

# 1. Branch 개념

## 1) Branch란?

- 협업 시, 각자 맡은 것을 작업
- 브랜치를 나누지 않으면, 여러 사람이 commit 할 경우에 충돌이 날 수 있다. 또한 서로 작업한 commit 내역을 내 로컬 repo에도 반영해줘야 하는 번거로움 발생

## 2) Branch 나누기

- main 브랜치가 있다면, 본인의 branch를 파서 내 작업만 진행하고, 나중에 main에 합쳐주면 된다!

# 2. Branch 생성

## 1) 브랜치 생성

- git branch : 현재 등록된 브랜치 확인
- git branch -v : 등록된 브랜치의 상세한 정보 확인

```
git branch <branch 이름>
git branch -v
```

## 2) 브랜치 리스트 확인

- 리스트 확인하고 q 눌러서 빠져나오기

```
git branch
```

## 3) 브랜치로 이동

- checkout : 작업할 브랜치로 바꾸는 것, 체크아웃된 브랜치에만 커밋이 반영된다.

```
git checkout <branch 이름>
```

# 3. 코드 수정

## 1) add

```
git add .
git add <파일 이름>
```

## 2) commit

```
git commit -m "메세지 입력"
```

## 3) push

```
git push origin <branch 이름>
```


# 4. Pull Request 생성 및 작성

👉 Pull Request를 날리는 이유?

협업 시, 코드에 잘못된 부분, 수정할 부분들을 미리 알기 위해서이다.

## 1) Pull Request

- push 후 해당 repo에 들어가면 사진과 같이 pull request 버튼이 활성화 되어 있음

 mark had recent pushes less than a minute ago

Compare & pull request

## 2) Pull Request 작성

작성 시 필요한 것

- 브랜치에서 어떤 기능을 수정했는지 확인
- 주석을 제대로 사용했는지 확인
- 수정전 코드가 그대로 있지 않은지 확인



test

Write

Preview

H

B

*I*

≡

<>

🔗

≡

≡

☑

@

🗨

↩


Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

M+

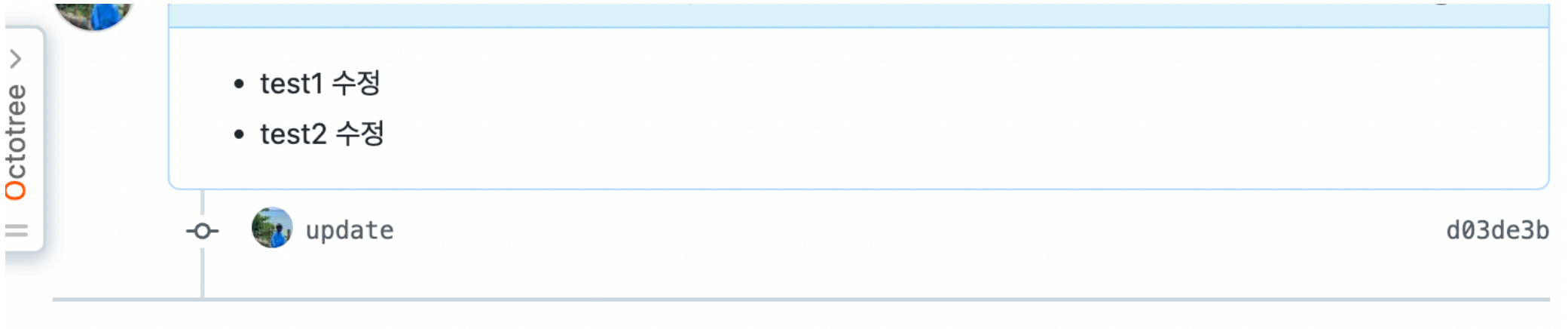
Create pull request

## 3) 완료 및 코드리뷰 받기

 코드 리뷰

해당 코드에 리뷰가 달리면, 코드를 다시 수정 후 푸쉬하면 된다.

좋은 코드라면 -> merge



## 5. Merge

### 1) Merge(병합)란?

- 작업 내용 합치기
- 서로 다른 브랜치에서 작업을 했거나, 작업 내용을 합쳐야 하는 경우 merge를 해주면 됨

### 2) 브랜치 상태 확인

- Checkout한 브랜치를 기준으로 `--merged` , `--no-merged` 옵션을 사용하여 merge가 된 브랜치인지 아닌지 필터링할 수 있다.

```
git branch --merged
```

```
git branch --no-merged
```

### 3) Merge 하기

- '현재' 브랜치에서 [브랜치 명]의 변경사항을 병합

예를 들어 `master`브랜치와 `test` 브랜치가 있다고 했을 경우,  
**`**git merge test**`**를 하게되면  
`test`브랜치에만 있던 코드가 `master`브랜치에 병합된다.

```
// master에 체크아웃  
git checkout master  
// test브랜치의 코드를 master에 합침  
git merge test
```

```
git merge [브랜치명]
```

## Merge시 충돌 해결하기

### 1) merge conflict

- merge할 때 발생하는 conflict(충돌)
- 에러를 안내느냐가 아닌 해결할 수 있느냐가 중요함

### 2) 충돌이 생길 경우

- git은 충돌 내용을 하단과 같이 코드 상에 보여준다.

```
&&<<<<<<<< HEAD  
{현재 브랜치의 다른 파일 내용}  
=====  
{충돌나는 브랜치명 또는 commit에서의 다른 파일 내용}  
>>>>>>> 충돌나는 브랜치명 또는 commit 아이디
```

- 예시

```
<<<<<<<< HEAD  
master content -> 현재 브랜치[master]에서 수정된 내용
```

```
=====
test content -> 머지할 브랜치[test]에서 수정한 내용
>>>>>>> 충돌나는 브랜치명 또는 commit 아이디
```

- 서로 다른 부분을 수정해줘야 함

### 3) 충돌 만나도록 명령어

```
# 대상 브랜치로 이동
git checkout [대상브랜치]

# 대상 브랜치의 로컬 최신화
git pull origin [대상브랜치]

# 다시 내 작업 브랜치로 이동
git checkout {작업 브랜치}

# 머지 요청
git merge [대상브랜치]

# 수정 후, add, commit, push 진행
```

## 6. Merge 후 브랜치 삭제

### 1) merge가 완료되었다면 사용하지 않는 브랜치를 삭제해준다.

```
git branch -d <branch 이름>
```