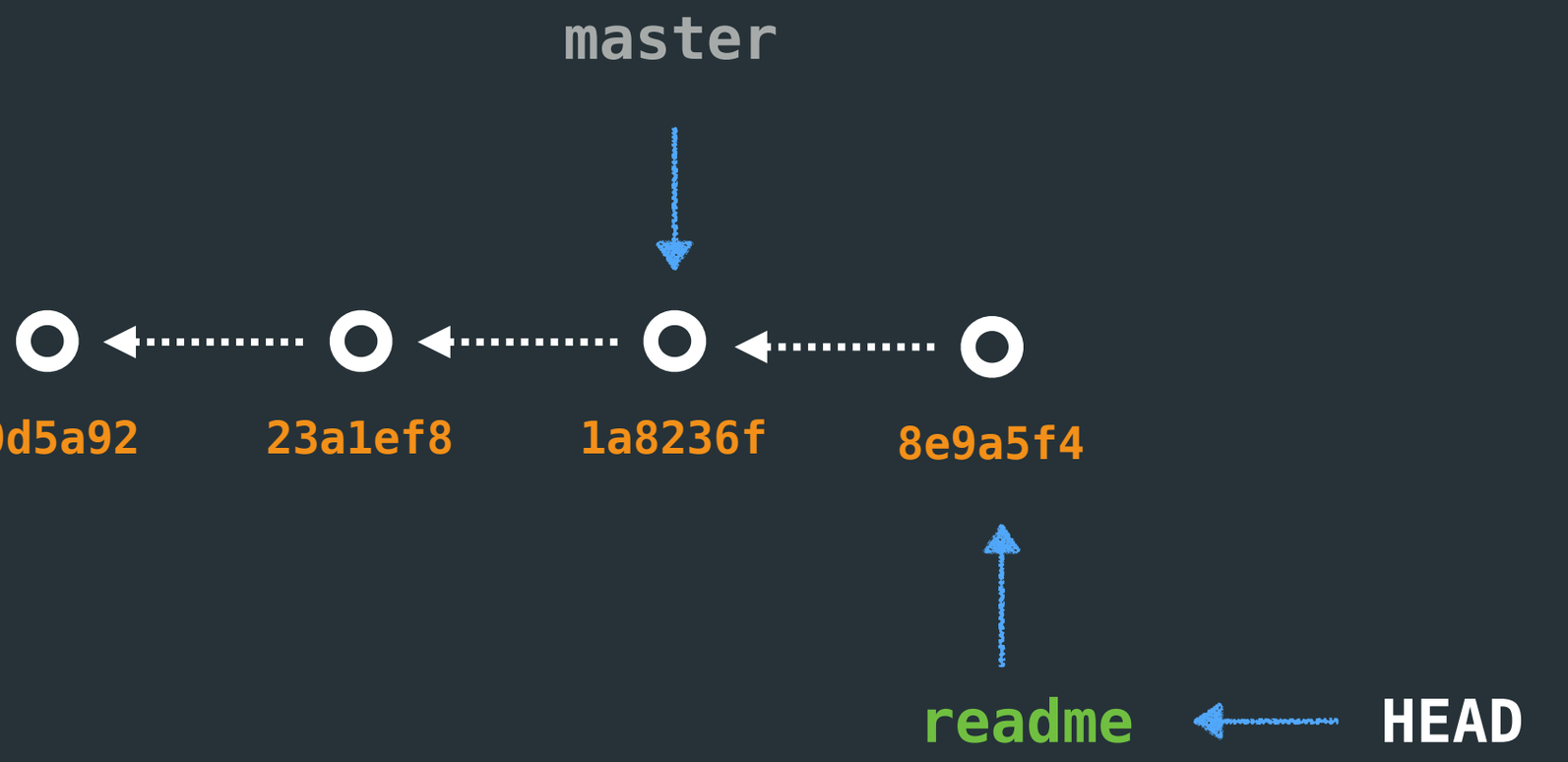
fast-forward merge workflow

~/git-repo $ **git checkout todo**



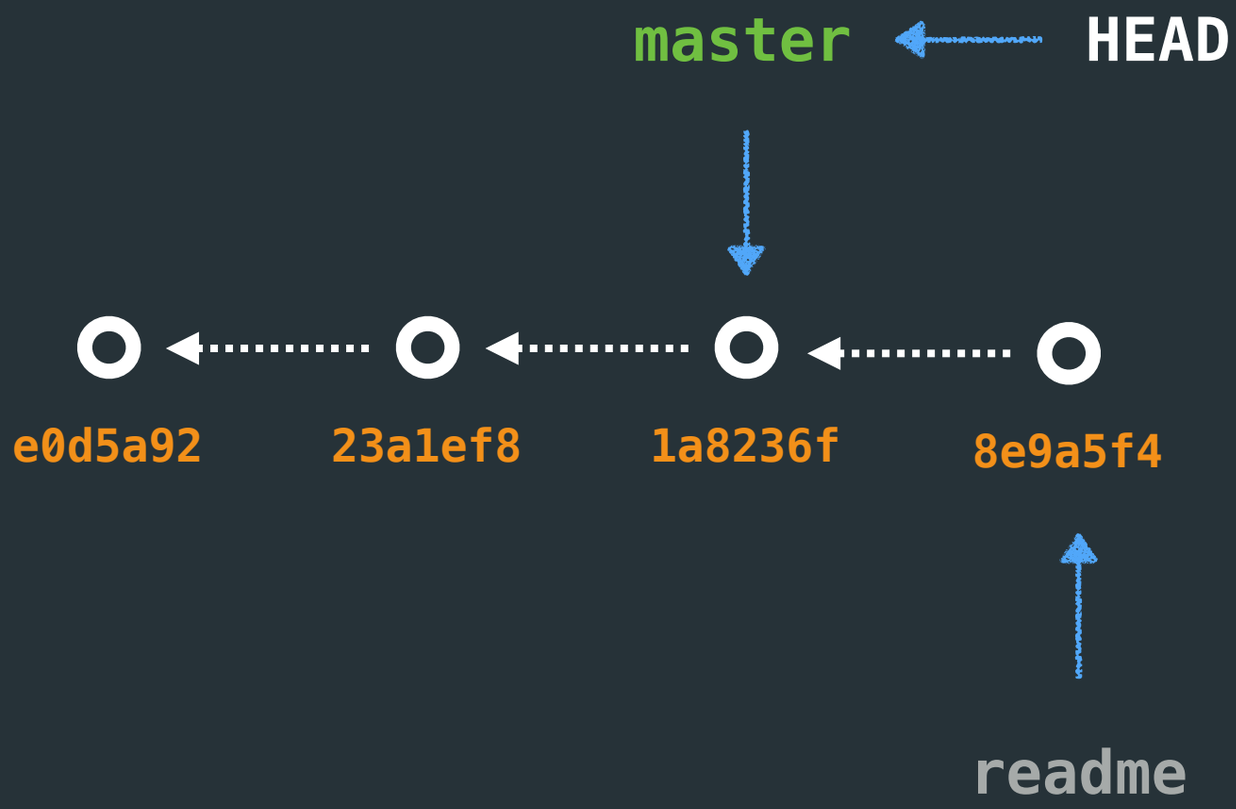
fast-forward merge workflow

~/git-repo $ **git commit -m “update readme for toolcon 2015"**



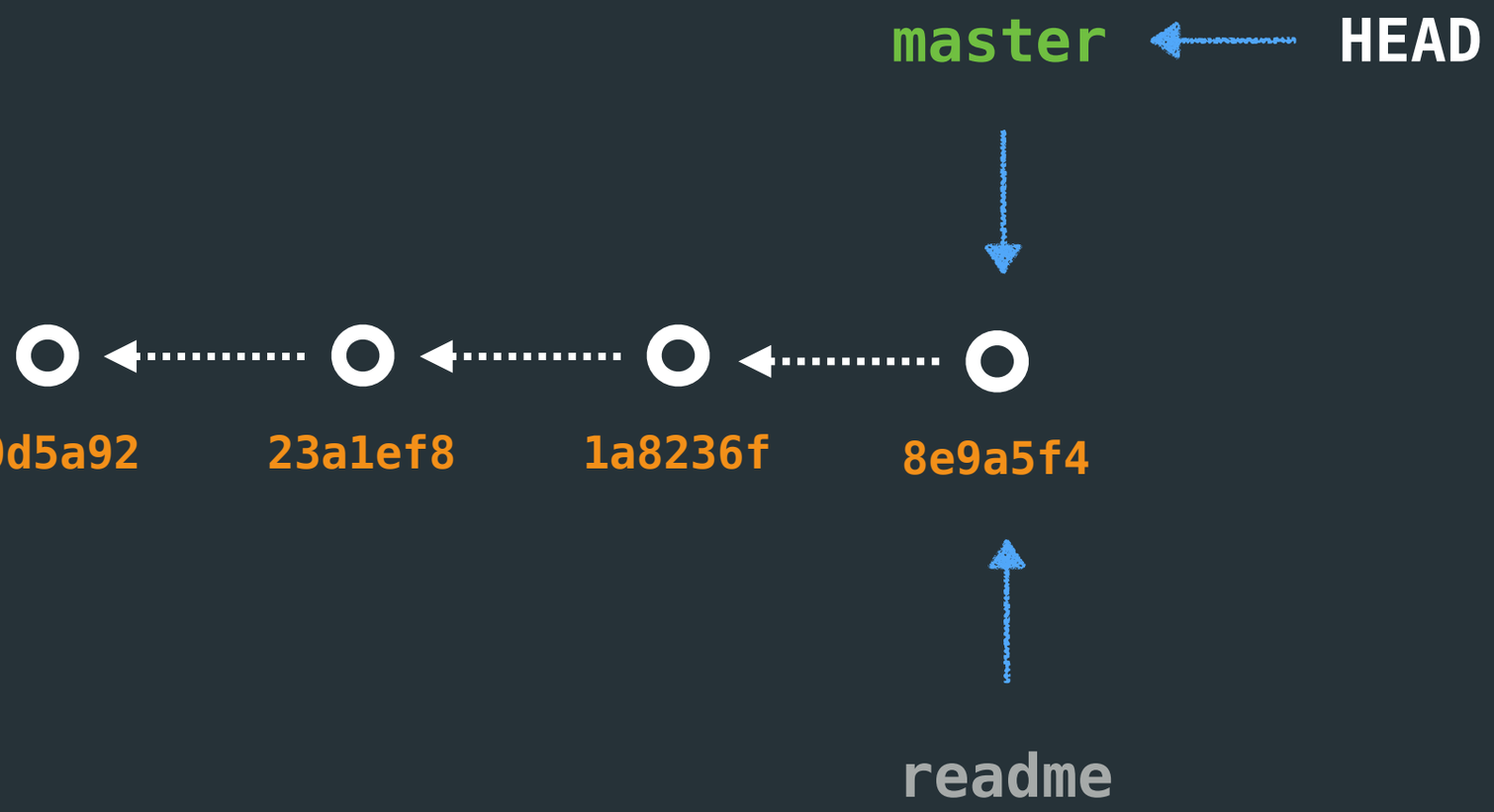
fast-forward merge workflow

~/git-repo $ **git checkout master**

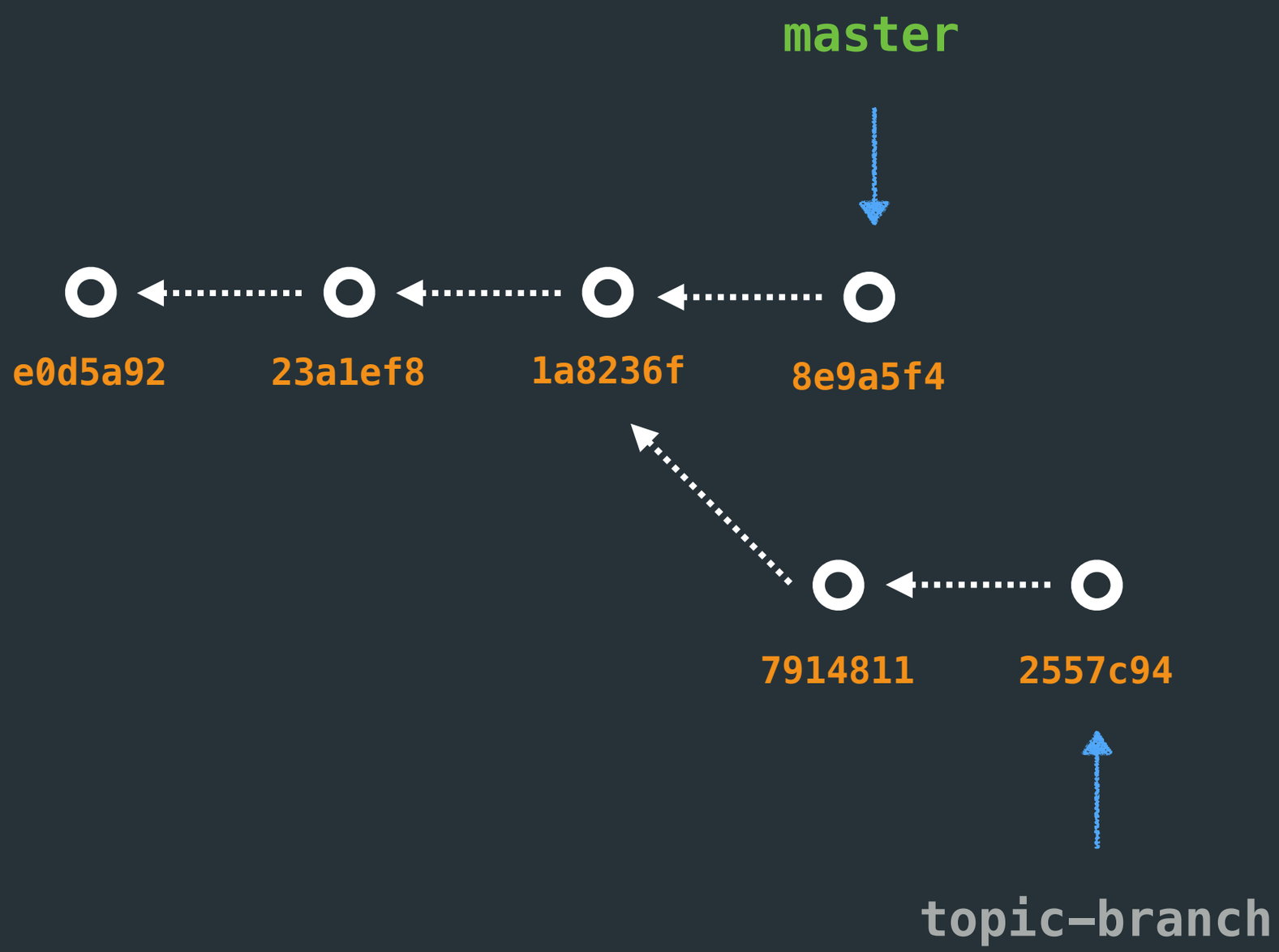


fast-forward merge workflow

~/git-repo $ **git merge readme**



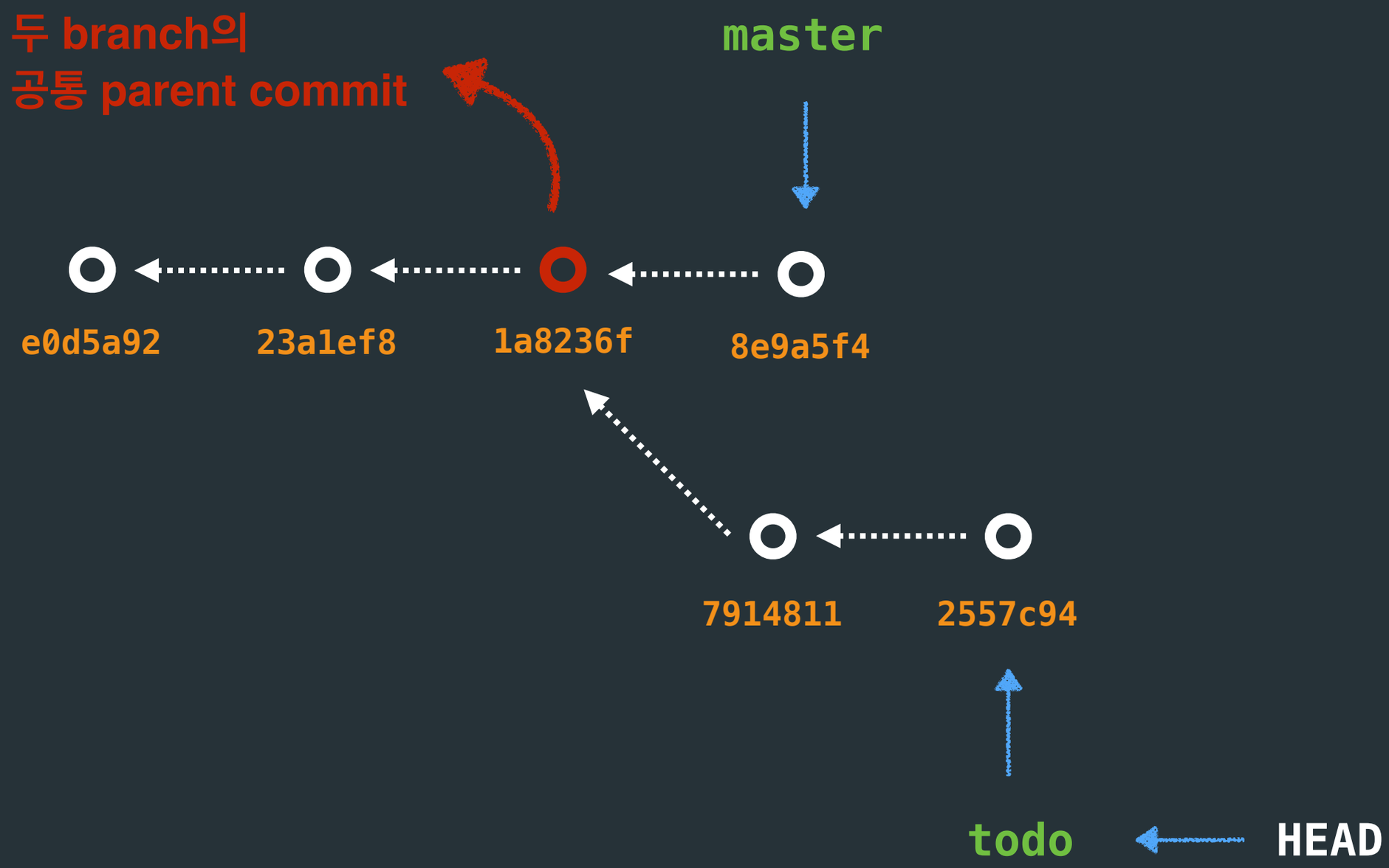
fast-forward merge 가 안된다면?



3-way merge

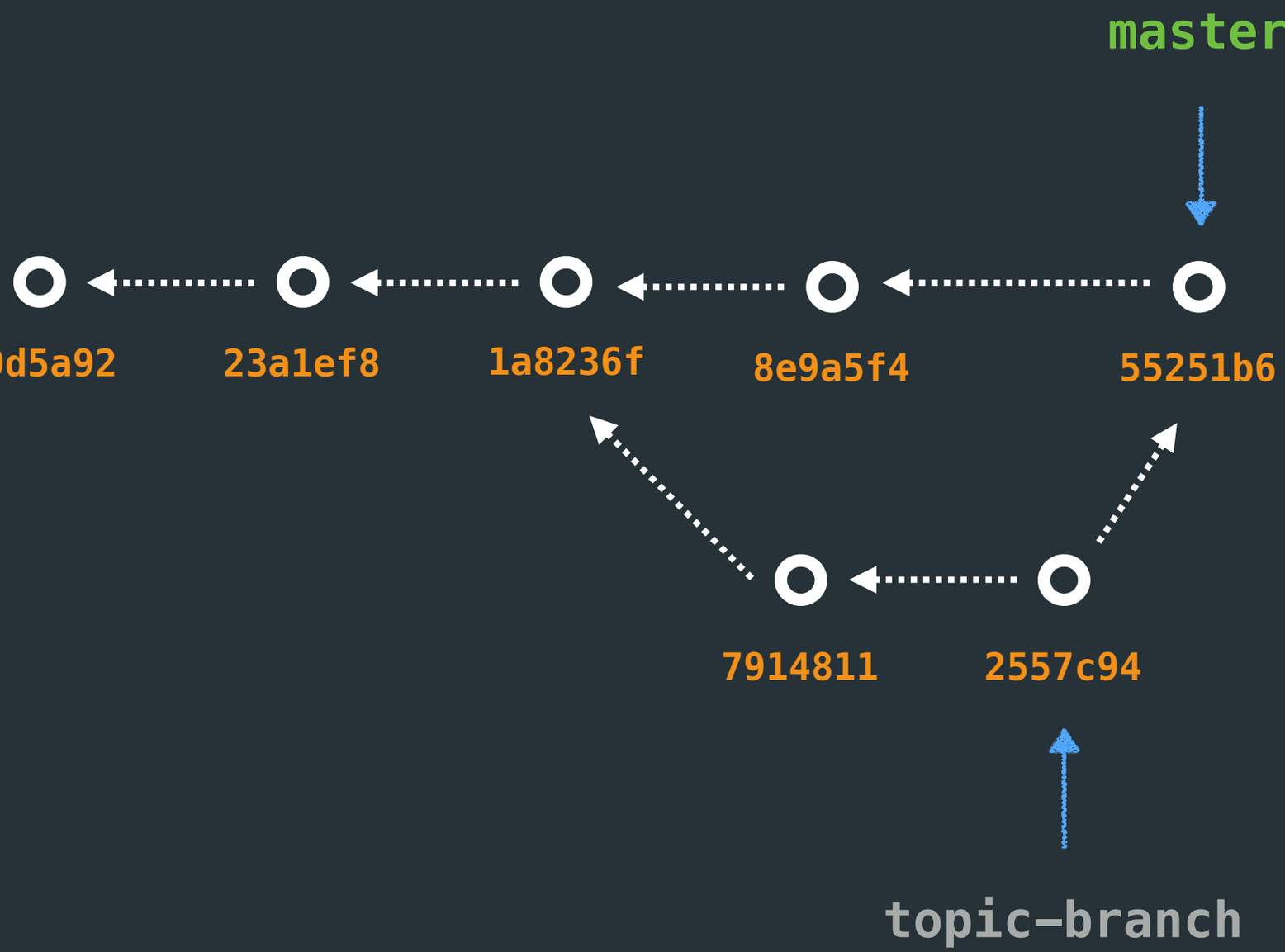
- 두 branch 간 공통의 parent commit 을 이용하여 순차적으로 merge 를 수행
- merge 의 결과를 별도의 commit object 로 저장
- 코드가 반영된 branch 가 바라보는 commit 을 새로 만들어진 commit 으로 변경
- 이때 만들어진 commit 을 merge commit 이라고 함.

공통 parent commit



3way merge

```
~/git-repo $ git merge topic-branch
```



git 3way-merge 실습 시나리오

- 컨퍼런스 티저 사이트를 만든다.
  - 티저사이트는 sample1 을 참고한다
- 이미 만들어 두었던 teaser branch 에서 작업 후 master branch 로 merge
- merge 된 이후 teaser branch 는 삭제

git 3way-merge 실습

```
~/git-repo $  git checkout teaser

~/git-repo $  git add .

~/git-repo $  git commit

~/git-repo $  git checkout master

~/git-repo $  git merge teaser

~/git-repo $  git branch -D teaser
```

bug fix와 new feature 시나리오

- 티저 사이트 기반으로 스케쥴 및 스피커 정보를 업데이트 하는 작업을 진행
- 스케쥴 부분의 글씨가 잘 안보이는 문제가 발견되어 수정 해야 함
- 두 작업 모두 현재 master branch 를 기준으로 branch 를 만들어서 진행 되어 짐

```
93 <section class="section-schedule" id="schedule">
94     <div class="cover-blur-dark invers"></div>
95     <div class="container">
96         <h2 class="sub-title">
97             <i class="glyphicon glyphicon-time vertical-middle"></i>
98             <span class="vertical-middle">Schedule</span>
99         </h2>
100         <p style="color:white">업데이트 예정...</p>
101     </div>
102 </section>
```

bug fix - font color 변경

```
~/git-repo $  git branch speakers //스피커 정보 업데이트를 위한 branch

~/git-repo $  git branch font-color //font color 를 변경하기 위한 branch

~/git-repo $  git checkout font-color

~/git-repo $  vi index.html

~/git-repo $  git add index.html

~/git-repo $  git commit

~/git-repo $  git checkout master

~/git-repo $  git merge font-color
```

new feature - 스피커 정보 업데이트

```
~/git-repo $  git checkout speakers //스피커 정보 업데이트를 위한 branch

sample 2 의 index.html 을 복사

~/git-repo $  git add index.html

~/git-repo $  git commit

~/git-repo $  git checkout master

~/git-repo $  git merge speakers
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

3-way 의 가장 큰 골치거리 conflict

3-way merge conflict

- merge 를 하려는 두 커밋 간 공통부분에 대한 수정이 있을 경우 충돌 발생
- 충돌이 발생하면 merge commit을 만들어 내지 못하고 동작이 멈추게 됨
- 사용자가 직접 충돌을 해결 후 merge commit 을 만들어 해결

3-way merge conflict 해결

충돌한 파일 확인

```
$ git status
```

충돌 내용 확인

```
$ git diff
```

해결 방법

충돌이 일어난 파일을 찾아 최종 결과로 저장되어야 할 내용으로 저장후 커밋

git conflict 해결

```
diff --cc index.html
index da880d2,a2f7444..0000000
--- a/index.html
+++ b/index.html
@@@ -97,7 -179,125 +179,129 @@@
            <i class="glyphicon glyphicon-time vertical-middle"></i>
            <span class="vertical-middle">Schedule</span>
          </h2>
++<<<<<<< HEAD
 +        <p style="color:white">업데이트 예정...</p>
++=======
+         <h3>Session 1.</h3>
+         <ul class="schedule">
+           <li>
```

~/git-repo $ **git status**

On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

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

        both added: index.html

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

~/git-repo $ **git diff**

~/git-repo $ **vi index.html**

~/git-repo $ **git add index.html**

~/git-repo $ **git commit**

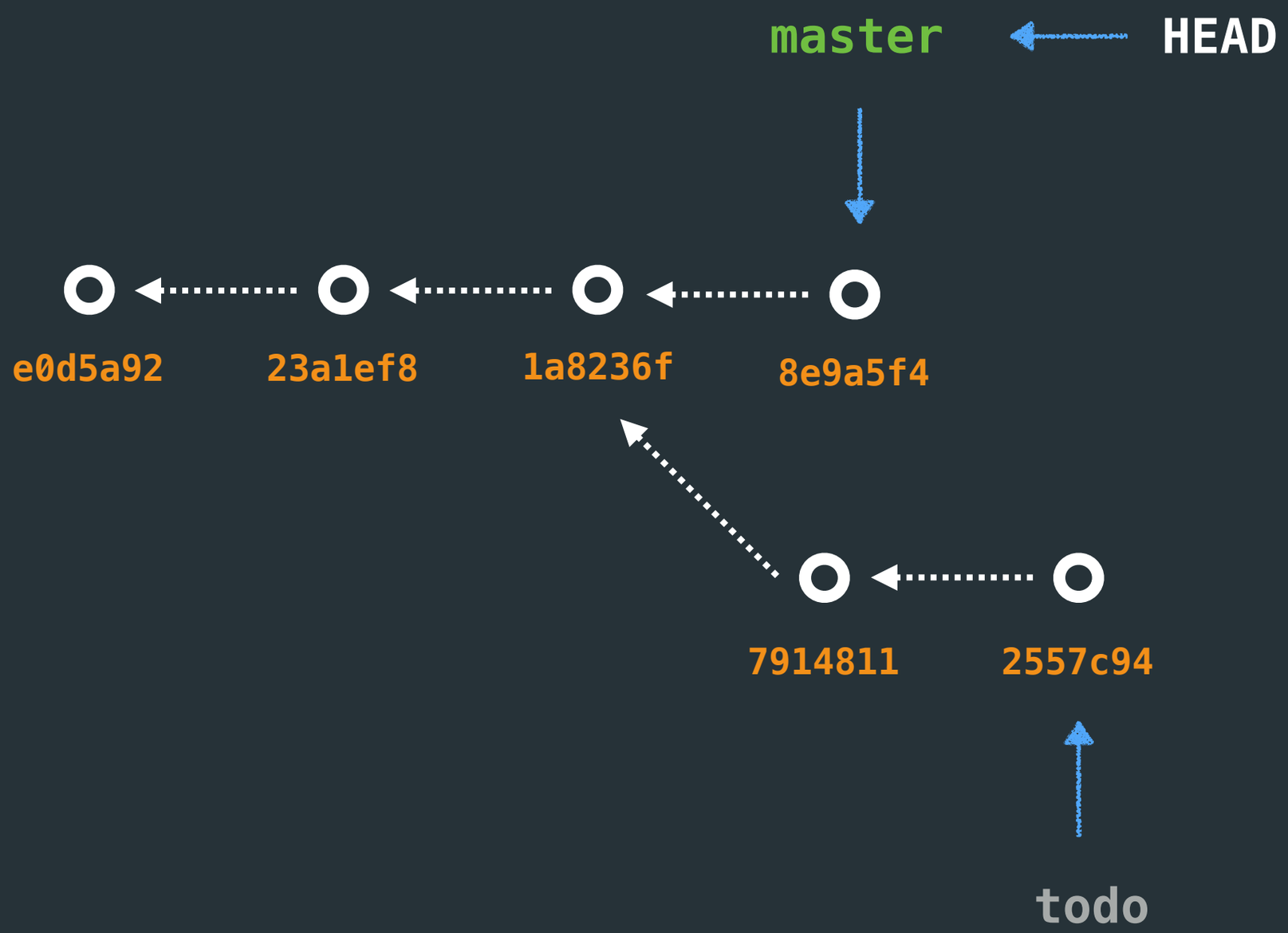
git rebase

git rebase

- 지정한 base 를 기준으로 현재 branch 의 base commit 을 변경
- 내부적으로는 base commit 을 기준으로 현재 branch 에서 추가된 변경 을 하나씩 적용
- merge 와 log history 자체가 변경 되기 때문에 호불호가 많이 갈리는 기능

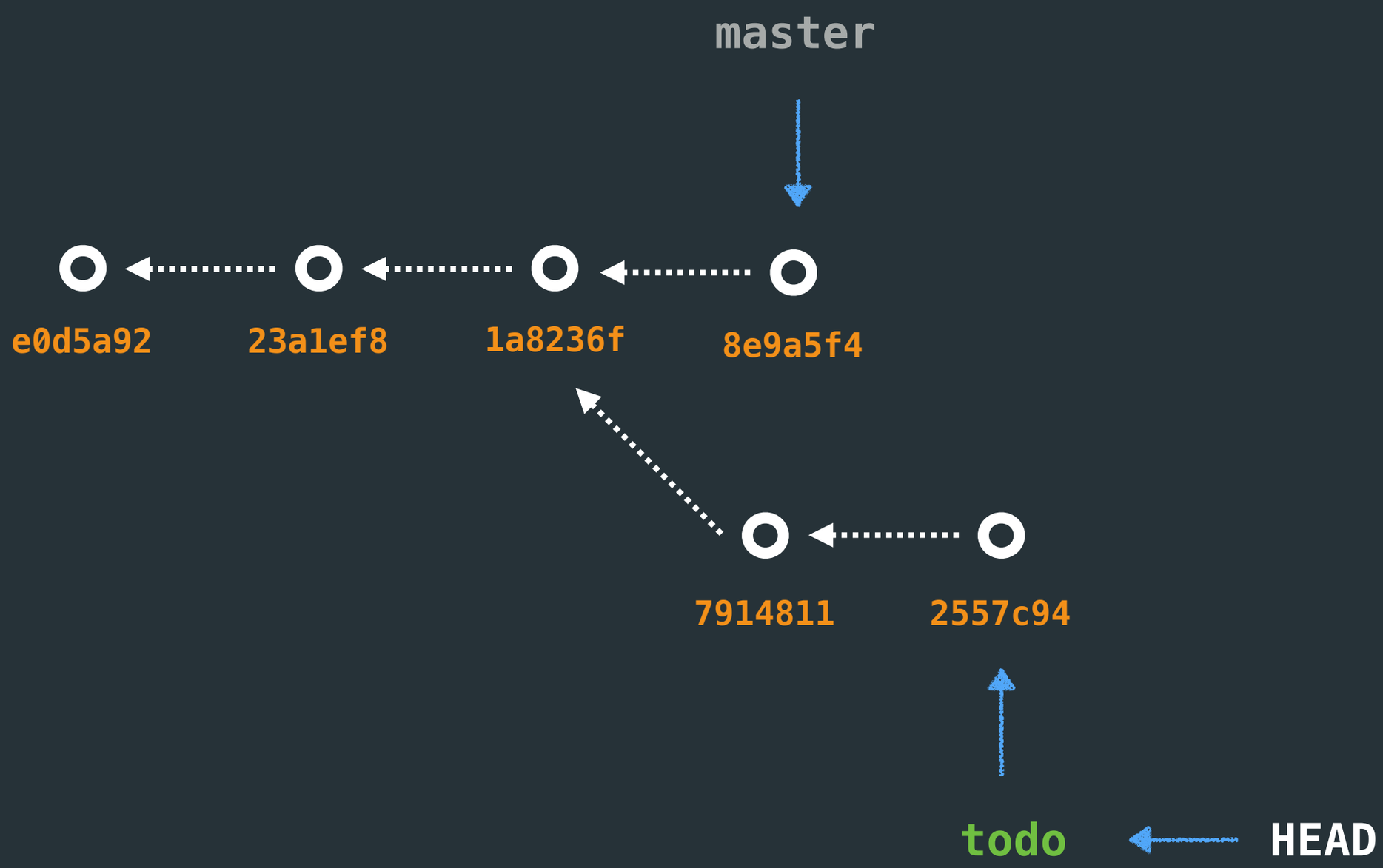
git rebase workflow

3-way merge 가 일어나는 상황의 branch



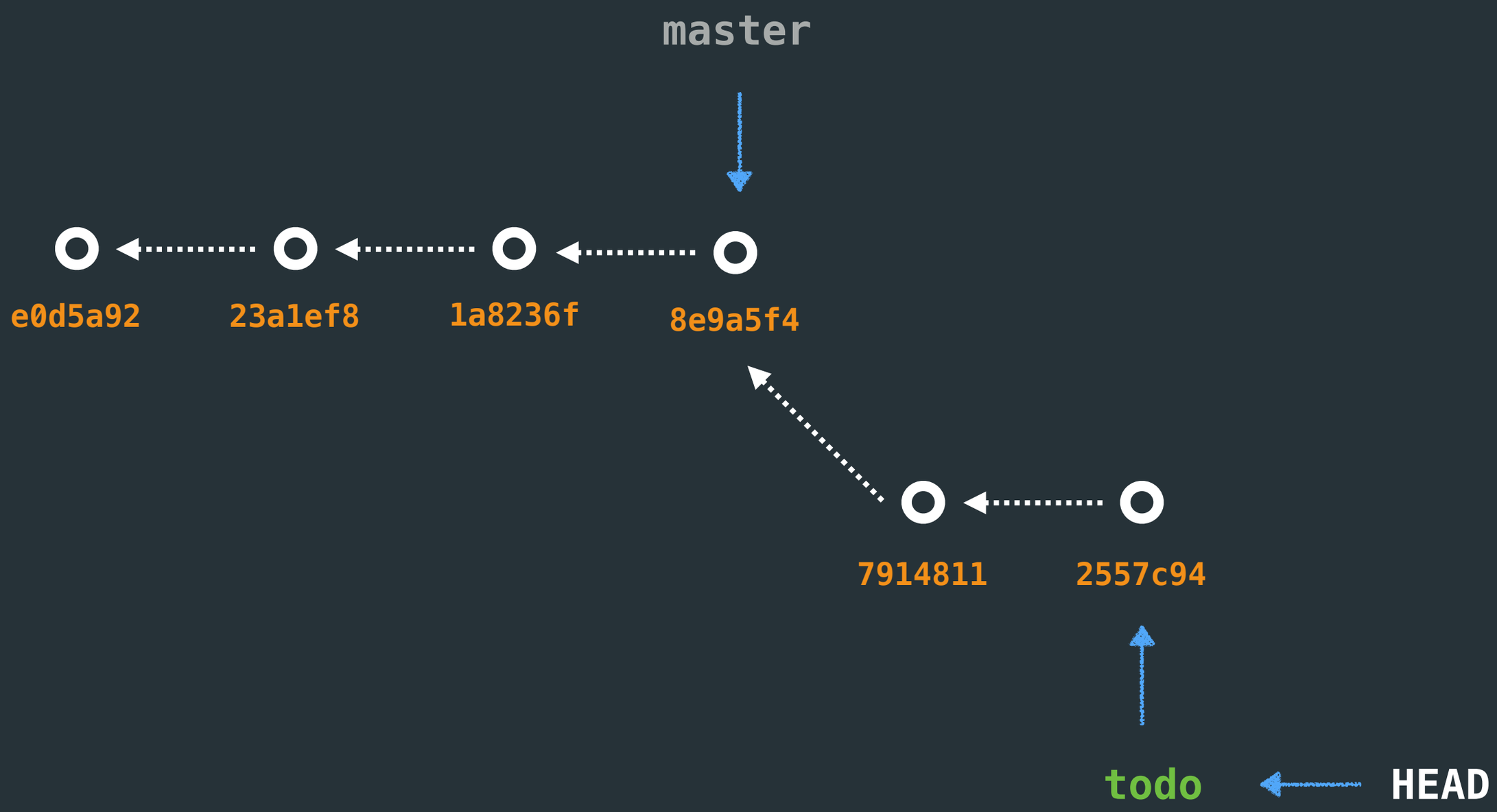
git rebase workflow

~/git-repo $ **git checkout todo**



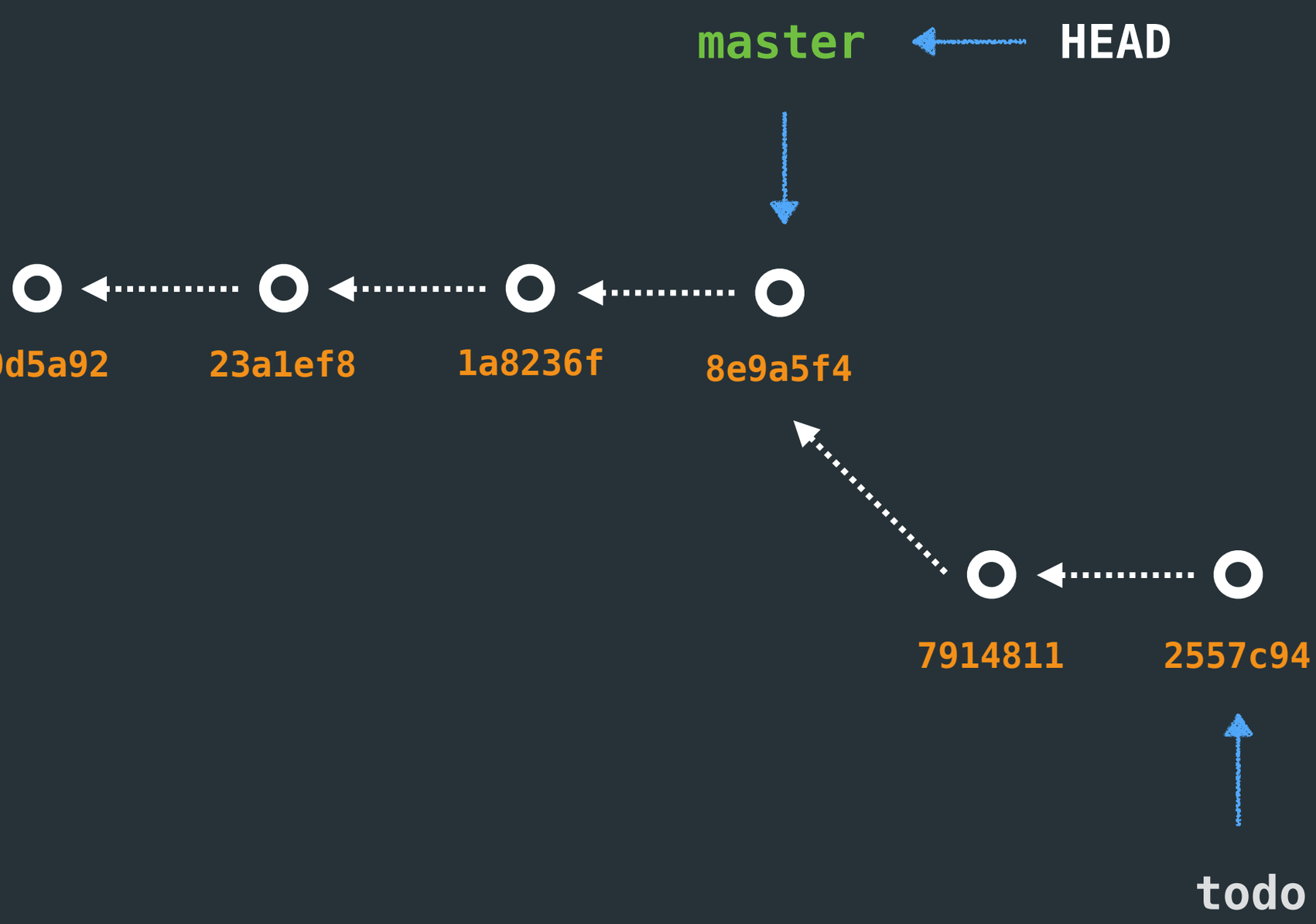
git rebase workflow

```
~/git-repo $ git rebase master
```



git rebase workflow

~/git-repo $ **git checkout master**



git rebase workflow

~/git-repo $ **git merge todo**

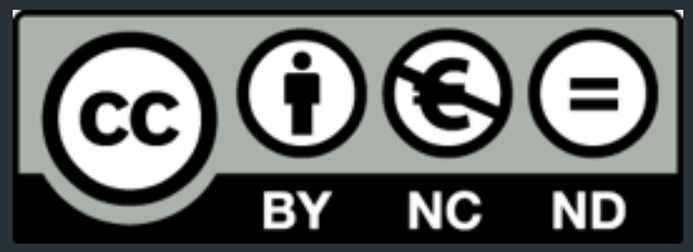
git rebase 주의 사항

- base commit 을 기준으로 커밋을 새롭게 만들기 때문에 상당한 conflict 상황이 발생 가능
- 작업 history 가 유지 되지 않게 되므로 작업 이력이 중요한 경우에는 적절하지 않음
- 개념을 잘 이해하지 못하고 사용하게 되면 더욱 큰 고통에 시달리게 됨

git rebase 충돌 해결

- 충돌이 발생한 파일을 수정 한다.
- merge 와 다르게 git add 까지만 수행
- git rebase --continue 명령을 통해 rebase 가 완료 될때까지 진행
- 충돌이 발생하면 이 작업을 반복해서 수행

Q & A



twitter : @insanehong
email : insane.hong@navercorp.com