

---

# 오픈소스 소프트웨어 개발 - 17

## (Online) Git & Github - forking

광운대학교 이윤상  
2017년 2학기

# 이번 시간에 할 것

---

- Forking Workflow
- [실습] Github를 이용한 Forking Workflow

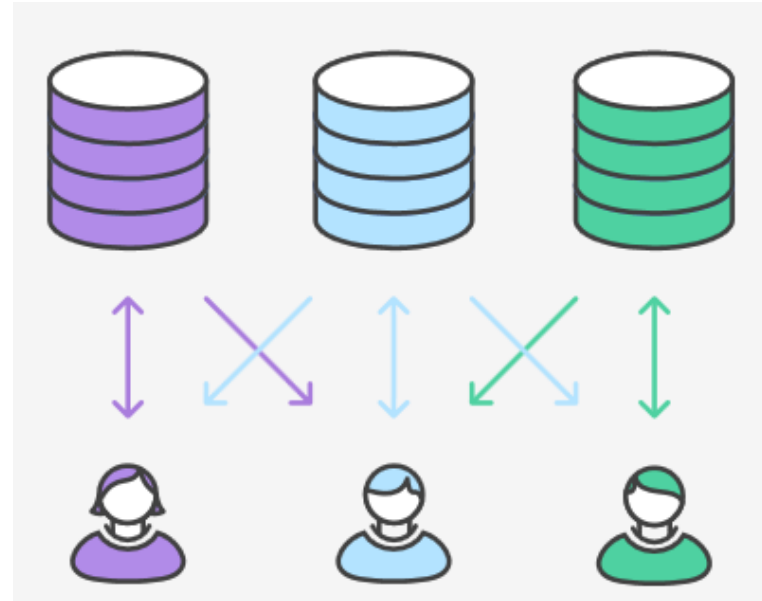
# Forking Workflow

- 하나의 공식 저장소(원격)

- read 권한: 누구나
- **write** 권한: 해당 프로젝트를 이끌어가는 소수의 개발자 (maintainer 혹은 committer)

- 각 개발자마다 각각의 개별 저장소(원격)

- read 권한: 누구나
- **write** 권한: 해당 개발자



- 개발자들은 공식 저장소를 복제하여(fork) 자신의 개별 저장소를 만들어 작업을 시작. **작업 내용은 각자의 개별 저장소에 push.**
- 작업을 완료하면 공식저장소에서 자신의 저장소에 추가된 변경사항을 **pull**하기를 maintainer(혹은 committer)에게 요청 (**Pull Request, PR**)

# Forking Workflow

---

- 공식저장소의 write 권한을 모두에게 주지 않고도 불특정 다수의 개발자로부터 (선별하여) commit을 받는 것이 가능
- Pull request를 통해 해당 작업 내용에 대한 토론도 가능
- 팀원이 정해져 있지 않은, 대규모의 유기적인 팀에 적절한 유연한 개발 방식
- 예) 오픈 소스 프로젝트
- Github에서 호스팅되는 대부분의 프로젝트들이 사용하는 방식

# Forking Workflow Example

<https://www.atlassian.com/git/tutorials/comparing-workflows>

The project maintainer initializes the official repository



official repository

# Forking Workflow Example

<https://www.atlassian.com/git/tutorials/comparing-workflows>

Developers fork the official repository



developer's  
forked  
repository



official  
repository



developer's  
forked  
repository

```
git clone https://user@bitbucket.org/user/repo.git
```

자신의 forked repository를 git clone.  
forked repository는 origin이라는 이름으로 추가됨.

```
git remote add upstream https://bitbucket.org/maintainer/r
```

(optional) official repository를 upstream이라는 이름으로 추가

# Forking Workflow Example

<https://www.atlassian.com/git/tutorials/comparing-workflows>

Developers work on their features



```
git checkout -b some-feature  
# Edit some code  
git commit -a -m "Add first draft of some feature"
```

보통 task 마다 새로운 branch를 만들어서 작업

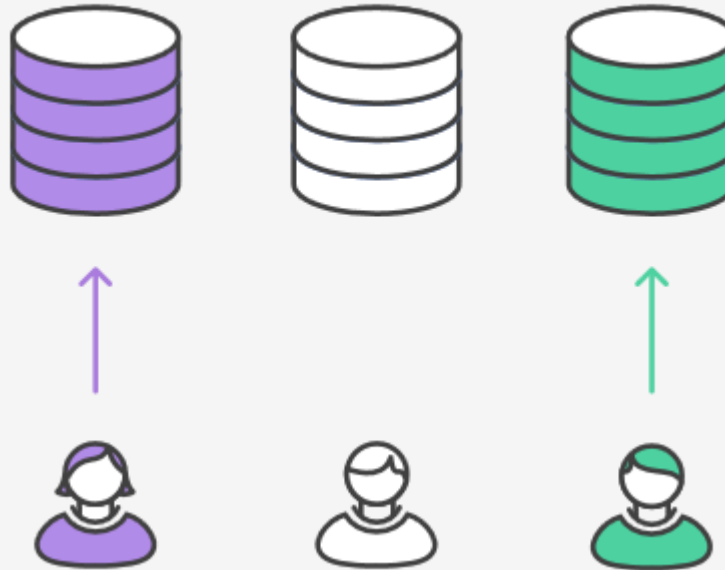
```
git pull upstream master
```

작업 중 official repository에 추가된 변경사항은  
git pull로 가져와서 merge

# Forking Workflow Example

<https://www.atlassian.com/git/tutorials/comparing-workflows>

Developers publish their features



```
git push origin feature-branch
```

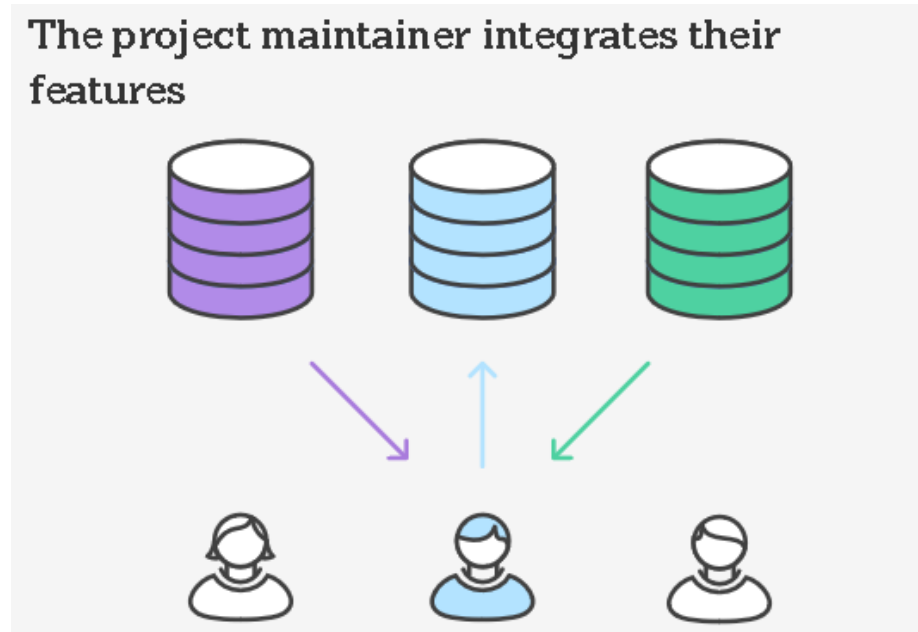
작업이 완료되면 자신의 forked repository로 push.

그리고 official repository의 maintainer에게 자신의 저장소의 변경사항을 pull할 것을 요청 (pull request)



# Forking Workflow Example

<https://www.atlassian.com/git/tutorials/comparing-workflows>

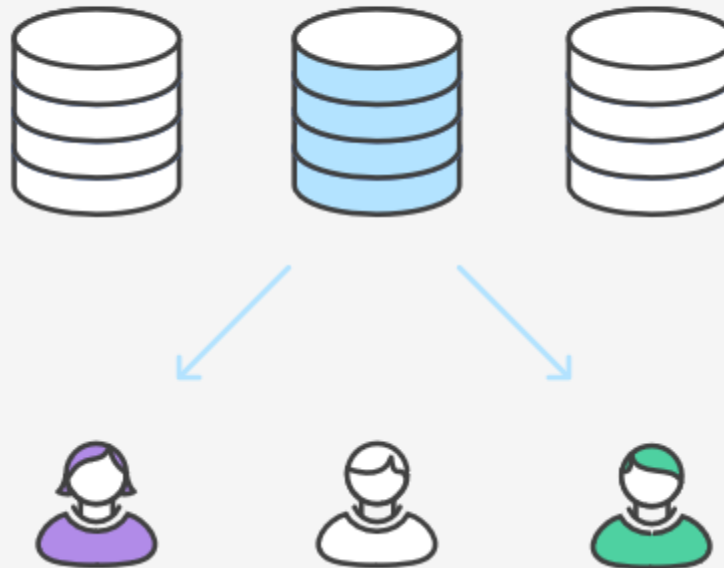


maintainer가 검토 후 문제없다고 생각하면  
공식저장소에 merge

# Forking Workflow Example

<https://www.atlassian.com/git/tutorials/comparing-workflows>

Developers synchronize with the official repository



```
git pull upstream master
```

# [실습] Github에서의 Forking Workflow

- id1이 'ForkingTest'라는 저장소를 만든다.
  - Github에 id1으로 로그인해서 저장소를 새로 만든 후,

## (Shell)

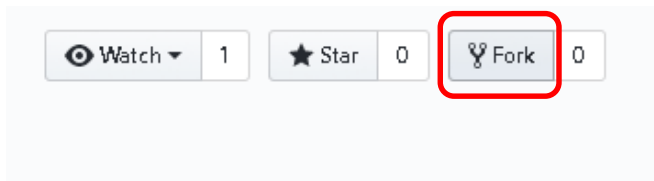
```
mkdir -p ~/github-2/id1/ForkingTest # 디렉터리 이름, 위치는 변경가능  
cd ~/github-2/id1/ForkingTest
```

# Github에서 ...or create a new repository on the command line  
에 나와있는 명령어를 copy & paste하여 파일추가, 지역저장소 생성,  
원격저장소 추가를 하자

- **id1/ForkingTest**를 ForkingTest 프로젝트의 공식저장소라고 생각하자.

# [실습] Github에서의 Forking Workflow

- id2가 id1/ForkingTest 페이지를 방문
  - id2로 로그인 한 browser에서  
[https://github.com/\(id1\)/ForkingTest](https://github.com/(id1)/ForkingTest) 방문
- id2가 오른쪽 상단의 'Fork' 버튼을 누르면 공식 저장소(id1/ForkingTest)를 복제한 자신의 개별 저장소(id2/ForkingTest)가 만들어진다.



# [실습] Github에서의 Forking Workflow

- id2가 ForkingTest에 contribution을 하기 위해 우선 자신의 forked repository를 git clone.

## (Shell)

```
mkdir -p ~/github-2/id2 #디렉터리 이름, 위치는 변경가능  
cd ~/github-2/id2
```

```
git clone https://github.com/(id2)/ForkingTest
```

```
cd ForkingTest
```

```
git config user.name (id2) # 이번 실습에서만, history상의 구분을 위해 이 저장소에서만 user name을 임시로 변경하자.
```

# [실습] Github에서의 Forking Workflow

---

- 두 종류의 다른 browser에서 각각 Github에 id1과 id2로 로그인 해보자.
- id1에서는 자신의 repository 목록에 ForkingTest가 보일 것임.
- id1에서는 자신의 repository 목록에 ForkingTest(Forked from id1/ForkingTest)라고 보일 것임.

# [실습] Github에서의 Forking Workflow

- id2가 새로운 기여를 위해 **새로운 branch를 만들어** 작업 후 자신의 저장소로 push.
  - 반드시 새로운 branch를 만들어서 작업해야 한다.

(Shell: id2/ForkingTest)

```
git checkout -b feature-a
```

```
vi fileA.txt # Add 'hello'
```

```
git add .
```

```
git commit
```

```
git push origin feature-a
```

```
# id2로 로그인하여 push. 자신의 repository ((id2)/ForkingTest)
# 에 push하는 것이므로 바로 push 가능함.
```


# [실습] Github에서의 Forking Workflow

---

- Github에서 id2/ForkingTest의 feature-a branch와 commit이 추가된 것을 확인해보자.
  - Code - Commits - Branch: feature-a 선택
- Commit 1개로 feature-a 작업이 마무리되었기 때문에, id2가 id1에게 공식저장소에 feature-a 내용의 적용해 줄 것을 요청하자.



# [실습] Github에서의 Forking Workflow

- id2가 자신의 ForkingTest 페이지에 가서 Branch를 feature-a로 변경한 후  Compare & pull request 혹은  New pull request 버튼을 누르면 Pull Request 페이지가 열린다.
- base fork(반영될 공식저장소)의 branch를 확인 후
  - 프로젝트에 따라 master가 아닌 branch로 pull request를 보내야 할 수도 있으므로, 프로젝트의 contribution guide 확인 필요
- 수정할 내용을 설명하는 제목과 설명을 작성하여 'Create Pull Request' 버튼을 누른다.

# [실습] Github에서의 Forking Workflow

---

- id1/ForkingTest에 Pull requests 숫자가 1로 변경되고 방금 id2가 보낸 PR이 추가된 것을 확인
- id1이 fileA.txt의 내용을 살펴본 후, 의견을 아래 두 가지 탭에서 올릴 수 있음.
  - Conversation 탭: 일반적인 comment 혹은 질문 등
  - File changed 탭: 라인 별 comment 혹은 코드 리뷰
- 여기에서는 코드 리뷰를 통해 수정 요청을 하기로 하자.

# [실습] Github에서의 Forking Workflow

---

- id1이 해당 PR의 File changed tab에서 fileA.txt의 변경 내용의 첫 번째 줄에 나타나는 '+' 표시를 누르고 comment를 입력한 후 'Start a review'를 누르자.
- 다른 파일에 대해 다른 언급할 내용이 없으면 'Finish your review'를 누른 후, 'Request changes'를 선택해 'Submit review'를 누르자.
  - comment 예: 'fileA를 고치고 문서도 업데이트 하라'

# [실습] Github에서의 Forking Workflow

- 'Request changes'로 리뷰를 받으면 해당 PR에 아래와 같이 표시가 됨.

The screenshot displays a GitHub Pull Request (PR) interface. At the top, a notification bar indicates that 'yssl requested changes 11 minutes ago', accompanied by a red 'X' icon and a 'View changes' button. Below this, a comment box contains the text 'Plz change the fileA'. Underneath the comment, a file named 'fileA.txt' is listed with a 'Show outdated' button. A yellow box highlights the 'Changes requested' section, which features a red circle icon with a white 'X' and the text 'Changes requested' and '1 review requesting changes Learn more.'. Below this section, a table shows the review details: a red 'X' icon, the reviewer's name 'yssl', and the action 'requested changes'. To the right of the review are buttons for 'Approve changes' and 'Dismiss review'. Below the review table, a green checkmark icon indicates that 'This branch has no conflicts with the base branch' and that 'Merging can be performed automatically.'. At the bottom, there is a green 'Merge pull request' button and a link to 'open this in GitHub Desktop or view command line instructions'.

View changes

Plz change the fileA

fileA.txt Show outdated

**Changes requested**  
1 review requesting changes [Learn more.](#)

Hide all reviewers

× yssl requested changes Approve changes Dismiss review

✓ This branch has no conflicts with the base branch  
Merging can be performed automatically.

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# [실습] Github에서의 Forking Workflow

- id2는 comment 받은 대로 파일을 수정해서 다시 자신의 저장소로 push

(Shell: id2/ForkingTest)

```
vi fileA.txt # 'Hello'로 변경
vi README.md # 'Feature A' 문구 추가
git commit -a -m "Update hello & doc"

git push origin feature-a
```

- 그럼 id1/ForkingTest의 해당 PR에 아래와 같이 자동으로 업데이트가 됨.



# [실습] Github에서의 Forking Workflow

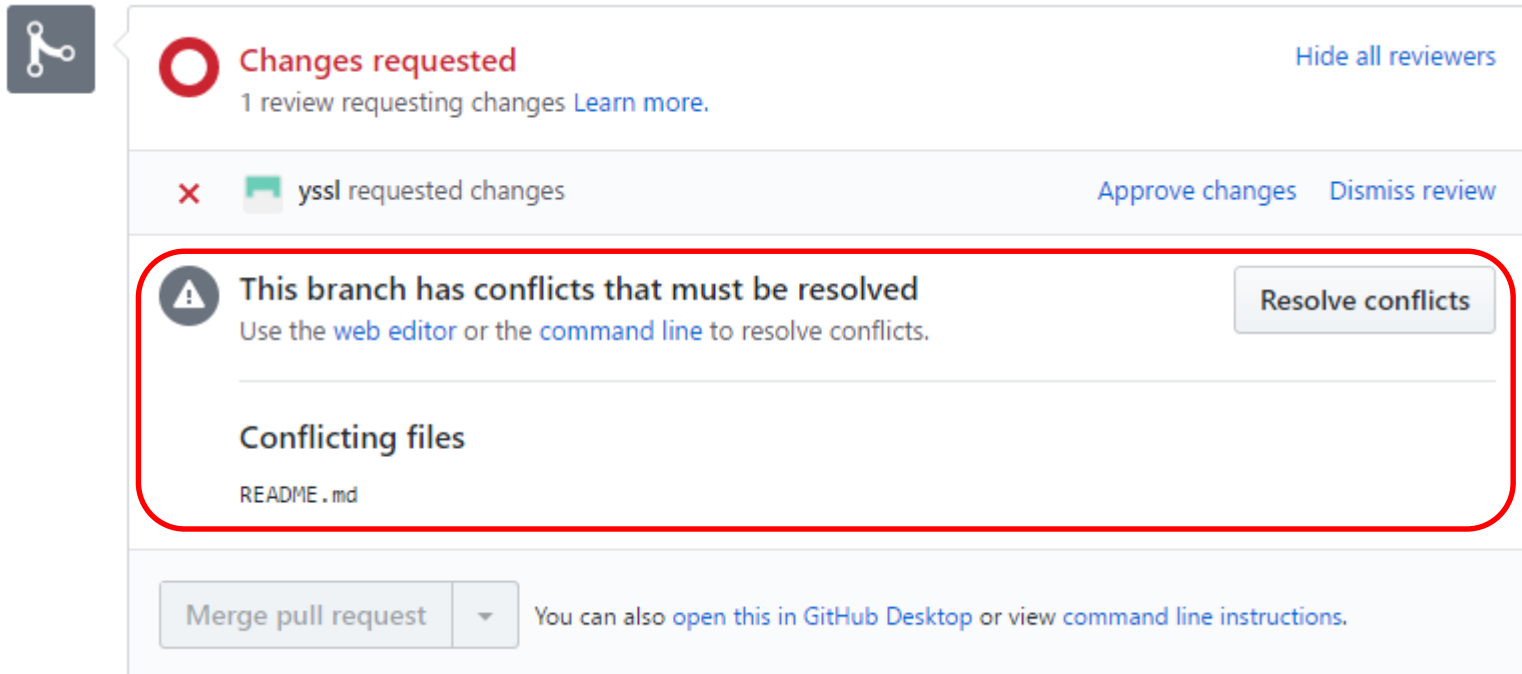
- 그런데 만일 그 직후 id1도 README.txt의 같은 부분을 수정하고 push 한다면,

(Shell: id1/ForkingTest)


```
vi README.md # 'This is blah blah~' 문구 추가  
git commit -a -m "Update doc"  
  
git push origin master
```



# [실습] Github에서의 Forking Workflow


- 이 경우 PR의 README.txt 수정부분에서 conflict이 발생하므로 아래와 같이 PR의 상태가 변경됨.




The screenshot shows a GitHub Pull Request interface. At the top, there is a 'Changes requested' section with a red circle icon and the text '1 review requesting changes'. Below this, a review by 'yssl' is shown with the status 'requested changes'. The main part of the interface is a red-bordered box with a warning icon and the text 'This branch has conflicts that must be resolved'. It also includes a 'Resolve conflicts' button. Below this, the 'Conflicting files' section lists 'README.md'. At the bottom, there is a 'Merge pull request' button and a link to 'view command line instructions'.

 **Changes requested** [Hide all reviewers](#)  
1 review requesting changes [Learn more.](#)

  yssl requested changes [Approve changes](#) [Dismiss review](#)

 **This branch has conflicts that must be resolved** [Resolve conflicts](#)  
Use the [web editor](#) or the [command line](#) to resolve conflicts.

**Conflicting files**  
README.md

[Merge pull request](#)  You can also [open this in GitHub Desktop](#) or [view command line instructions](#).

# [실습] Github에서의 Forking Workflow

---

- 따로 id1이 충돌 해결을 요청하지 않더라도, id2가 충돌을 해결하여 다시 push해야 한다.
- 'Resolve conflicts' 버튼을 눌러 web editor로 충돌을 해결할 수도 있지만, 여기에서는 local machine에서 작업하여 해결해보자.



# [실습] Github에서의 Forking Workflow

- id2가 id1/ForkingTest의 master branch를 자신의 feature-a branch에 merge 시도

(Shell: id2/ForkingTest)

```
git pull https://github.com/(id1)/ForkingTest master
(출력메시지)
...
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the
result.

# 당연히 conflict이 남
```

# [실습] Github에서의 Forking Workflow

- 충돌을 해결하고 id2/ForkingTest로 push하면, id1/ForkingTest의 해당 PR이 자동으로 merge 가능한 상태로 업데이트 됨.

(Shell: id2/ForkingTest)

```
vi README.md # resolve conflict
git add .
git commit

git push origin feature-a
```



This branch has no conflicts with the base branch

Merging can be performed automatically.

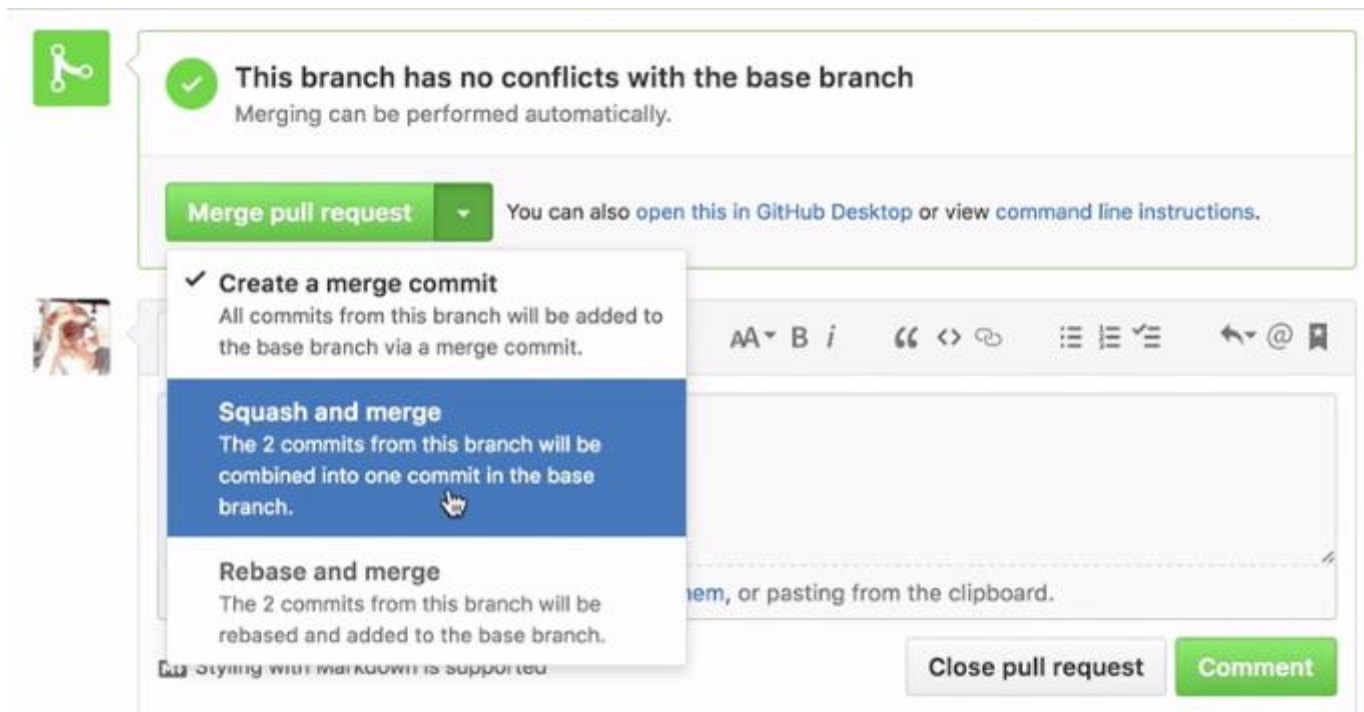
Merge pull request



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# [실습] Github에서의 Forking Workflow

- id1은 해당 PR이 자신이 요청했던 문서 업데이트가 된 것을 확인하고 'Approve changes' 누름
- 또한 해당 PR이 merge 가능 상태로 바뀐 것을 확인하고 'Squash and merge'를 선택하여 최종적으로 merge.



# [실습] Github에서의 Forking Workflow

- 해당 PR의 상태가 Open에서 Merged로 변경됨.
- (id1)/ForkingTest의 commit history를 확인해보자.
- id2는 이제 feature-a branch를 삭제해도 된다.
- (참고) Pull request의 작성 기간이 길어지면 중간중간 공식 저장소의 변경사항을 가져와야 하는데, 이를 위해 공식저장소를 별도의 이름으로 remote add 해두면 편하다 (ex. upstream)

(Shell: id2/ForkingTest)

```
git remote add upstream https://github.com/(id1)/ForkingTest.git  
  
git pull upstream master
```