

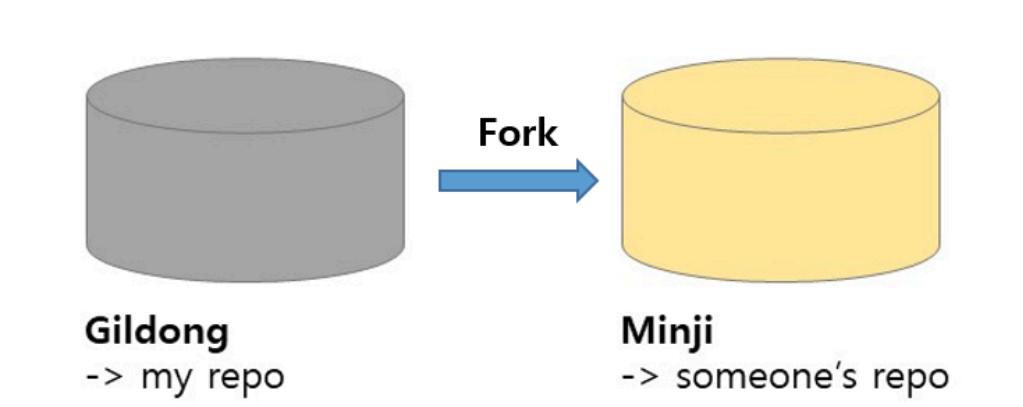
# 코드 기여 원리 (Fork 와 Pull Request (PR))

다음과 같이 Gildong와 Minji 라는 깃헙 리포지토리가 있다고 하자.



이때 Minji가 Gildong 개발자의 프로젝트가 마음에 들어, 같이 프로젝트에 참여해 기여자(Contribute)로서 공헌을 하고 싶다고 한다.  
하지만 내가 다른 사람의 저장소에 있는 코드를 수정하거나, 다른 사람의 저장소의 코드를 내가 수정하려면 관리자가 직접 나를 기여자(Contribute)로 등록이 되어있어야 한다.  
하지만 모든 사람을 다 Contribute로 등록할 수는 없는 현실이다.

이때 사용하는 것이 **Fork** 이다.

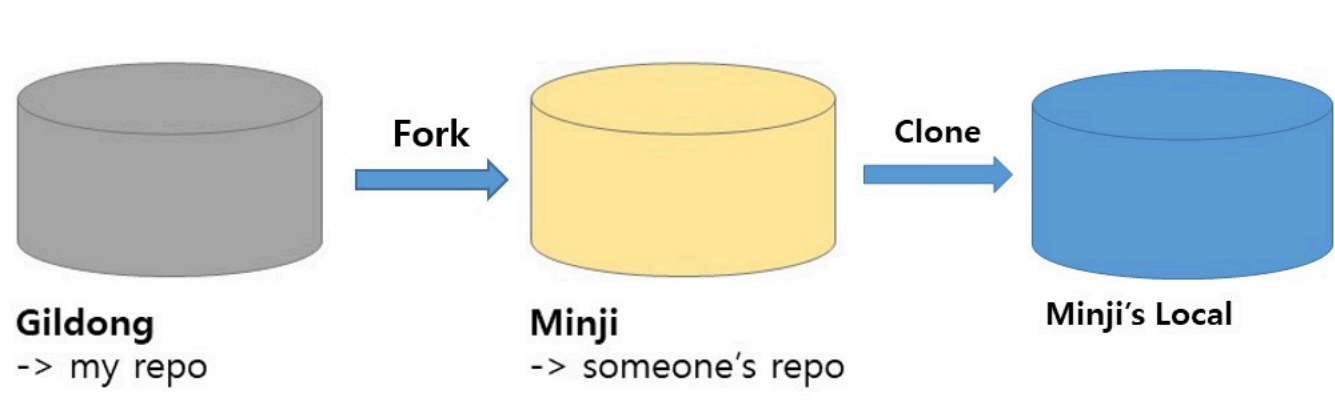


포크로 쿡 찢러 가져오듯 **다른 사람의 저장소에 있는 레포지토리를 내 원격 저장소, 깃허브로 가져오는 것이다.**  
즉, Minji라는 유저가 Gildong의 레포지토리 중 하나를 Fork하였다면, Minji의 github에 해당 레포지토리가 그대로 가져와 지제 된다.  
그리고나서 Fork해온 원격 레포지토리를 내 로컬에 Clone 한 후 코드를 수정하면 된다.

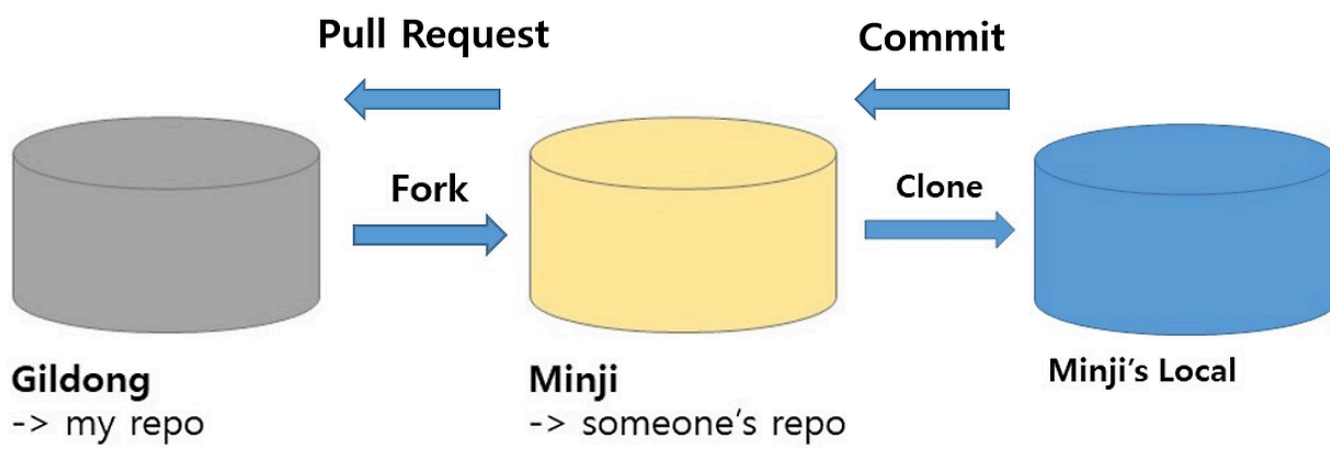
Tip

Fork : 레포지토리를 원격저장소에 복사

Clone : 레포지토리를 로컬저장소에 복사



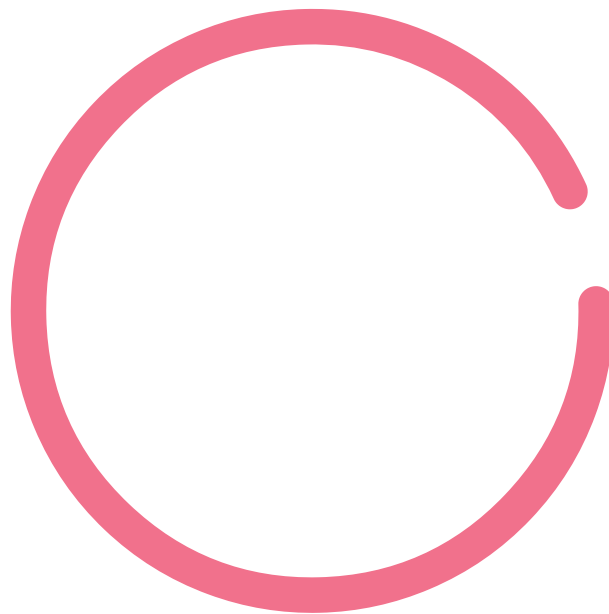
이제 Minji 개발자가 이것 저것 코드를 수정하거나 업그레이드 한 후, 아까 fork했던 Minji의 원격 저장소에 커밋한 뒤,  
Gildong의 레포지토리에 반영이 되면 좋겠다 생각되면 **Pull Request**를 보내는 것이다.  
그러면 Gildong이 코드 리뷰를 하고 문제가 없으면 자신의 메인 브랜치에 merge를 하는 식으로 프로젝트를 기여하는 방식이다.



## Pull Request 란?

따라서 Pull Request(PR)을 정리하자면, 내가 수정한 코드가 있으니 내 branch를 가져가 검토 후 병합해주라고 요청 해주는 것이라고 보면 된다. PR을 통해 코드 충돌을 최소화할 수 있고 push 권한이 없는 오픈 소스 프로젝트에 기여할 때 많이 사용한다.

## pull request 사용 단계



위에서 Gildong과 Minji를 예를 들어 설명했지만 다시한번 총정리하자면 다음과 같다.

내 원격 리포지토리에 Fork

clone 설정

remote 설정

branch 생성

수정 작업 후 add, commit, push (만일 나 뿐만 아니라 다른 기여자(팀원)이 소스를 수정했다면 fetch로 코드를 가져와 충돌을 해결)

Pull Request 생성

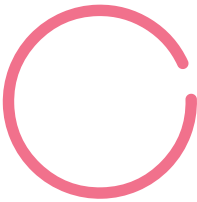
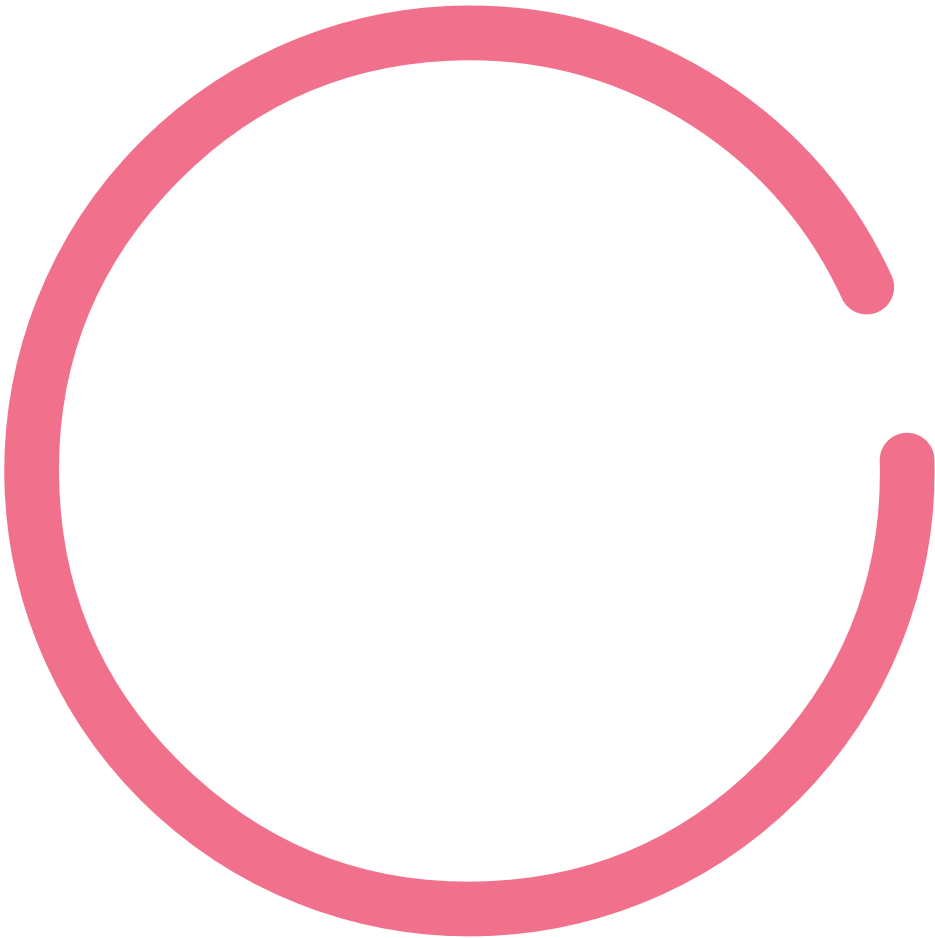
Merge Pull Request

Merge 이후 동기화 및 branch 삭제


# 깃헙 Pull Request 실습

## 1. 원본 저장소를 Fork 하기



기여하기 원하는 리포지토리의 우측 상단에 포크 버튼 눌러 Fork한다.













Fork가 완료되면 **내 원격 저장소**에 해당 레포지토리가 복제 된다.  
그리고 리포지토리 이름 밑에 **Fork From 원래저장소**라고 출처가 출력되는 것을 확인할 수 있다.

 inpa-dev / markdown-tistory Public

forked from jojoldu/markdown-tistory

 Pin  Watch

<> 코드  풀 리퀘스트  액션  프로젝트  위키  보안  인사이트  설정



 master ▾  9 브랜치  0 태그



Go to file








Add file ▾

Code ▾

This branch is up to date with jojoldu/markdown-tistory:master.

 Contribute ▾  Fetch upstream ▾

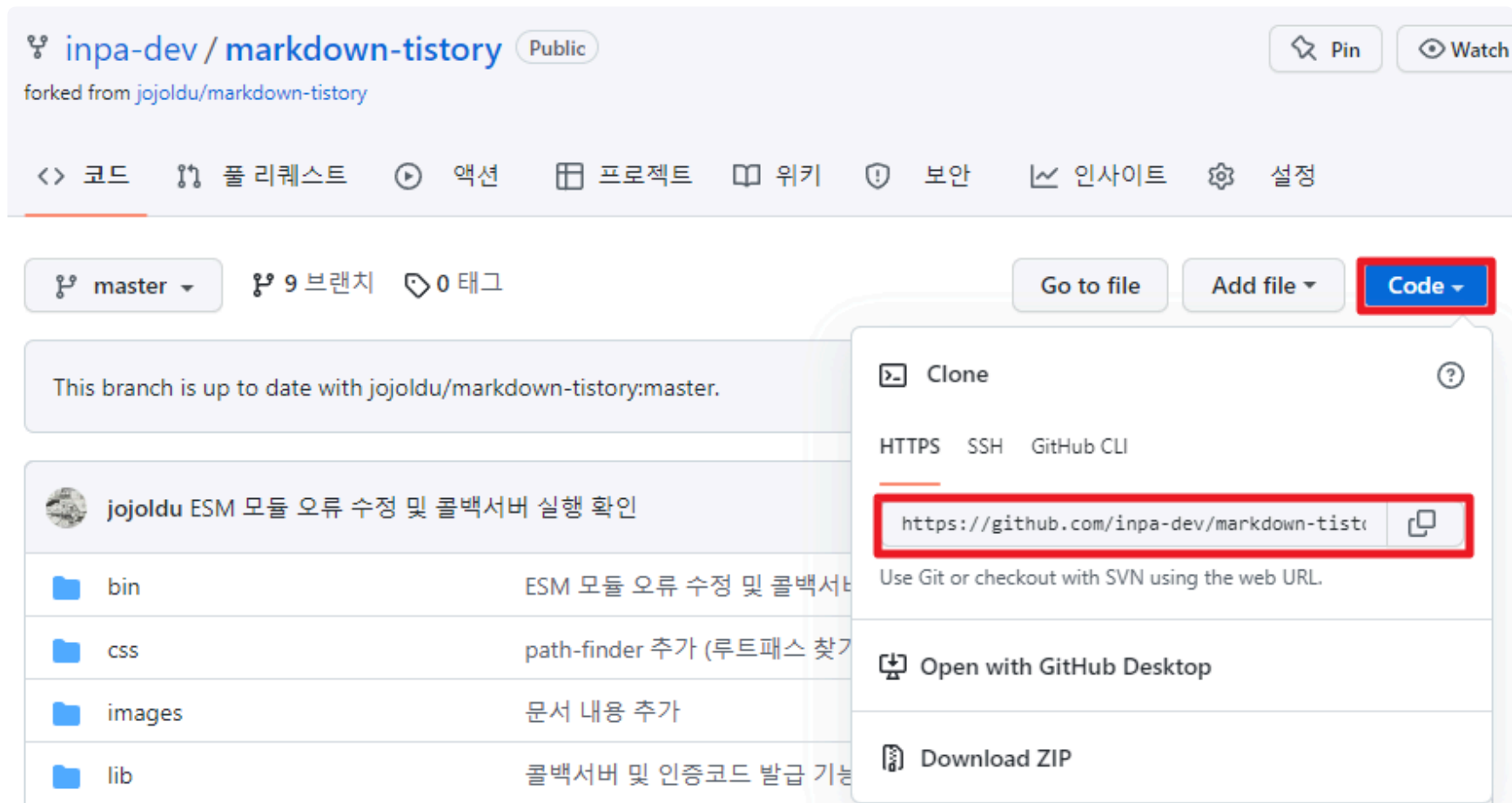
 jojoldu ESM 모듈 오류 수정 및 콜백서버 실행 확인 908bc33 on 17 May 2020  119 commits

 bin	ESM 모듈 오류 수정 및 콜백서버 실행 확인	2년 전
 css	path-finder 추가 (루트패스 찾기용)	5년 전
 images	문서 내용 추가	3년 전
 lib	콜백서버 및 인증코드 발급 기능 추가	2년 전
 src	ESM 모듈 오류 수정 및 콜백서버 실행 확인	2년 전
 test	LocalImage Path만 골라내는 Collection 클래스추가	2년 전
 .babelrc	ESM 모듈 오류 수정 및 콜백서버 실행 확인	2년 전

## 2. Fork한 저장소를 로컬로 Clone 하기

이제 작업하기 위해 포크 한 원격 저장소 소스를 그대로 내 로컬 컴퓨터에 복사해 추가한다.

Code 버튼을 눌러 표시되는 URL을 복사하고 명령어를 실행한다.



BASH

```
$ cd C:\Users\MS\Desktop\공부 # clond할 로컬 폴더로 이동
$ git clone <레포지터리 URL>
```

```
C:\Users\MS\Desktop\공부
$ git clone https://github.com/inpa-dev/markdown-tistory.git
Cloning into 'markdown-tistory'...
remote: Enumerating objects: 1013, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 1013 (delta 14), reused 0 (delta 0), pack-reused 988
Receiving objects: 100% (1013/1013), 897.17 KiB | 27.19 MiB/s, done.
Resolving deltas: 100% (522/522), done.
```

## 3. 원본 저장소 Remote 설정

레파지토리를 clone하면 **origin** 이라는 이름으로 remote url이 내 원격 저장소로 기본으로 설정되어 있다.

하지만 내 원격 저장소 뿐만 아니라, 포크 했던 원본 저장소도 remote로 등록 해주는 게 좋다. 나중에 원본 저장소 내용과 동기화 하기 위해 사용할 것이기 때문이다.

BASH

```
# 원본 저장소 remote 등록
$ git remote add <별명> <원본 저장소 url>
```

```
# 원격 저장소 확인
$ git remote -v
```

```
C:\Users\MS\Desktop\공부\markdown-tistory (master -> origin) (markdown-tistory@0.2.0)
$ git remote add father https://github.com/jojoldu/markdown-tistory.git father이라는 별칭으로 원본 저장소를 remote 등록

C:\Users\MS\Desktop\공부\markdown-tistory (master -> origin) (markdown-tistory@0.2.0)
$ git remote -v
father https://github.com/jojoldu/markdown-tistory.git (fetch)
father https://github.com/jojoldu/markdown-tistory.git (push)
origin https://github.com/inpa-dev/markdown-tistory.git (fetch)
origin https://github.com/inpa-dev/markdown-tistory.git (push)
```

원본 저장소

내 저장소

## 4. Branch 생성

현업에서 팀과 함께 일할때 브랜치 분기는 필수이다.

예를 들어, 버그를 고치는 작업을 한다고 하면 bug-fix라는 branch를 만들어 코드를 고친 후 push하면 master 브랜치의 코드는 그대로 유지된 채로 bug-fix 브랜치만 최신 코드가 되게 된다.

그래서 만약에 bug-fix 브랜치에서 수정된 코드가 사이드 이펙트를 발생한다고 해도 master에 코드가 남아있기 때문에 다시 이전 코드로 되돌리기가 쉽다.

따라서 현업에서는 branch를 만들어 작업한 뒤 다시 작업이 완료되면 삭제하는 식으로 진행하는 경우가 많다.

다음과 같이 개발용 branch를 만들어서 진행한다.

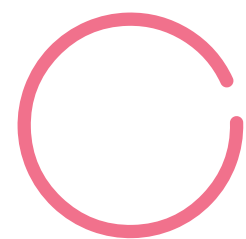
BASH

```
# 브랜치 생성
$ git branch <branch 이름>
# 브랜치 이동
$ git checkout <branch 이름>
# 브랜치 생성후 이동 (단축 명령어)
$ git checkout -b <branch 이름>
# 브랜치 리스트
$ git branch
```



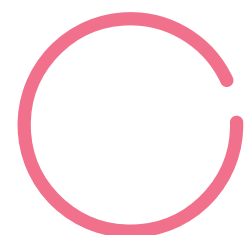
## 5. 코드 작업 후 Add / Commit / Push

코드를 수정하거나 기능 추가 작업이 완료되면, 자신의 깃헙 리포지토리(origin)에 add, commit, push 하여 반영한다.

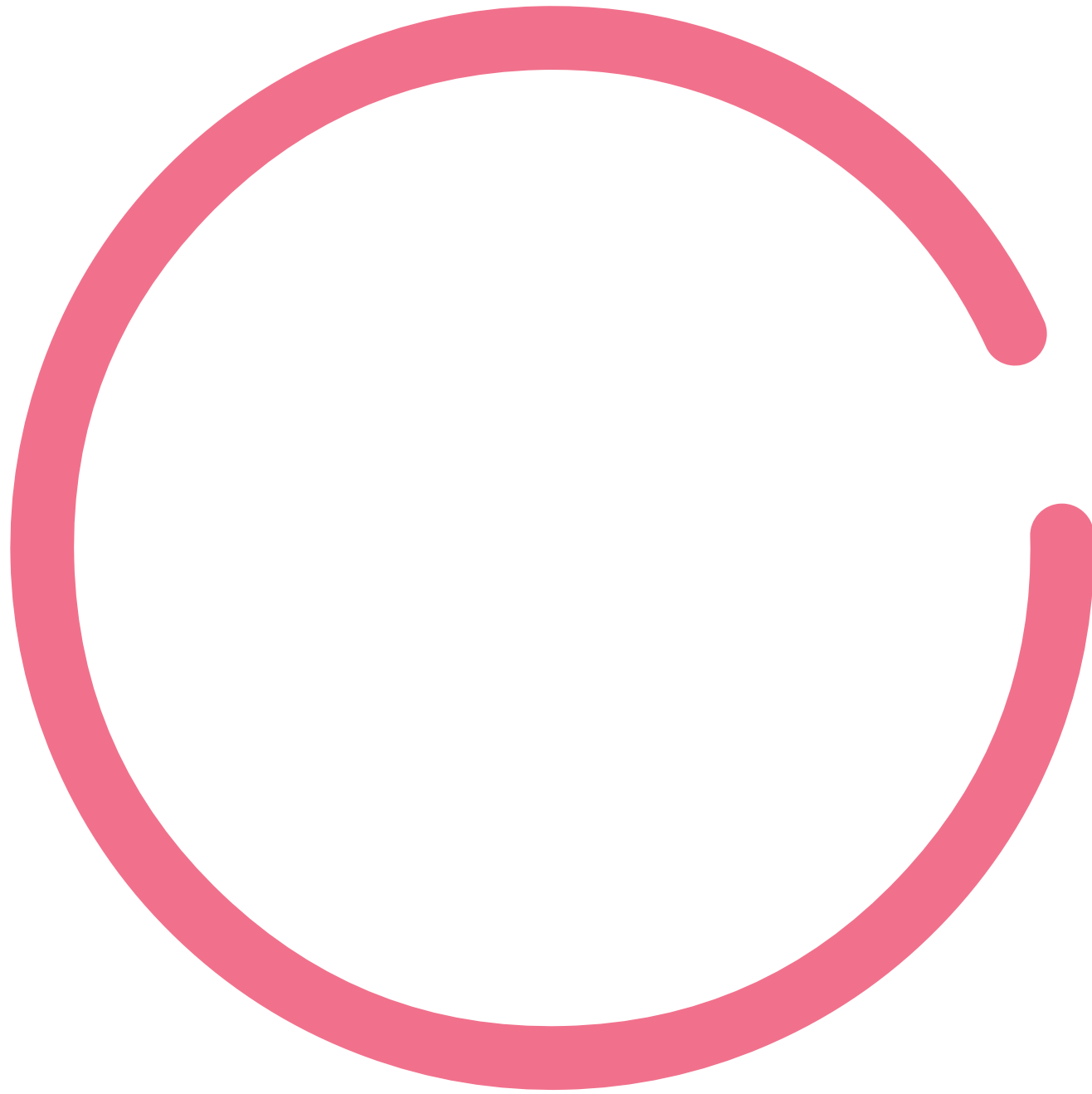


BASH

```
$ git add .
$ git commit -m "test입니다. 무시하세요"
$ git push origin develop
```



Push 완료후, 깃헙 리포지토리에 접속해보면 내가 변경한 사항(text.txt)이 반영되어 있는것을 볼 수 있다.



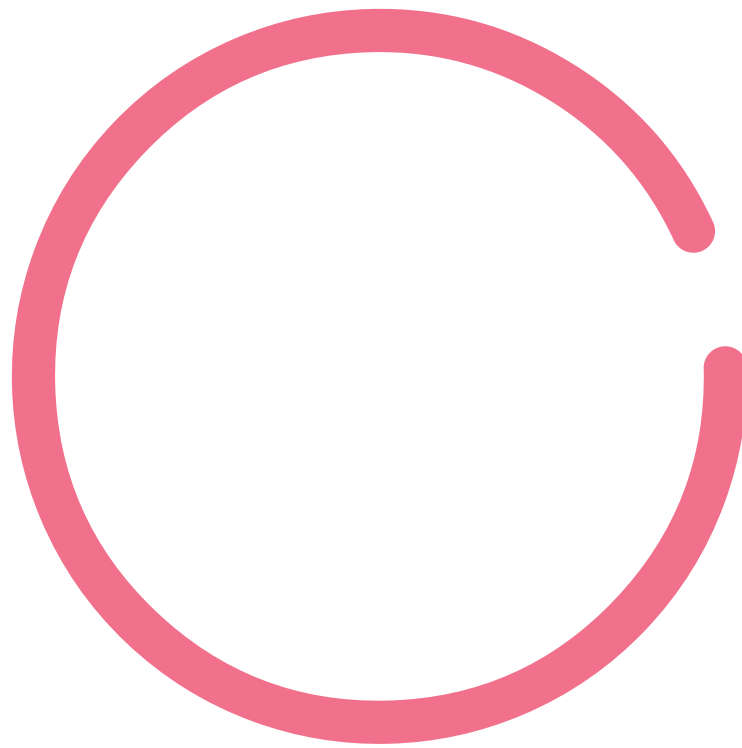
## 6. Pull Request 생성


깃헙 저장소 상단을 보면 **Compare & pull request** 버튼이 활성화되어 있는 것을 볼 수 있다.

상단에 버튼을 클릭하고, PR 메시지를 작성하고 **Create pull request** 버튼을 누르면 풀 리퀘스트를 생성하게 된다.

### Tip

PR 메시지를 쓸 때에는 브랜치에서 어떤 기능을 어떻게 수정했는지 자세하게 쓰는 것이 좋다.












test입니다. 무시하세요


Write

Preview

H B I  <>     @  


테스트

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

☒ Allow edits by maintainers 

Create pull request

Helpful resources  
[GitHub Community Guidelines](#)


 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

1 commit



1 file changed

1 contributor

Commits on Jun 14, 2022

test입니다. 무시하세요  
 inpa-dev committed 9 minutes ago


내가 기여한 작업

 17f906e 

Showing 1 changed file with 1 addition and 0 deletions.

Split

Unified

1 test.txt 

... @@ -0,0 +1 @@  
1 + "Hello World !!"

## 7. Merge Pull Request

Pull Request를 받은 원본 저장소 관리자는 코드 변경내용을 확인하고 Merge 여부를 결정하게 된다.

원작자가 승인을 하면 Merge Confirm으로 원본 저장소에 변경된 사항이 반영이 되고, pull request의 상태는 closed로 변경된다. 만일 맘에 들지 않는다 하면 Reject 된다.

🔗 10 열람 ✓ 7 닫힘
<div> <div>🔗</div> <div> <div>Bump lodash from 4.17.15 to 4.17.19 ✓</div> <div>dependencies</div> </div> </div> <div>#12 by dependabot bot 닫음 on 17 Mar</div>
<div> <div>🔗</div> <div> <div>등록/수정시 옵션값 추가.</div> </div> </div> <div>#11 by ini8262 닫음 on 18 Jan 2020</div>
<div> <div>🔗</div> <div> <div>등록/수정시 옵션값 추가.</div> </div> </div> <div>#10 by ini8262 닫음 on 16 Jan 2020</div>
<div> <div>🔗</div> <div> <div>Update README.md : 티스토리 링크 업데이트</div> </div> </div> <div>#6 by jhlead was merged on 14 Nov 2018</div>
<div> <div>🔗</div> <div> <div>README 설명 추가</div> </div> </div> <div>#3 by glqdlit was merged on 3 May 2018</div>
<div> <div>🔗</div> <div> <div>markdow tistroy 오타 수정</div> </div> </div> <div>#2 by baeharam was merged on 28 Apr 2018</div>
<div> <div>🔗</div> <div> <div>버그 수정</div> </div> </div> <div>#1 by eschocolat was merged on 15 Jan 2017</div>

## 8. Merge 이후 동기화 및 Branch 삭제

원본 저장소에 Merge가 완료되면 로컬 코드와 원본 저장소의 코드를 동기화한다.  
그리고 작업하던 로컬의 branch는 더이상 사용을 안하기 때문에 삭제한다.

BASH

```
# 코드 동기화
$ git pull father # 원본 remote 와 동기화

# branch 삭제
$ git branch -d develop # 용무가 끝난 브랜치 삭제
```

이런식으로, 나중에 추가로 작업할 일이 있으면 git pull 명령을 통해 원본 저장소와 동기화를 진행하고 위의 과정을 반복하면 된다.