

오픈소스 소프트웨어 개발 및 활용

14주차 Github 이론 및 실습

대구가톨릭대학교

김병창

- 본 강의자료는 과학기술정보통신부 및 정보통신기획평가원에서 지원하는 『소프트웨어중심대학』 사업의 결과물입니다.
- 본 강의자료는 내용은 전재할 수 없으며, 인용할 때에는 반드시 과학기술정보통신부와 정보통신기획평가원의 '소프트웨어중심대학'의 결과물이라는 출처를 밝혀야 합니다.



목차(이론)

- 1장 : Github란?
 - Github 정의 및 특징
 - Github 를 사용하기 위한 필수 요소 (repository, branch, commit, pull request, Issue, Fork)
 - Github 에서 활성화된 오픈소스 프로젝트 참여하기

목차(실습)

- Github 실습 예제
 - Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

Github 란?

1장

Github란?(Github 정의)

- 정의 : 버전 제어 및 공동 작업을 위한 코드 호스팅 플랫폼
- 특징
 - Git 저장소를 직접 설치하지 않고 Github 웹 페이지를 통해 코드 및 프로젝트 관리 가능
 - 코드 저장소 뿐만 아니라 다양한 기능 보유
 - 질문 & 답변, 이슈 사항을 기록하기 위한 Issues 페이지 제공
 - Wiki 제공
 - 월 계정권 구입을 통해 사설 저장소 제공
 - 개인 프로젝트 뿐만 아니라 공개된 프로젝트에 참여하여 프로젝트 기여를 할 수 있음

Github 란?(Github 를 사용하기 위한 필수 요소)

- Create a Repository
 - 저장소는 단일 프로젝트를 구성하기 위해 사용
 - 저장소에는 프로젝트에 필요한 폴더 및 파일, 이미지, 비디오, 스프레드 시트 및 데이터 셋이 포함
 - README나 프로젝트에 대한 정보가 담긴 파일 포함
 - 라이센스 파일과 같은 저작권 옵션 제공

Github 란?(Github 를 사용하기 위한 필수 요소)

- Create a Branch
 - Branching은 한 번에 여러 저장소에서 작업하는 방법
 - 저장소에는 최종 branch로 간주되는 master branch 존재
 - 우리는 브랜치를 사용하여 마스터하기 전에 실험하고 편집합니다.
 - 마스터 브랜치에서 브랜치를 만들면 그 시점의 마스터의 복사본 또는 스냅 샷을 만들 수 있습니다.
 - 브랜치에서 일하는 동안 다른 누군가가 마스터 브랜치를 변경한 경우에 해당 업데이트를 가져올 수 있습니다.

Github 란?(Github 를 사용하기 위한 필수 요소)

- Make and commit changes
 - GitHub에서 저장된 변경 사항을 commit 이라 칭함
 - Commit에는 관련 변경 메시지가 있으며, 이는 특정 변경에 대한 이유를 설명
 - Commit message는 변경 기록을 가지고 있어 다른 작성자가 수행한 작업과 작업 내용을 이해 가능

Github 란?(Github 를 사용하기 위한 필수 요소)

- Open a Pull Request
 - Master branch에서 변경 사항 존재 시 Pull Request 가능
 - Pull Request 는 GitHub의 공동 작업의 핵심
 - Pull Request 을 통해 수정 혹은 추가한 코드를 제안 및 요청
 - 프로젝트 관리자의 검토에 따라 master branch에 병합 여부 결정
 - Pull Request 는 나의 branch와 master branch 간의 내용 중 다른 부분을 표시
 - 코드 내 변경/추가/제거 내역은 **녹색**과 **빨간색**으로 표시
- Merge your Pull Request
 - Pull Request 의 병합 여부가 승인 될 때 변경 사항을 master branch에 적용

Github 란?(Github 를 사용하기 위한 필수 요소)

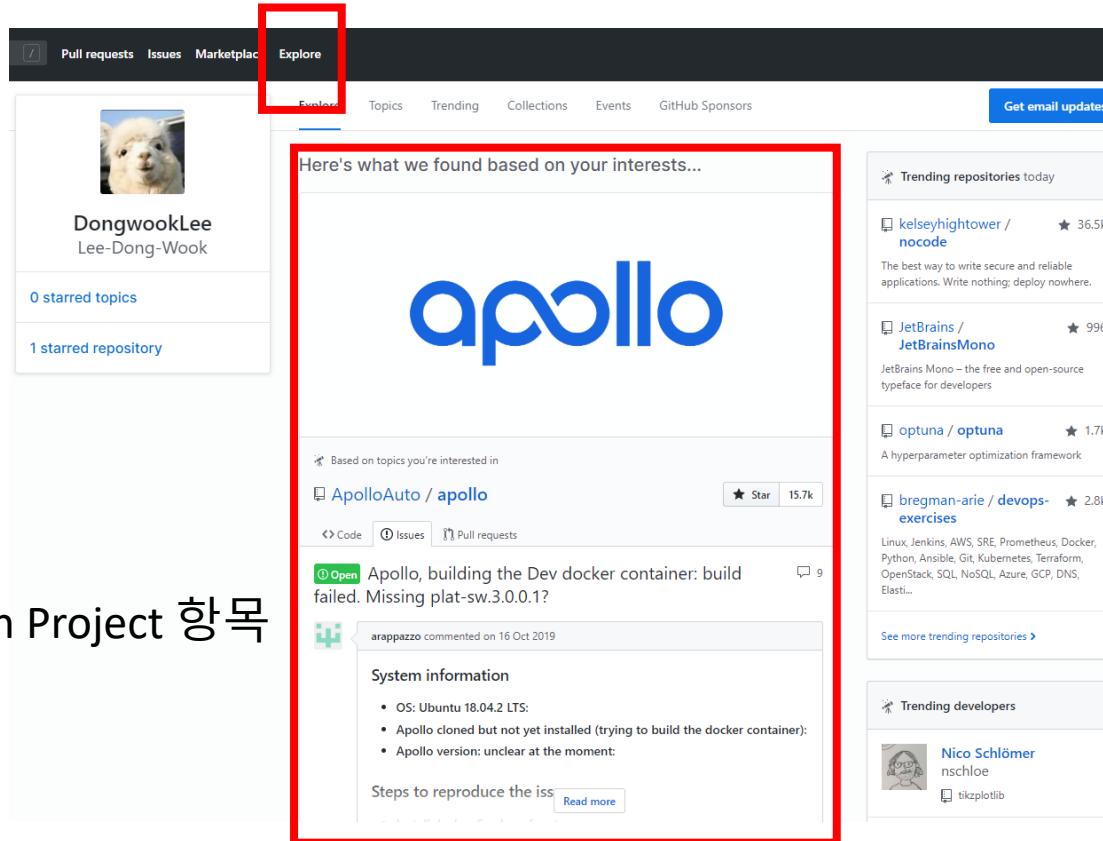
- Use Issue page
 - 프로젝트의 버그 리포트, 기능 제안, 질문 등을 말하며, Github repository 에서 Issue 탭에 들어가면 다양한 토론을 볼 수 있음
- Fork other project
 - 다른 사람의 원격 저장소(remote repository)를 그대로 복사하여 내 계정의 원격 저장소로 만드는 것

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- 마음에 드는 오픈소스 프로젝트 고르기
 - 자신의 프로젝트에 기여하기
 - Github explore 탭에서 자신과 맞는 언어/환경을 사용하는 프로젝트 고르기
 - 흥미가 생기는 프로젝트 고르기
- 오픈소스 프로젝트 수행 시 마음가짐
 - 중대한 버그를 고치거나 기능을 개선/추가에 압박 받을 필요 없음
 - 실제 전체 기여 비율 중 코드 수정 보다 오타 수정, 번역, 디자인 작업 의견 제시 등이 대다수 차지

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

Open Project 탐색 Page 탭



Open Project 항목

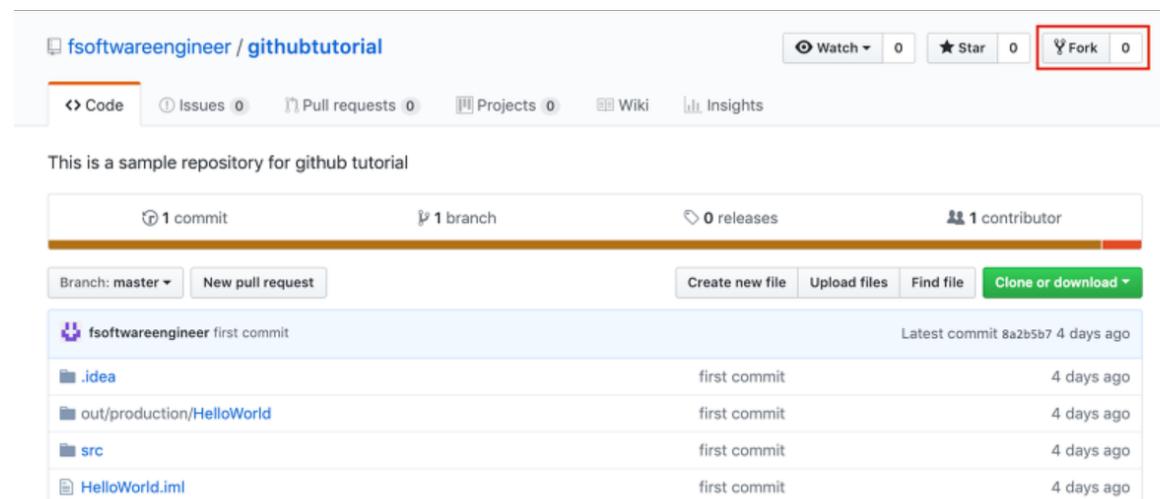
Github open project page

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Fork

- 원하는 오픈소스 프로젝트를 Fork
- Ex) <https://github.com/fsoftwareengineer/githubtutorial> 에 참여 하고 싶다면 Fork 버튼을 눌러 내 원격 저장소에 복사

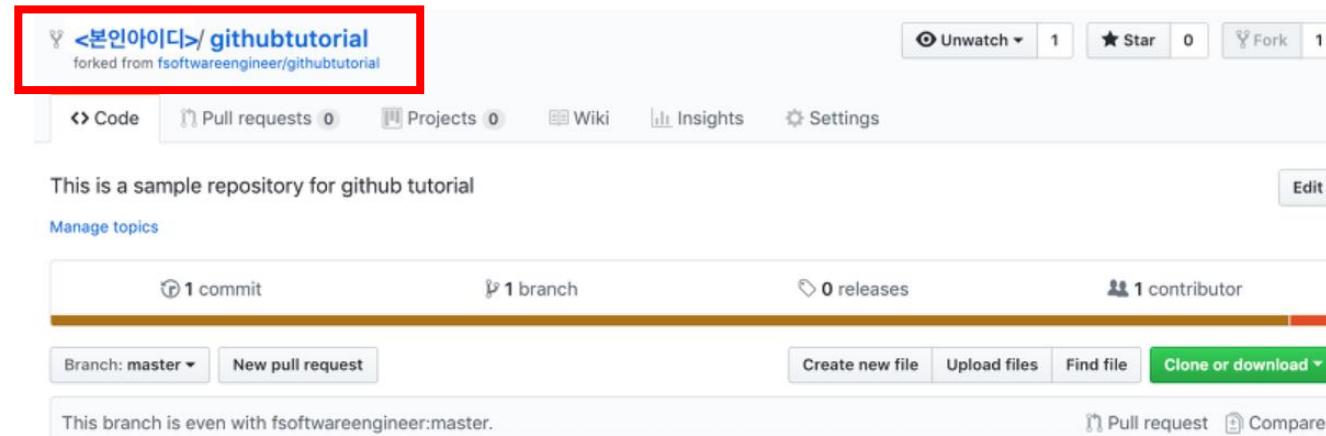
Fork 버튼



Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Fork

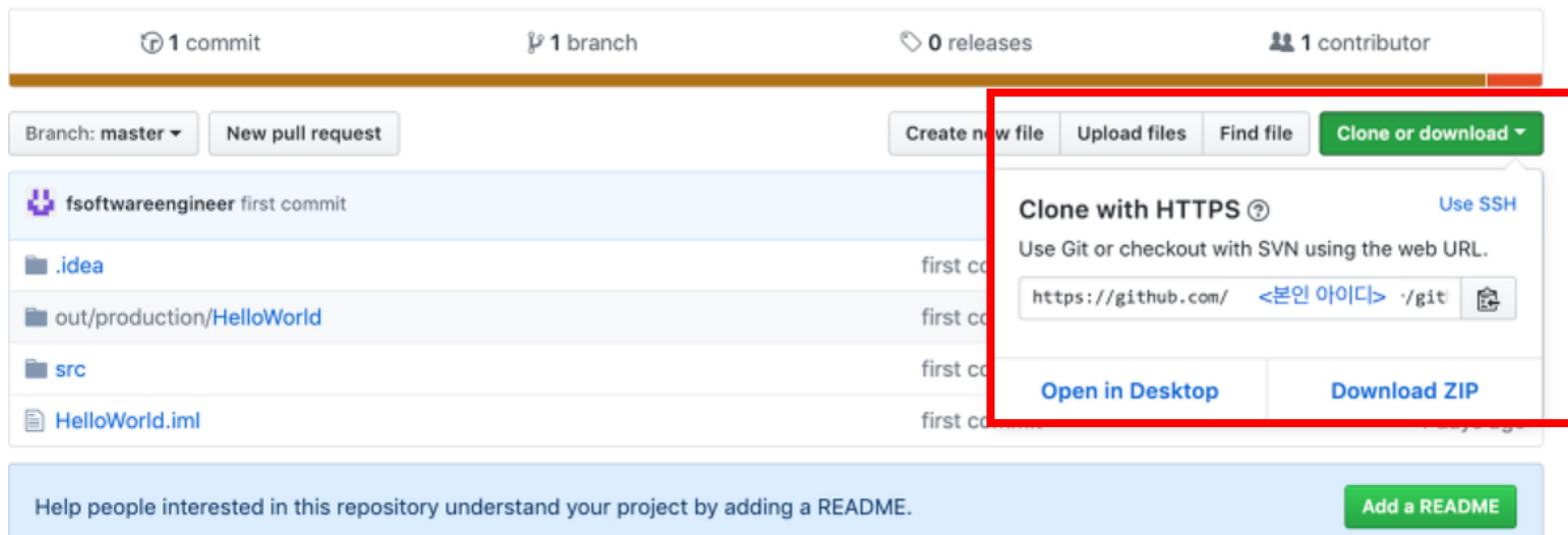
- Fork하는 것은 별도 버전의 원격 저장소를 생성하는 것과 동일(자신의 원격 저장소 페이지로 이동)
- 별도 버전이 생겼다는 것은 Fork 된 버전의 원격 저장소를 clone 및 commit 가능
- <아이디>/<포크 한 원격 저장소 이름>으로 설정 됨



Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Clone

- 원격(remote)환경에 있는 repository를 로컬(local)환경으로 다운로드 하는 것이 clone
- Clone을 하기 위해서 repository의 주소가 필요
 - 주소는 clone or download 버튼을 눌러 찾을 수 있음



Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Clone

- 클론 할 주소를 복사한 뒤 Git-Bash 혹은 CMD 창에서 주소를 입력하여 원하는 디렉토리로 클론
- 클론이 완료되었다면 현재 디렉토리에 githubtutorial 프로젝트 파일이 생성

```
$git clone https://github.com/<본인 아이디>/githubtutorial.git  
Cloning into 'githubtutorial'...  
remote: Enumerating objects: 18, done.  
remote: Counting objects: 100% (18/18), done.  
remote: Compressing objects: 100% (15/15), done.  
remote: Total 18 (delta 0), reused 18 (delta 0), pack-reused 0  
Unpacking objects: 100% (18/18), done.
```

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Issue

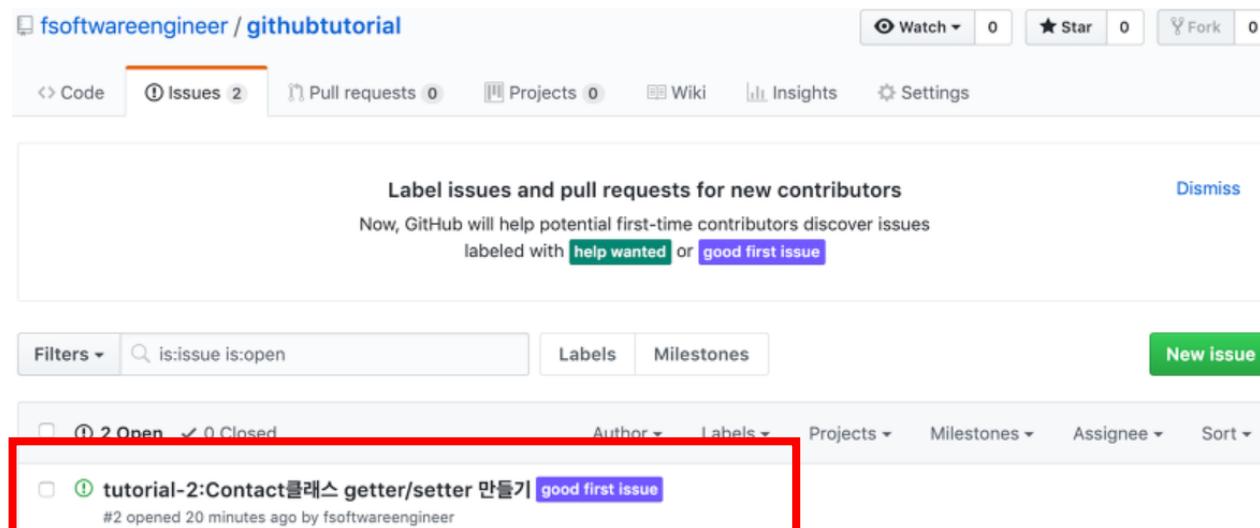
- 오픈소스 프로젝트에는 수많은 수정/추가 사항이 존재
- 수정/추가 사항은 Issues 라는 Issue tracking board에서 확인 가능
- Issues 탭을 누르면 githubtutorial의 Issue board로 이동 가능

The screenshot shows the GitHub repository page for `fsoftwareengineer/githubtutorial`. The top navigation bar includes 'Code', 'Issues 2' (which is highlighted), 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. Below the navigation, there's a modal window titled 'Label issues and pull requests for new contributors' with the message: 'Now, GitHub will help potential first-time contributors discover issues labeled with `help wanted` or `good first issue`'. At the bottom of the page, the 'Issues' tab is selected, showing a list of open issues. The first issue is highlighted with a red box and has the following details:
- Title: `tutorial-2:Contact클래스 getter/setter 만들기`
- Label: `good first issue`
- Description: `#2 opened 20 minutes ago by fsoftwareengineer`
The word '이슈 목록' is written vertically next to the highlighted issue.

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

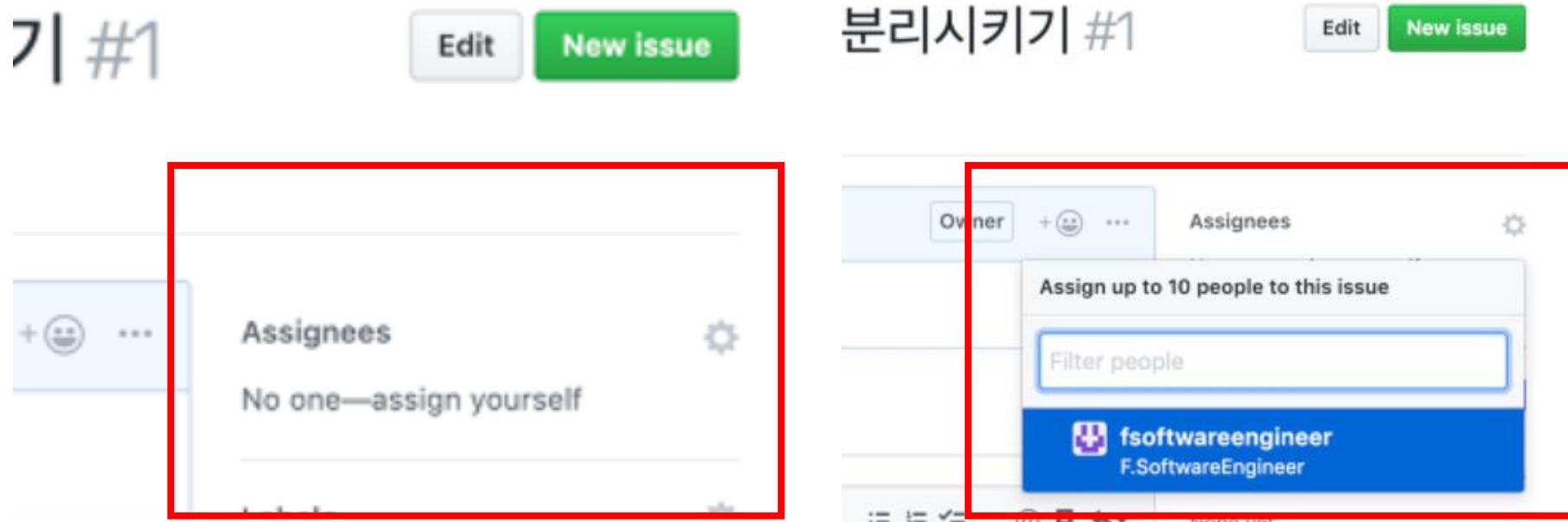
- Github Issue

- 이슈 경우 아래와 같이 [프로젝트 이름]-[이슈번호]-[이슈제목]으로 구성
- Github 경우 몇 가지 태그 존재(Good for first 는 처음 기여하는 개발자에게 좋다는 뜻)
- 이슈 목록에서 원하는 것을 선택



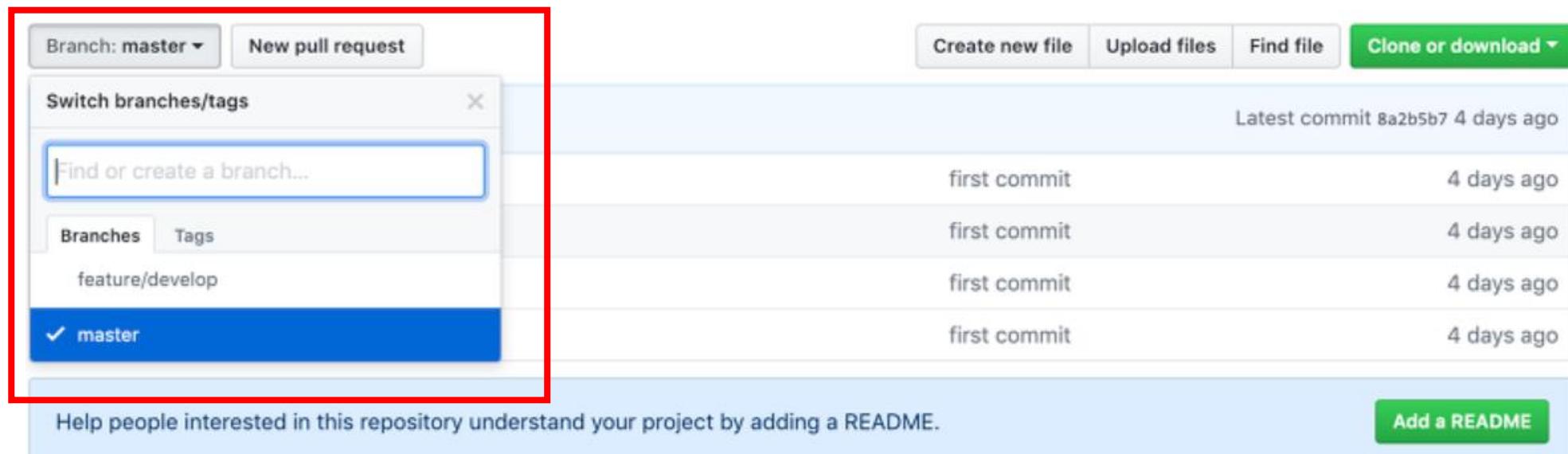
Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Issue
 - 적당하다고 판단될 경우 오른쪽 Assignees를 통해 이슈를 할당 받음



Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Branch
 - Master branch는 보통 production release branch 인 경우가 다수
 - Master branch에 있는 코드가 실제 서비스 중인 코드가 많음
 - Master branch 코드에 자신이 개발하던 것을 그대로 commit 하는 경우는 흔치 않음



Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Branch
 - 색이 있는 동그라미는 commit, 회색 동그라미는 branch off(가지치기) 또는 merge in을 의미
 - Master branch를 release branch로 사용할 경우 개발용 branch를 따로 생성
 - Master branch 외 feature/develop brach 존재(개발용 branch이며 branch off 하였다고 함)

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Branch

- 터미널로 들어가 clone한 프로젝트 내부로 들어갈 수 있음
- 아래 그림을 보면 master branch와 개발용 branch 인 feature/develop branch를 확인 가능

```
$cd githubtutorial  
$git branch  
  
feature/develop  
* master  
(END)
```

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Branch
 - 개발 중인 branch 인 feature/develop branch을 checkout 수행
 - Checkout 이란 자신이 개발할 branch로 switch 하는 것을 의미

```
git checkout feature/develop
Switched to branch 'feature/develop'
Your branch is up to date with 'origin/feature/develop'.
```

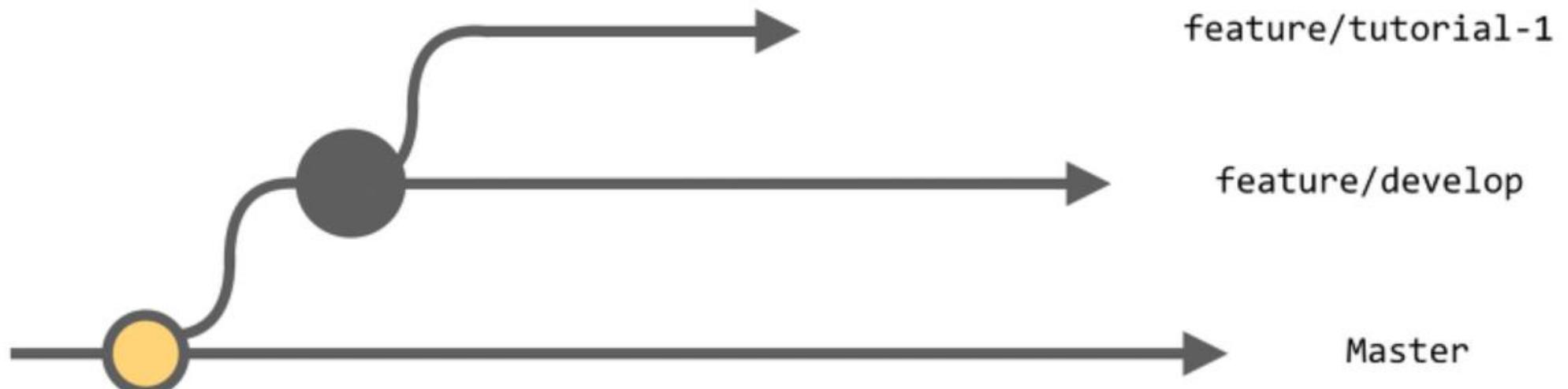
Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Branch
 - 개발용 branch로 이동하였다고 해서 개발을 시작할 수 없음
 - 내가 commit/push할 코드는 나의 개발이 끝날 때까지 다른 사람들의 코드와 분리 되어있어야 함
 - 다른 사람들의 코드와 섞이는 것을 막기위해 “나만의 branch”를 다시 생성
 - Issue branch 경우 해당 branch의 <상위 branch>/<프로젝트 명>-<이슈 번호>로 구성
 - 지금 생성할 Issue branch 경우의 이름은 feature/tutorial – 1

```
git checkout -b feature/tutorial-1
Switched to a new branch 'feature/tutorial-1'
```

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Branch
 - 현재 branch 타임라인은 다음과 같음



Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Branch

- Feature/tutorial – 1 branch에서 Issue에 해당하는 코드를 수정
- 수정 완료 시 git add을 통해 파일을 로컬 저장소에 추가
- Git commit을 이용하여 commit 수행
- Commit 수행 시 commit message을 “이슈 제목”로 함

```
:githubtutorial feature/tutorial-1 X 4d △ ✓ ⊖ → git add src/Contact.java  
:githubtutorial feature/tutorial-1 X 4d △ ✓ → git add src/Main.java
```

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Branch

- Commit 수행 시 commit message을 “이슈 제목”로 함
- Commit 수행 후 push 진행
- Push 수행 시 개발을 위해 만든 branch가 branch 목록에 출력됨
- 본인의 commit hash code을 기억해야 함(ex) 8aeb5b7)

```
→ git commit -m "tutorial-1:separate Contact from Main class"
[feature/tutorial-1 8aeb5b7] tutorial-1:separate Contact from Main class
 3 files changed, 14 insertions(+), 6 deletions(-)
    create mode 100644 .gitignore
    create mode 100644 src/Contact.java
```

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

```
→ git push --set-upstream origin feature/tutorial-1

    Counting objects: 6, done.

    Delta compression using up to 4 threads.

    Compressing objects: 100% (6/6), done.

    Writing objects: 100% (6/6), 582 bytes | 582.00 KiB/s, done.

    Total 6 (delta 2), reused 0 (delta 0)

    remote: Resolving deltas: 100% (2/2), completed with 2 local objects.

    remote:

    remote: Create a pull request for 'feature/tutorial-1' on GitHub by visiting:

    remote:     https://github.com/fsoftwareengineer/githubtutorial/pull/new/feature/tutorial-1

    remote:

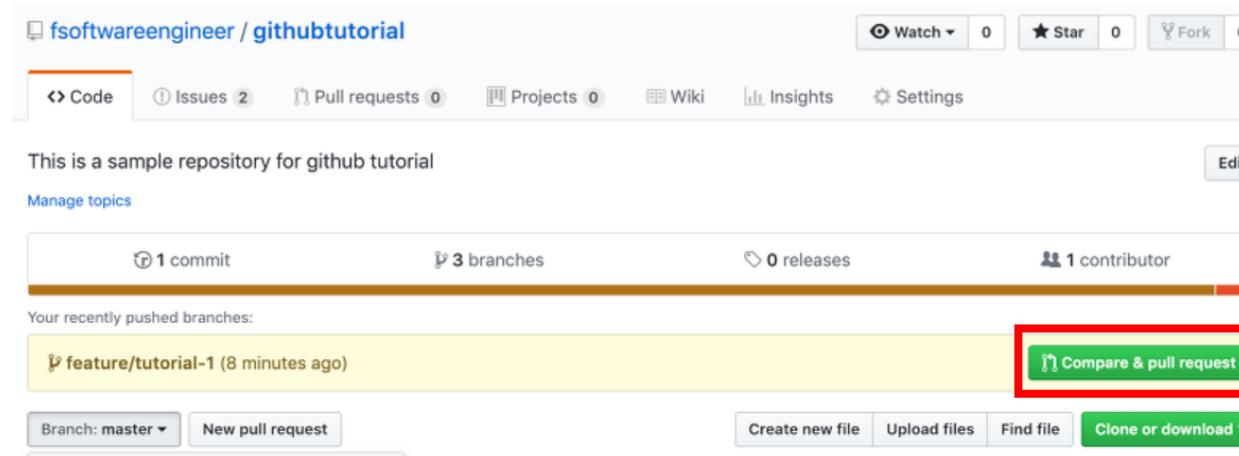
    To https://github.com/fsoftwareengineer/githubtutorial.git
      * [new branch]      feature/tutorial-1 -> feature/tutorial-1

    Branch 'feature/tutorial-1' set up to track remote branch 'feature/tutorial-1' from 'origin'.
```

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Pull Request

- 지금 까지 feature/tutorial-1 branch 에서 개발을 진행
- 개발이 완료되어 해당 branch를 remote repository에 push(새로 만들어진 branch가 github에 반영)
- Merge 단계 수행(내 branch(feature/tutorial-1) 와 feature/develop branch를 합침)
 - Pull Request 요청(관리자 또는 리드 개발자에게 요청)



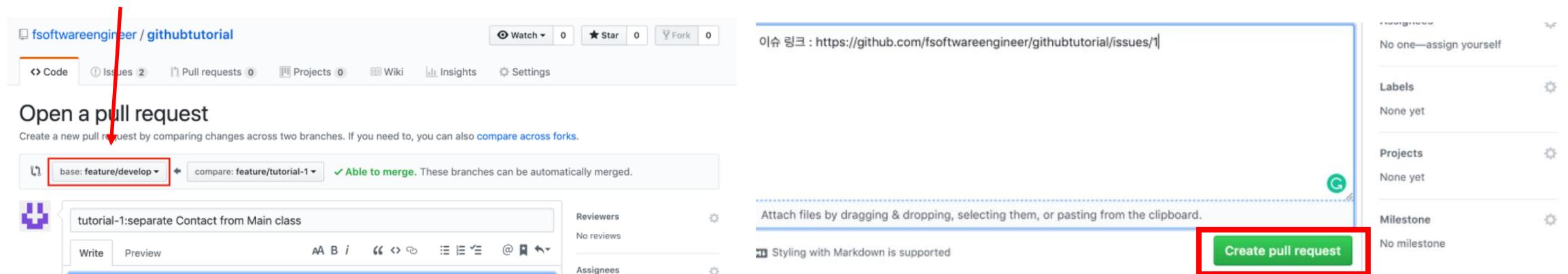
Pull Request 요청

Github란?(Github에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Pull Request

- 어떤 branch에 자신의 branch를 merge 시킬지 선택(base는 보통 자신이 branch off한 branch 선택)
 - Feature/develop branch에서 branch off 하여 feature/tutorial-1 branch 생성
 - 상단에 Pull Request 제목 작성, Write에 목적 작성

Merge 할 branch 설정



Pull Request 요청

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Pull Request

- 잘못된 파일이 commit 되었는지 다시 확인

기존 파일과 변경된 사항을 확인 가능



5 src/Contact.java

```
@@ -0,0 +1,5 @@
+ public class Contact {
+     String name;
+     String email;
+     String phoneNumber;
+ }
```

6 src/Main.java

```
@@ -85,10 +85,4 @@
85     System.out.println("Removed " + c.name);
86 }
87 }

88 -
89 -     class Contact {
90 -         String name;
91 -         String email;
92 -         String phoneNumber;
93 -     }
94 }
```

Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Pull Request

- 성공적인 pull Request 화면
- 프로젝트 관리자가 제출한 Pull Request을 검토
 - Master branch 타임라인에서 현재 상태는 Pending review 상태

fsoftwareengineer / githubtutorial

Watch 0 Star 0 Fork 0

Issues 2 Pull requests 1 Projects 0 Wiki Insights Settings

tutorial-1:separate Contact from Main class #3

Open fsoftwareengine... wants to merge 1 commit into feature/develop FROM feature/tutorial-1

Conversation 0 Commits 1 Checks 0 Files changed 3 +14 -6

fsoftwareengine... commented a minute ago • edited • Owner + 3 ...

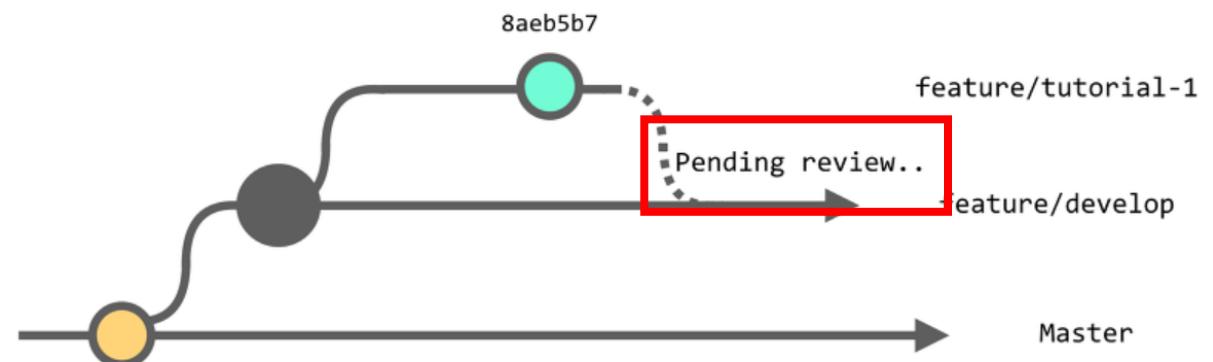
이슈 링크 : #1

tutorial-1:separate Contact from Main class 77b193c

Add more commits by pushing to the feature/tutorial-1 branch on fsoftwareengineer/githubtutorial.

Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.

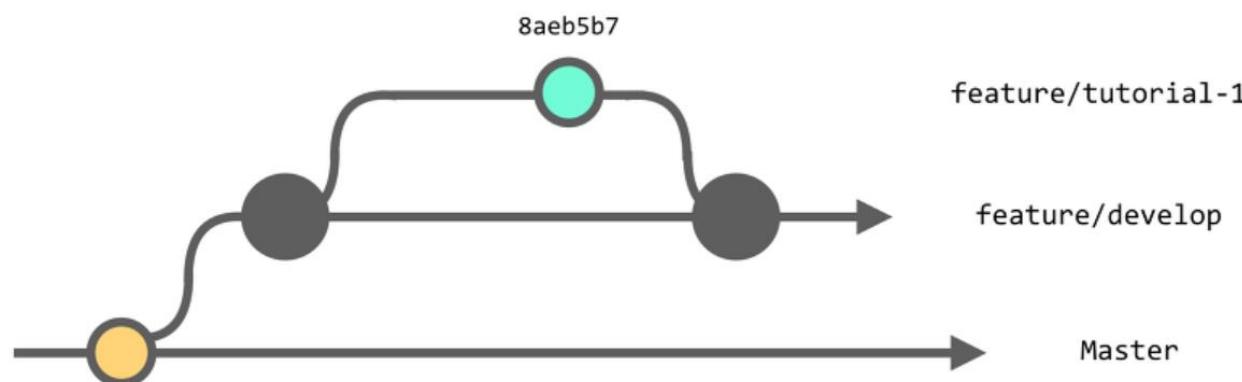
This branch has no conflicts with the base branch
Merging can be performed automatically.



Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Review

- 관리자의 검토에 따라 Merge 혹은 Reject(반려)
- Reject 될 경우 거부 사유가 명시
- 리뷰 과정이 끝나고 코드가 검토에서 통과될 경우 base branch와 Merge 수행
 - 다음 릴리즈에 내가 개발한 코드가 함께 출시
- 아래는 master branch 의 타임라인



Github란?(Github 에서 활성화된 오픈소스 프로젝트 참여하기)

- Github Review

- Pull quest의 요청이 검토 후 Merge 된 화면

tutorial-1:separate Contact from Main class #3

Merged fsoftwareengine... merged 1 commit into feature/develop from feature/tutorial-1 19 seconds ago

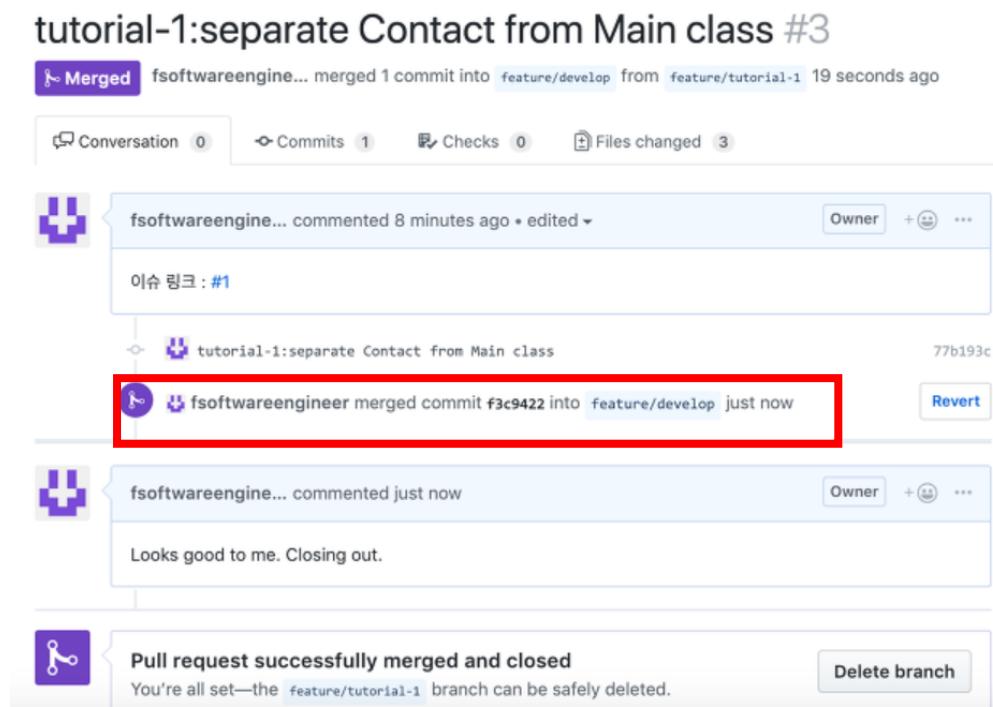
Conversation 0 Commits 1 Checks 0 Files changed 3

fsoftwareengine... commented 8 minutes ago * edited ▾ Owner + ...
이슈 링크 : #1

tutorial-1:separate Contact from Main class 77b193c
fsoftwareengineer merged commit f3c9422 into feature/develop just now 

fsoftwareengine... commented just now
Looks good to me. Closing out.

Pull request successfully merged and closed
You're all set—the feature/tutorial-1 branch can be safely deleted. Delete branch



tutorial-1: Contact클래스를 메인 클래스에서 분리시키기 #1

Closed fsoftwareengineer opened this issue 2 hours ago · 1 comment

fsoftwareengine... commented 2 hours ago * edited ▾ Owner + ...
Contact클래스를 메인클래스로부터 분리시키기

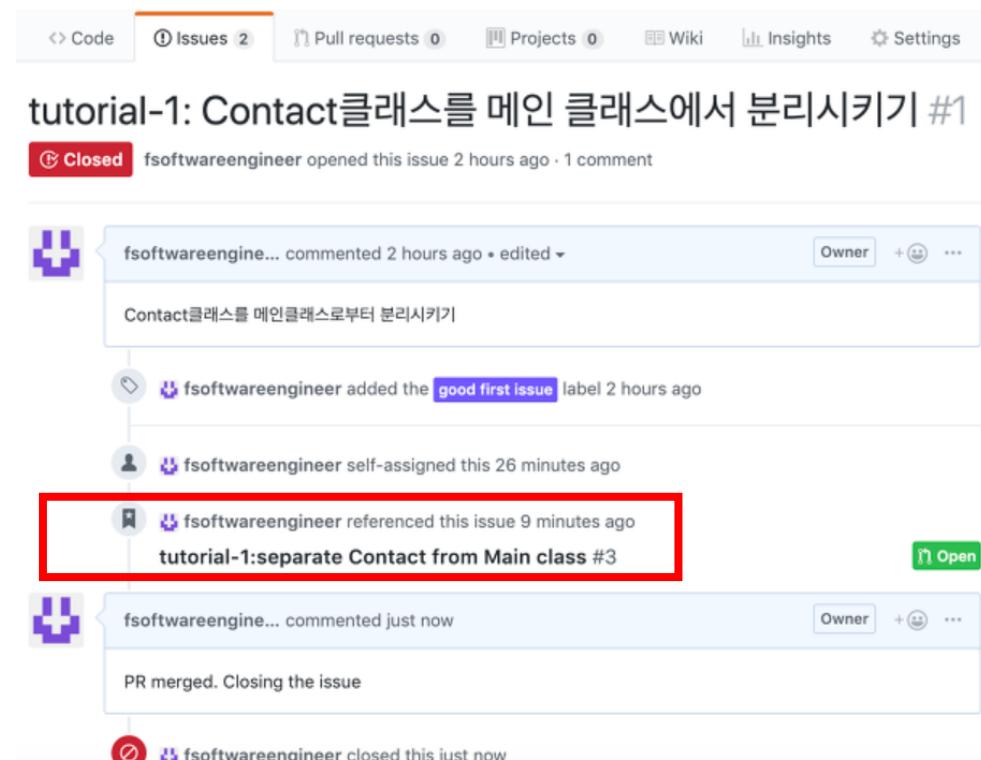
fsoftwareengineer added the good first issue label 2 hours ago

fsoftwareengineer self-assigned this 26 minutes ago

fsoftwareengineer referenced this issue 9 minutes ago **tutorial-1:separate Contact from Main class #3** 

fsoftwareengine... commented just now
PR merged. Closing the issue

fsoftwareengineer closed this just now



Github 실습 예제

Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Github 실습 예제는 Github desktop 프로그램을 사용하여 진행
- Github desktop 은 CMD 나 Git-bash을 통해 command line을 이용하지 않음
- GUI을 통해 원격 저장소(remote repository)에 코드 업로드 가능
 - 사용자 친화적으로 손쉽게 코드 확인/관리/업로드 가능
- 본 실습에서 지난 실습 때 생성한 원격 저장소에 branch을 생성, Pull Request 요청을 통해 Merge 하는 것을 목표

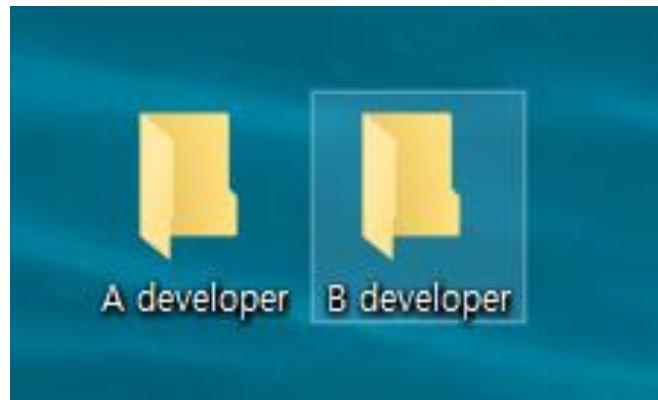
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 기존 프로젝트에서 branch 새로 만들기
 - 2명의 개발자가 개발하는 것을 가정하여 A developer, B developer로 branch 생성

The screenshot shows a GitHub repository named 'Lee-Dong-Wook / gitest'. The repository has 3 commits, 2 branches, 0 packages, 0 releases, and 1 contributor. The 'Branch: master' dropdown is selected. A modal window titled 'Switch branches/tags' is open, showing the 'Find or create a branch...' input field and a list of branches. The 'master' branch is checked, and the 'A-developer' branch is listed below it. The 'B developer' branch is also visible in the list.

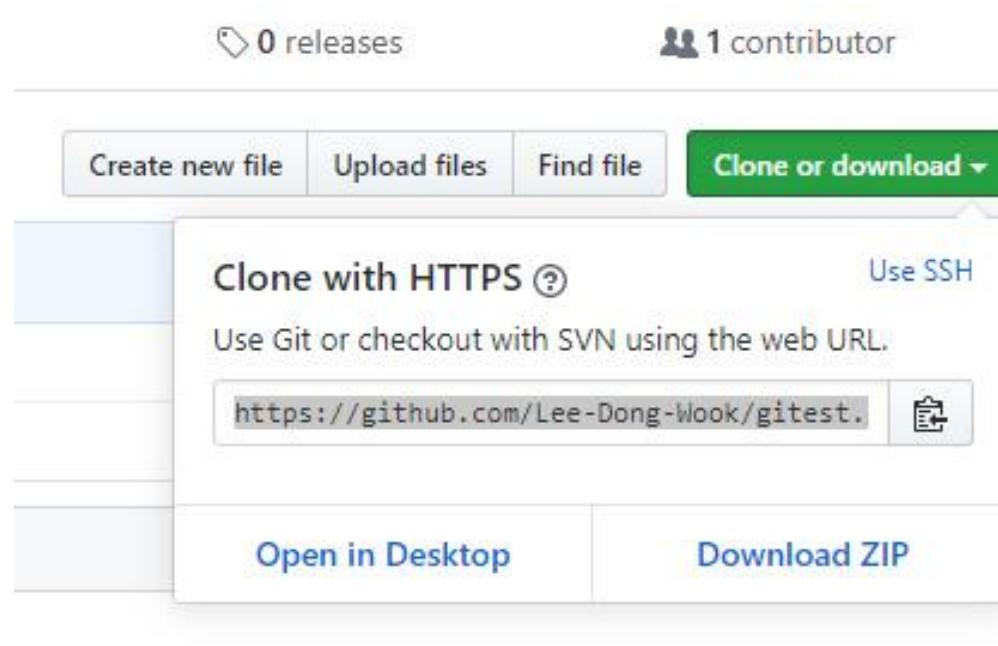
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 여러 사람이 개발하는 것을 가정하여 이전에 생성된 branch 이름으로 폴더 생성
 - 바탕화면에 프로젝트 파일 생성



Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 기존 프로젝트 URL 복사
 - Clone을 하기 위해 기존 프로젝트의 URL을 복사
 - Open in Desktop 혹은 Download Zip을 통해 프로젝트 파일 다운로드 가능



Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Github desktop 설치

- <https://desktop.github.com/>



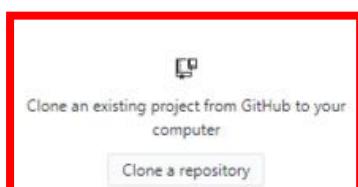
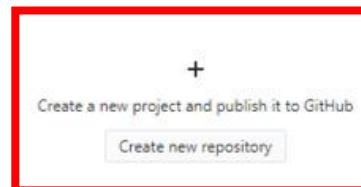
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Github desktop 실행



선택된 원격 저장소

로컬 저장소에 기존프로젝트 클론 및 Github에 publish

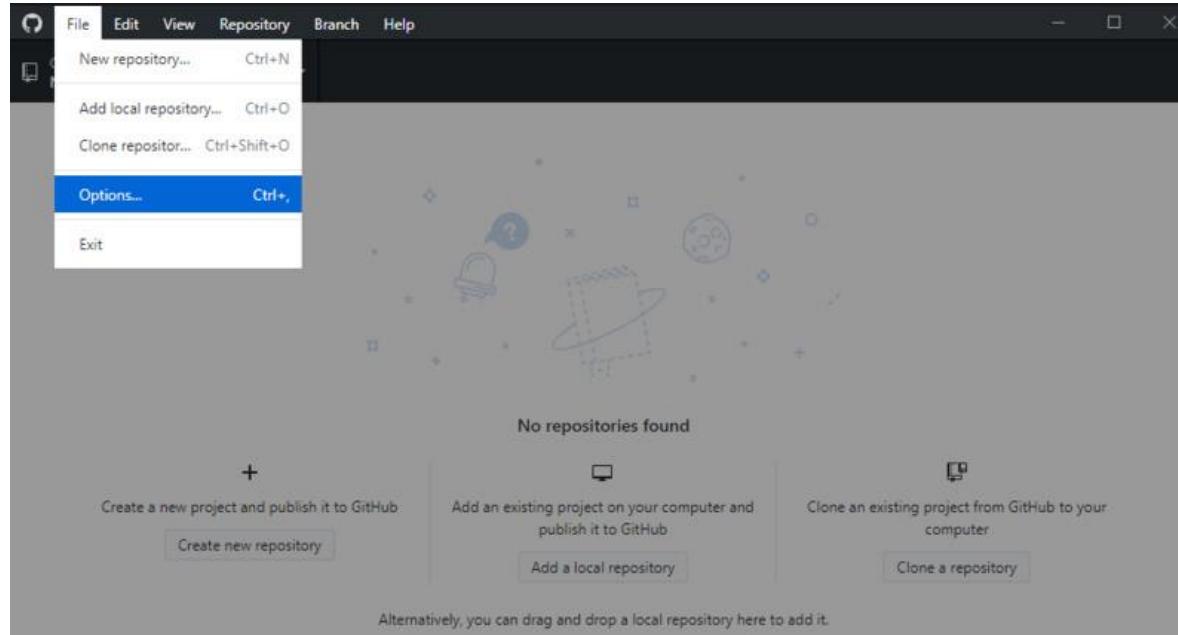


새 원격 저장소 생성

로컬 저장소에 기존프로젝트 클론

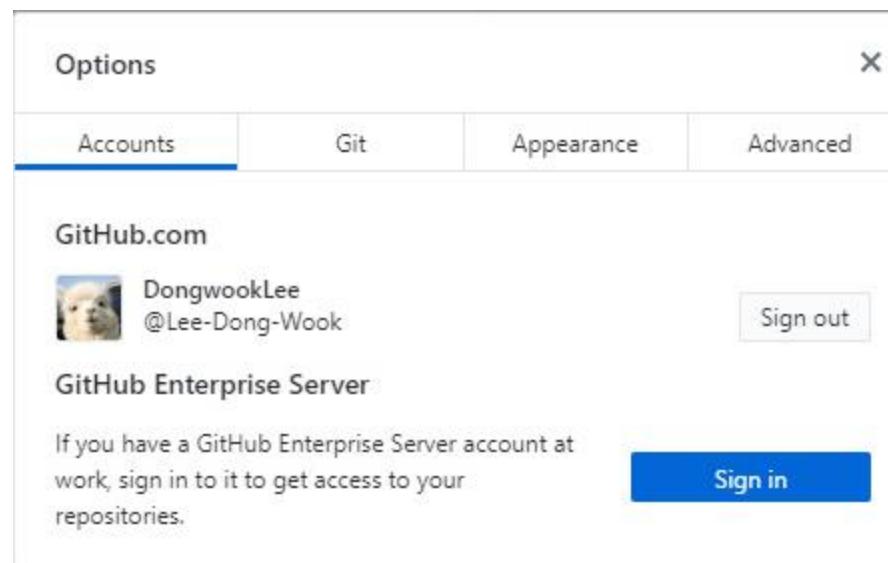
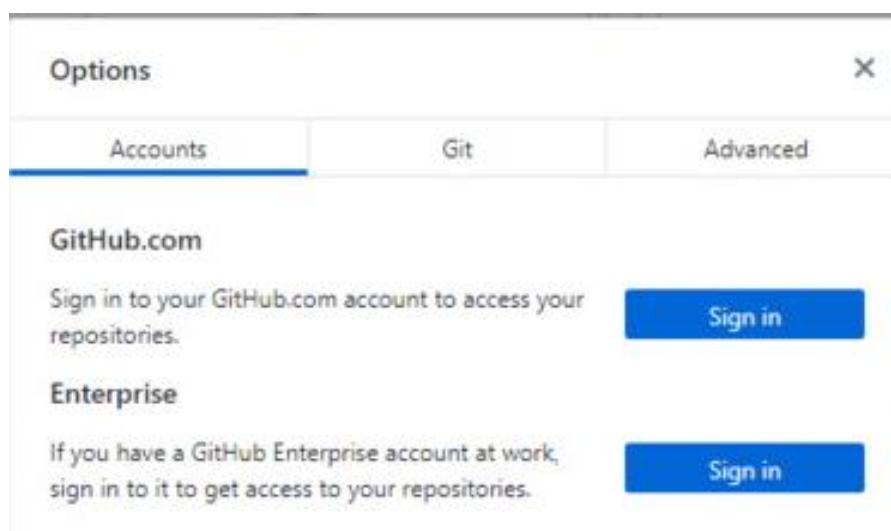
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Github desktop 실행
 - 최초 시작 시 로그인 필요



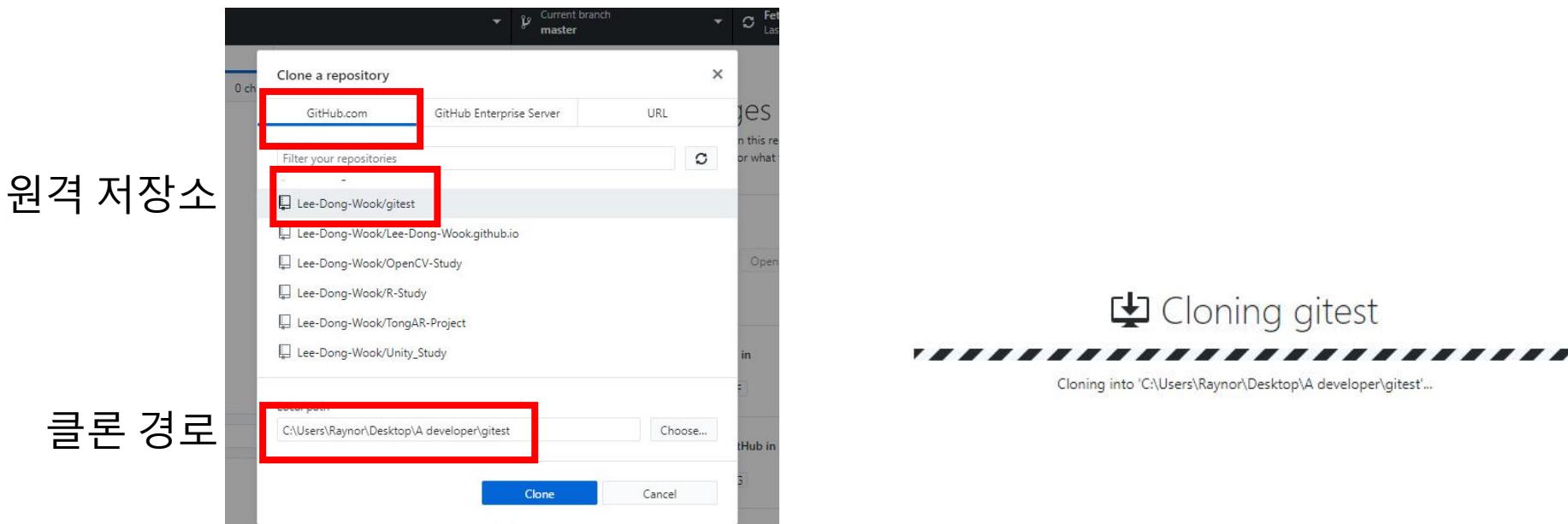
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Github desktop 실행
 - File – Option로 이동 후 Account 탭에서 Github.com 항목의 Sign in 버튼을 통해 로그인 수행



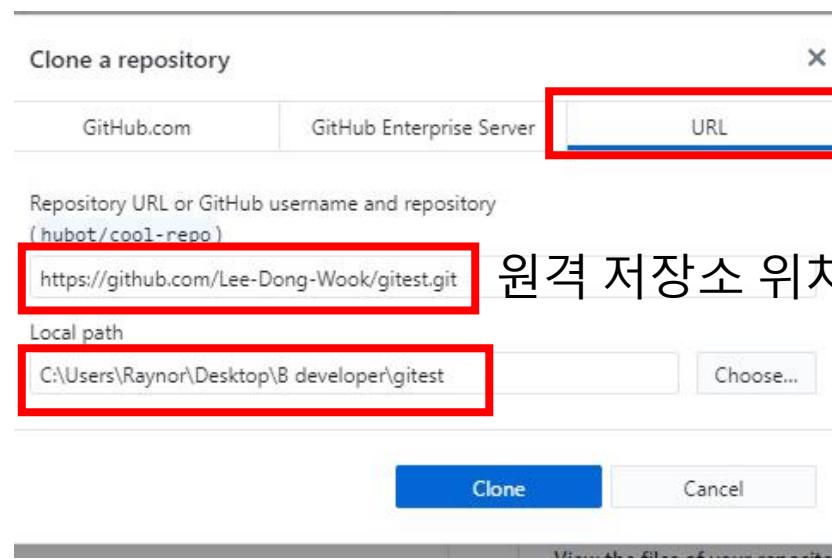
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 저장소 Clone 하기(A 개발자)
 - 다양한 Clone 방법 존재
 - File – Clone a repository – Github.com 항목 선택 – 원하는 원격 저장소 선택 – clone할 경로 선택



Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 저장소 Clone 하기(B 개발자)
 - 다양한 Clone 방법 존재
 - File – Clone a repository – URL 항목 선택 – 원하는 원격 저장소 주소 입력 – clone할 경로 선택

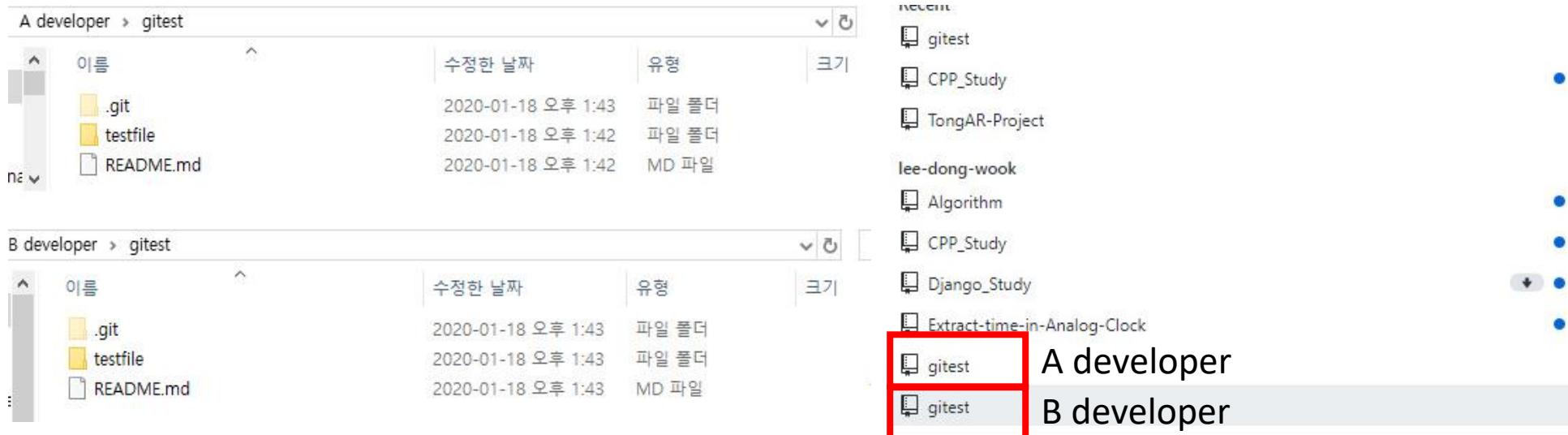


클론 경로



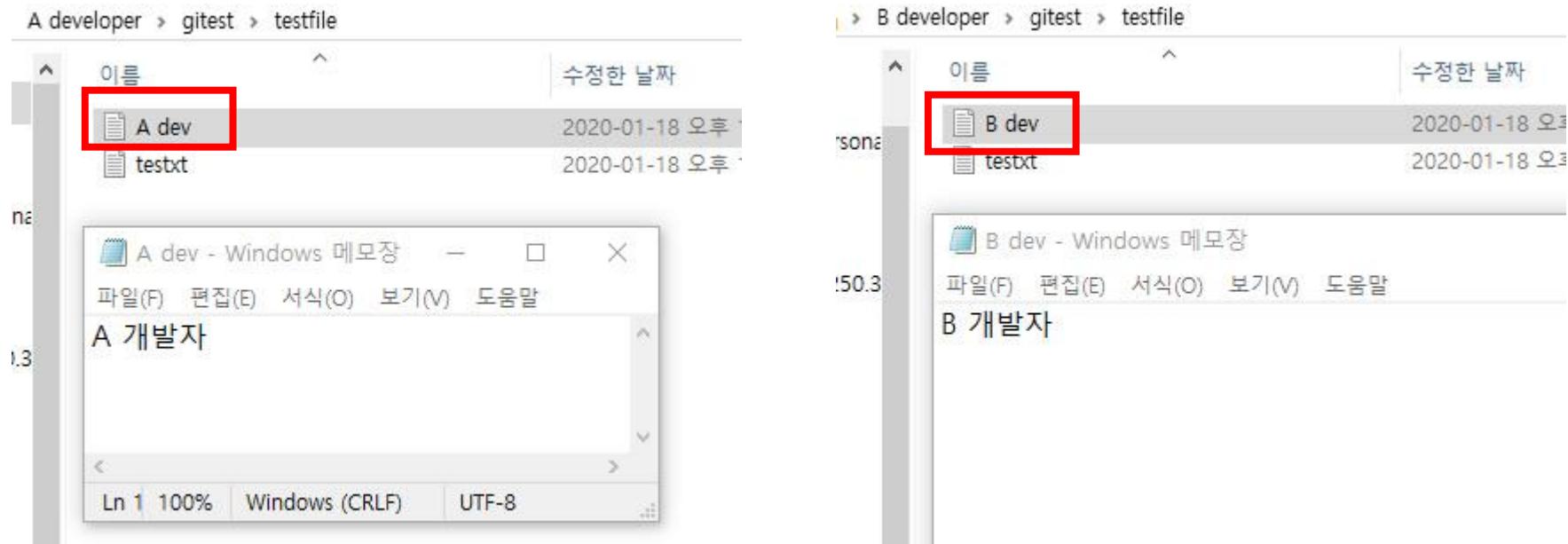
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 바탕화면에 생성한 각 폴더 내 동일 프로젝트가 생성된 것을 확인
 - A 개발자와 B 개발자가 동시 개발 상황을 가정
- Github desktop 에는 현재 clone한 프로젝트 표시
 - 한 PC에 동일 프로젝트를 2번 clone 했기 때문에 동일한 이름으로 표시



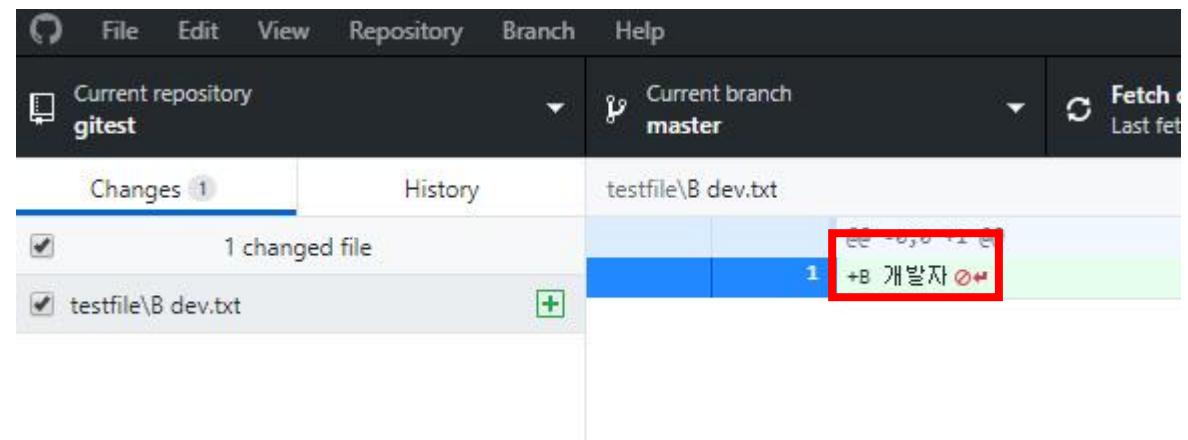
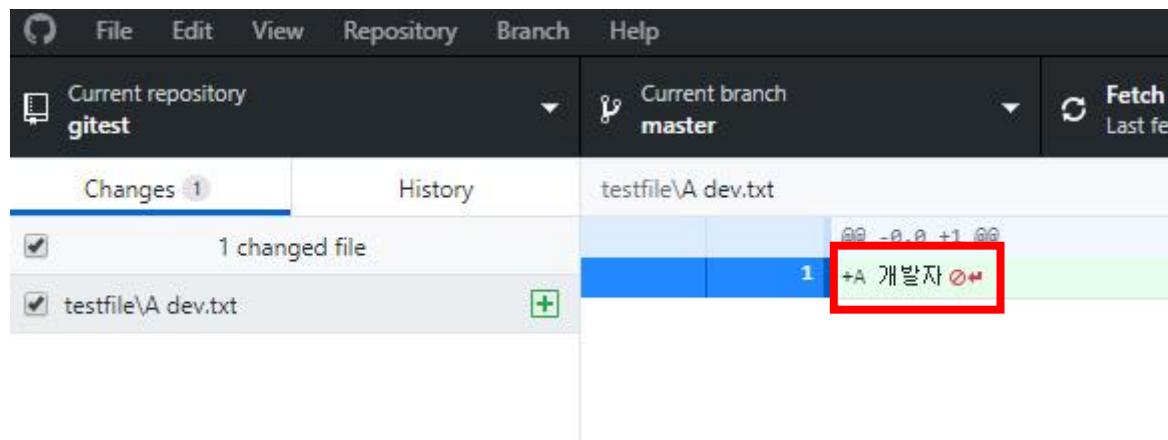
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- A 개발자, B개발자 각자 동일 프로젝트 내에서 다른 코드 작성



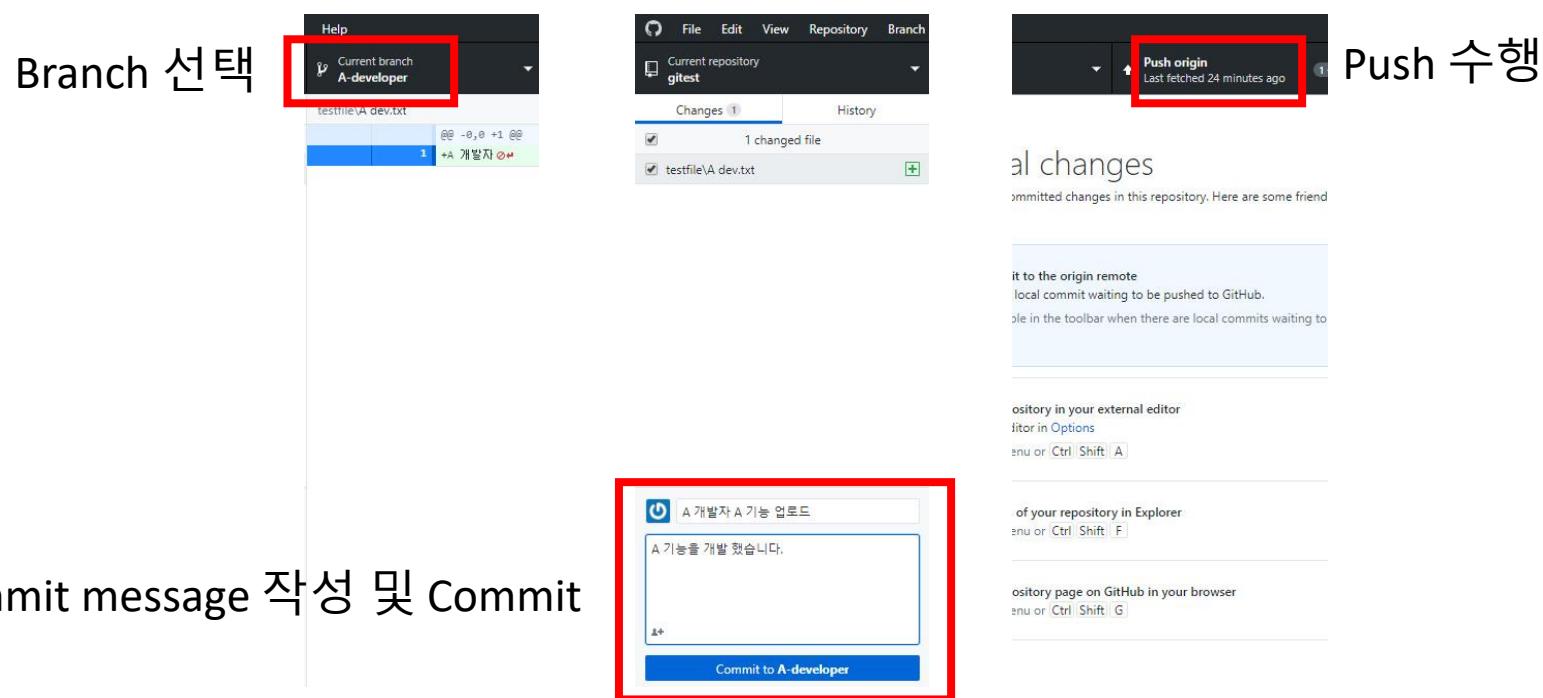
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Github desktop 에는 clone 했던 프로젝트 내용과 작성한 코드 내용을 비교하여 변경 내용을 표시
- 각 개발자에 따라 변경 내용이 다름



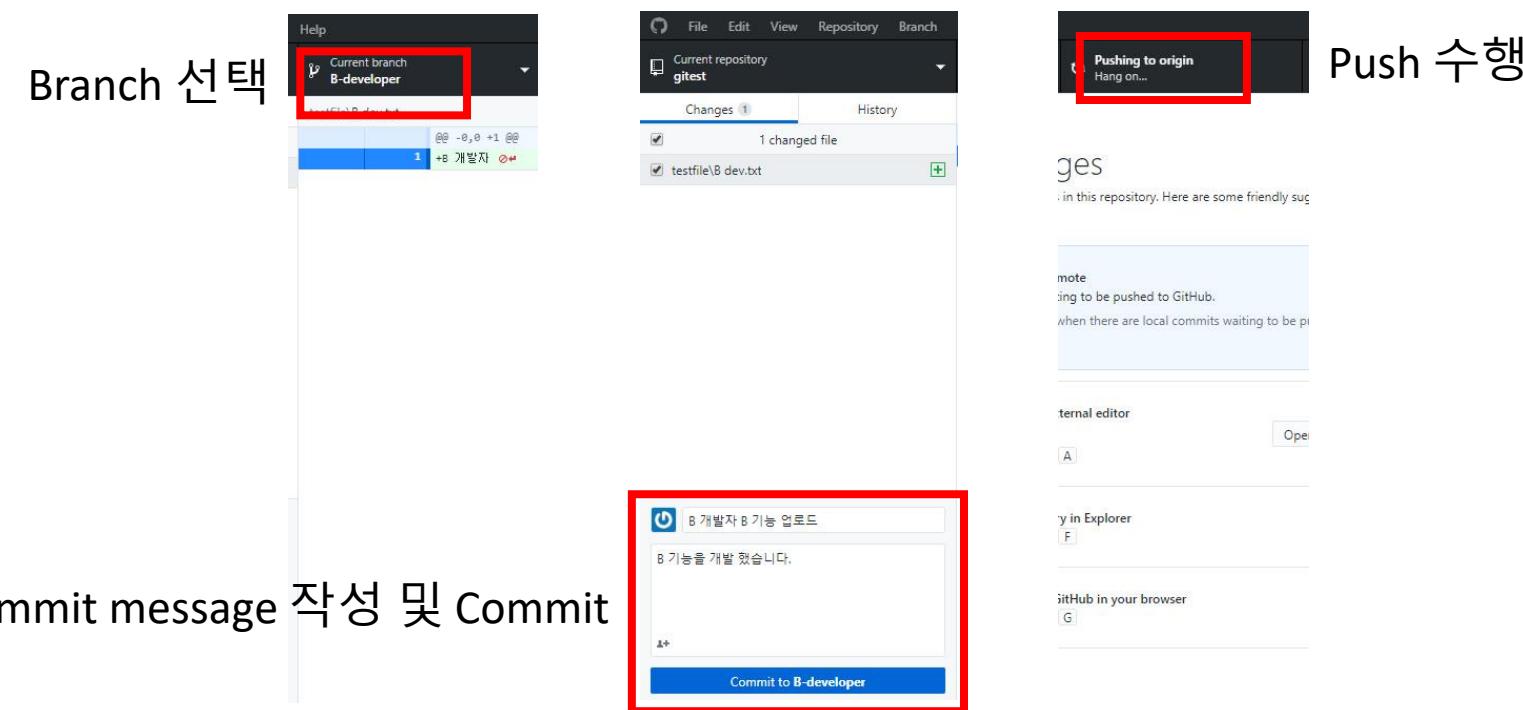
Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 개발자(A developer) 자신의 branch을 선택(파일 유지)
- commit message을 작성 후 commit 수행
- Commit 수행 후 Push origin 버튼을 눌러 원격 저장소로 Push 수행



Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 개발자(B developer) 자신의 branch을 선택(파일 유지)
- commit message을 작성 후 commit 수행
- Commit 수행 후 Push origin 버튼을 눌러 원격 저장소로 Push 수행



Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- 원격 저장소에는 A 개발자, B개발자의 branch로 push 가 된 것을 확인
- Compare & Pull request을 통해 관리자에게 검토 요청

The screenshot shows a GitHub repository page for 'Lee-Dong-Wook / gitest'. The top navigation bar includes 'Unwatch' (1), 'Star' (0), and 'Fork' (0) buttons. Below the navigation, there are links for 'Code', 'Issues 0', 'Pull requests 0', 'Actions', 'Projects 0', 'Wiki', 'Security', 'Insights', and 'Settings'. The main content area displays the repository name 'Lee-Dong-Wook / gitest' and a commit message 'tttt'. There is a 'Manage topics' link and an 'Edit' button. Below these, summary statistics are shown: '3 commits', '3 branches', '0 packages', '0 releases', and '1 contributor'. A red box highlights the 'Your recently pushed branches:' section, which lists two entries: 'A-developer (5 minutes ago)' and 'B-developer (less than a minute ago)'. To the right of each entry is a green 'Compare & pull request' button, also highlighted with a red box.

Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Pull Request 요청

Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Pull Request 요청 목록

- 예제에서는 한 계정으로 다른 branch을 이용하여 여러 사람이 Pull Request 요청 상황을 재현
- 실제 개발 상황은 각 branch마다 다른 개발자가 Pull Request 요청

The screenshot shows a GitHub repository page for 'Lee-Dong-Wook / gitest'. The top navigation bar includes 'Code', 'Issues 0', 'Pull requests 2' (which is highlighted in orange), 'Actions', 'Projects 0', 'Wiki', 'Security', 'Insights', and 'Settings'. On the right side, there are buttons for 'Unwatch 1', 'Star 0', 'Fork 0', and a 'Dismiss' button for a modal window.

The main content area displays a modal window titled 'Label issues and pull requests for new contributors'. It contains the text: 'Now, GitHub will help potential first-time contributors discover issues labeled with good first issue' and a 'Dismiss' button.

Below the modal, there are filters: 'Filters ▾' (set to 'is:pr is:open'), a search bar ('is:pr is:open'), 'Labels 9', 'Milestones 0', and a green 'New pull request' button.

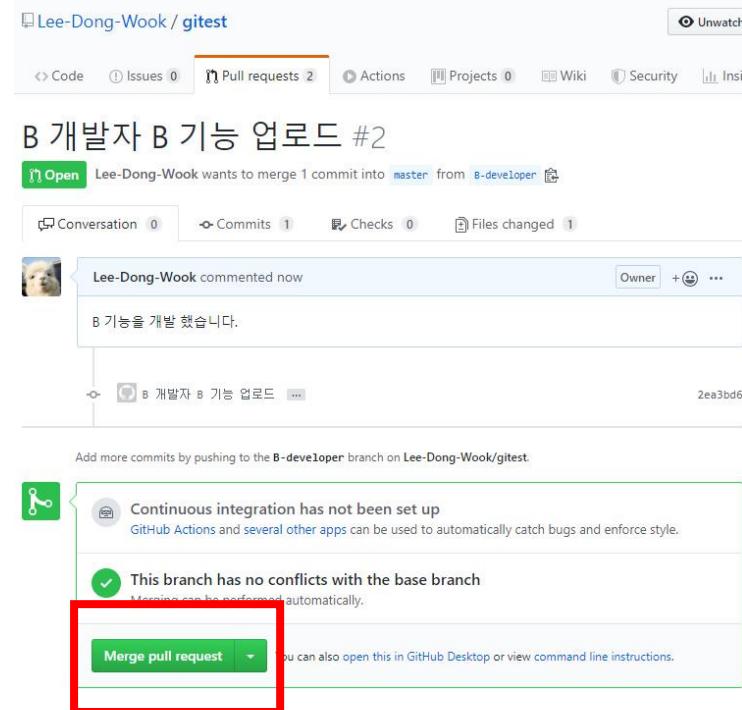
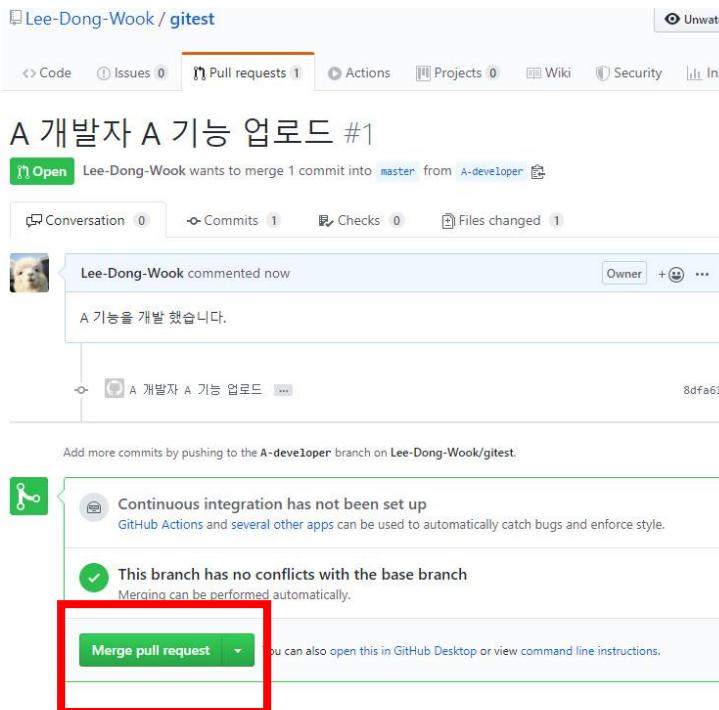
The pull request list shows two items:

- #2 opened 18 seconds ago by Lee-Dong-Wook: 'B 개발자 B 기능 업로드'
- #1 opened 2 minutes ago by Lee-Dong-Wook: 'A 개발자 A 기능 업로드'

The first pull request (#2) is highlighted with a red box.

Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Merge pull request 버튼을 눌러 merge 수행



Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- Merge 성공

The image contains two side-by-side screenshots of a GitHub repository interface, illustrating a successful merge process.

Left Screenshot (A developer's view):

- Title:** A 개발자 A 기능 업로드 #1
- Status:** Merged
- Comment:** Lee-Dong-Wook commented 2 minutes ago: A 기능을 개발 했습니다.
- Commit:** A 개발자 A 기능 업로드 ... by Lee-Dong-Wook (commit hash: 8dfa616)
- Merge:** Lee-Dong-Wook merged commit ab40f53 into master now
- Message:** Pull request successfully merged and closed. You're all set—the A-developer branch can be safely deleted.

Right Screenshot (B developer's view):

- Title:** B 개발자 B 기능 업로드 #2
- Status:** Merged
- Comment:** Lee-Dong-Wook commented 1 minute ago: B 기능을 개발 했습니다.
- Commit:** B 개발자 B 기능 업로드 ... by Lee-Dong-Wook (commit hash: 2ea3bd6)
- Merge:** Lee-Dong-Wook merged commit dd00eca into master now
- Message:** Pull request successfully merged and closed. You're all set—the B-developer branch can be safely deleted.

Github Desktop을 이용해 쉽게 원격 저장소에 파일 올리기

- A, B 개발자가 각각 개발한 내용을 하나의 프로젝트에 모두 적용된 모습

The screenshot shows a GitHub repository named 'Lee-Dong-Wook / gitest'. The repository has 1 unwatched star and 0 forks. The navigation bar includes links for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security, Insights, and Settings. The branch is set to master. The main area displays a list of commits:

Commit	Message	Time
A dev.txt	A 개발자 A 기능 업로드	20 minutes ago
B dev.txt	B 개발자 B 기능 업로드	15 minutes ago
testxt.txt	gitest 작업에 대한 실시	23 hours ago

The commit for developer B ('B dev.txt') is highlighted with a red box.

참조(Reference)

- 오픈 소스 참여하기 주제
 - 슬라이드 12~33
 - 출처 : <https://imasoftwareengineer.tistory.com/5>