



Git / GitHub

(2주차)

송 상 효

samsong@skku.edu

Contents

I
Git/GitHub
Intro

II
Hands-on
(Basic)

III
Hands-on
(Advanced)

IV
팀프로젝트
준비

Contents

I
Git/GitHub
Intro

II
Hands-on
(Basic)

III
Hands-on
(Advanced)

IV
팀프로젝트
준비

git 이란?

개발과정, 소스파일 등을 관리하는 도구

History 관리가 되어 개발 되어온 과정, 역사를 볼 수 있고, 특정시점으로 복구 가능

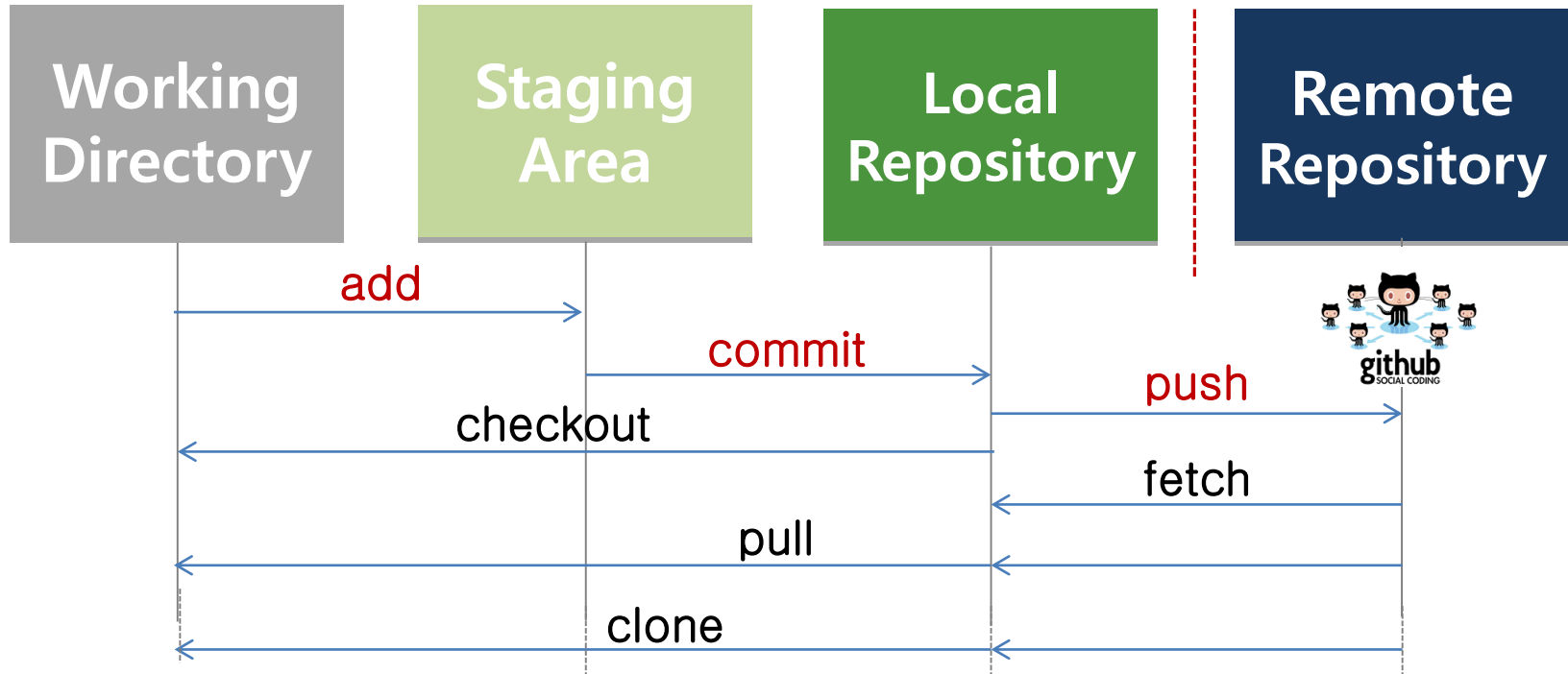
왜 git 을 써야할까?

개발자로서 평생 간단한 코드만 짤 순 없기 때문에

가면 갈수록 점점 더 복잡하고 관리하기 힘든 코드를 다루는데 도움을 줄 강력한 도구!

혹은 팀 프로젝트에 있어서 협업을 도와 줄 수 있는 유용한 도구!

Git 필수 개념 – Git의 세 가지 상태



▣ Git을 통한 작업 순서

- 워킹 디렉토리에서 파일을 수정
- 워킹 디렉토리에서 변경된 파일을 **스테이징 영역에 추가**(커밋할 스냅샷 생성)
- 스테이징 영역의 파일을 커밋하여 **Git 디렉토리에 영구적으로 저장**



GitHub 이란?

네트워크 상 에서 내가 원하는 콘텐츠를 관리할 수 있는 서비스

프로젝트 협업, 오픈소스 커뮤니티 등 다른 사람과의 커뮤니케이션 공간

왜 GitHub 을 써야 할까?

다른 개발자들과의 커뮤니케이션 공간

회사 및 연구소에서의 프로젝트 관리 협업 툴, 다른 오픈소스 커뮤니티 참여

SNS(Facebook, 인스타그램 등) 의 개발자 버전

Contents

I
Git/GitHub
Intro

II
**Hands-on
(Basic)**

III
Hands-on
(Advanced)

IV
팀프로젝트
준비

Step0—지난 배운 것

■ 지난 배운 것에 대해 생각해보기

- (1) Git 설치
- (2) Git 로컬 저장소 만들기
- (3) Git 로컬 저장소 초기화(init)
- (4) 파일 등록(add)
- (5) Git 로컬 저장소에 업로드(commit)
- (6) Git을 편리하게 사용하는 방법
 - Git 저장소에 업로드한 내용과 새로 업데이트 된 변경 사항 확인(diff)
 - commit 간소화(commit -m)
- (7) Github 회원가입
- (8) Github 원격 저장소 만들기
- (9) Git 로컬 저장소에서 Github 원격 저장소에 저장(push)

Step1- 매뉴얼 보는 방법과 명령어 구조

■ 매뉴얼 보는 방법

(1) Git-bash 혹은 터미널 실행

```
$ git commit --help
```


```
$ git add --help
```


■ Git 명령어 구조

```
$ git add report-card.txt
```


```
$ git config --global user.name "OSS"
```

```
$ git commit -m "commit1"
```

 Git command

 Sub command

 Argument(인자)

 Option
 -- Long name
 - Short name

Step2- 브랜치 실습 사전준비

- branch-test 폴더 만들고 branch.txt 라는 파일 만들자

(1) 해당 폴더에 들어가서 git 초기화 하기

```
$ cd branch-test; git init
```

```
$ vim myfile.txt
```

- myfile.txt 열어서 (아래 간단한 글(만) 작성 해보자)

- 쉽게 배우는 Git 명령어

(2) 해당 파일 첫 커밋하기

```
$ git add myfile.txt; git commit -m "first commit"
```



Step2- 브랜치 만들기

- **issue1** 이라는 이름으로 새로운 브랜치를 만들자
브랜치는 `branch` 란 명령어로 만들 수 있다.

(1) 새로운 브랜치 만들기

```
$ git branch issue1
```

- 옵션을 지정하지 않고 `branch` 명령어 실행하면 브랜치 목록 전체 확인 할 수 있다. 앞 부분에 *이 붙어있는 것이 현재 선택된 브랜치다.

(2) 브랜치 확인하기 (현재는 master)

```
$ git branch
```

```
issue1  
*master
```



Step2- 브랜치 전환하기

- 앞에서 만든 **issue1** 이라는 이름의 브랜치를 사용하여 어떤 작업을 수행하려면 이 브랜치를 사용 하겠다고 명시적으로 지정 해야함.
- 이 때 사용하는 명령어가 checkout이다. 체크아웃(checkout)이란 내가 사용할 브랜치를 지정하는 것을 의미한다.

```
$ git checkout issue1
```

```
$ git branch
```

```
* issue1  
master
```

Step2- 브랜치 만들고 전환하기

- checkout 명령에 -b 옵션을 넣으면 브랜치 작성과 체크아웃 한꺼번에 실행할 수 있다.

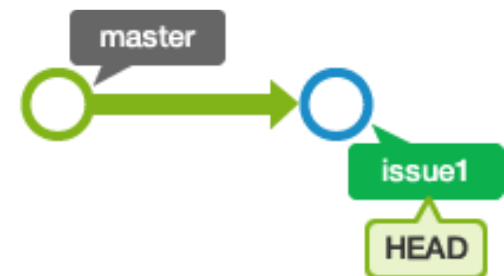
```
$ git checkout -b <branch>
```

- **issue1** 브랜치를 체크아웃한 상태에서 커밋을 수행하면 **issue1** 브랜치에 그 이력이 기록된다.

- myfile.txt에 아래와 같은 문장을 추가한 후 커밋
– 쉽게 배우는 Git 명령어

add: 변경 사항을 만들어 인덱스 등록하기

```
$ git add myfile.txt; git commit -m "add message"
```



Step2- 브랜치 병합하기(1)

- 이번에는 **issue1**의 브랜치 변경사항을 **master** 브랜치에 병합해보자.
- 브랜치 병합은 merge 명령어로 실행한다.
- 이 명령어에 병합할 커밋 이름을 넣어 실행하면 지정한 커밋 내용이 'HEAD'가 가리키고 있는 브랜치에 넣어진다.
'HEAD'는 현재 사용중인 브랜치에 위치함.
- master 브랜치에 **issue1**을 넣기 위해서는 우선 **master** 브랜치에 'HEAD'가 위치하게 만들어야함. 이 때 checkout 명령어 사용하여 현재 사용중인 브랜치를 **master**로 전환하자.

```
$ git checkout master
```

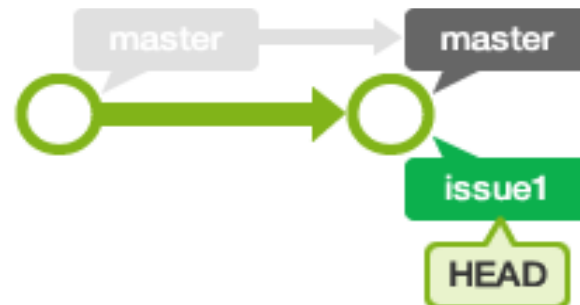
*병합 전 myfile.txt 파일을 열어 내용을 확인해보자.

Step2- 브랜치 병합하기(2)

- 이제 이번 실습에서의 파일 편집은 **issue1** 브랜치에서 실행했기 때문에 **master** 브랜치로 브랜치를 전환한 지금 myfile.txt 파일을 열었을때 그 내용이 **변경 되어 있지 않아야 한다.**

```
$ git merge issue1
```

- 이제 **master** 브랜치가 가리키는 커밋이 **issue1**과 같은 위치로 이동했다. 이런 방식의 병합을 '**fast-forward 병합**' 이라고 한다.



Step2- 브랜치 병합하기(3)

- myfile.txt 파일 열어 내용 확인해보자.
 - 쉽게 배우는 Git 명령어
add: 변경 사항을 만들어 인덱스 등록하기
- “add: 변경 사항을 만들어 인덱스 등록하기”
내용이 **추가 되어 있는 것**을 확인 할 수 있다.

Step2- 브랜치 삭제하기

- **issue1** 브랜치의 내용이 모두 **master**에 통합되어 더 이상 **issue1** 브랜치는 필요가 없다.
- 브랜치 삭제하려면 branch 명령에 **-d** 옵션 지정하여 실행한다.

```
$ git branch -d <branchname>
```

- **issue1** 브랜치를 삭제한다.

```
$ git branch -d issue1
```

- 제대로 삭제되었는지 확인해보자.

```
$ git branch
```

```
*master
```



Step2- 동시에 여러 브랜치 작업하기(1)

- 이번에는 2개의 브랜치 생성하여 동시에 여러 작업을 처리하는 상황을 만들어보자.
- **issue2** 와 **issue3** 브랜치 만들자

```
$ git branch issue2
```

```
$ git branch issue3
```

- **issue2** 브랜치로 전환해보자

```
$ git checkout issue2
```

```
$ git branch
```

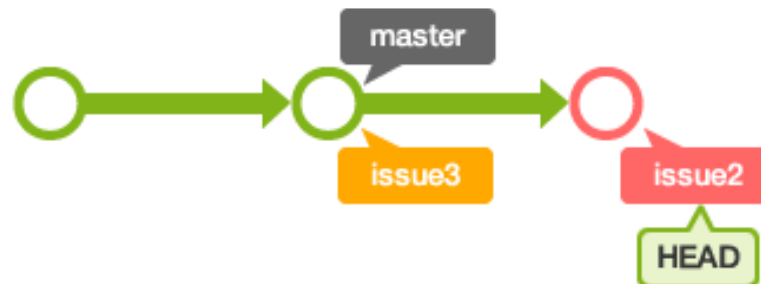
```
*issue2  
issue3  
master
```



Step2- 동시에 여러 브랜치 작업하기(2)

- **issue2** 브랜치의 myfile.txt에 아래와 같이 commit에 대한 설명을 추가
 - 쉽게 배우는 Git 명령어
 - add: 변경 사항을 만들어 인덱스 등록하기
 - commit: 인덱스 상태를 기록하기**
- 문장을 추가했으면 커밋해보자.

```
$ git add myfile.txt; git commit -m "commit explain plus"
```



Step2- 동시에 여러 브랜치 작업하기(3)

- 이번에는 **issue3** 브랜치로 전환하자.

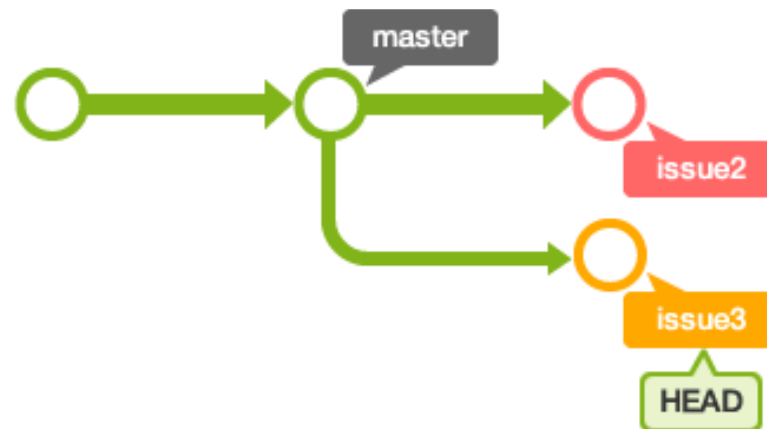
```
$ git checkout issue3
```

- myfile.txt 파일 열어 내용 확인해보자.
- commit 명령의 설명은 **issue2** 브랜치에서 추가했기 때문에 **issue3** 브랜치로 전환한 후의 myfile.txt 파일에는 add 명령의 설명밖에 없을 것이다.

Step2- 동시에 여러 브랜치 작업하기(4)

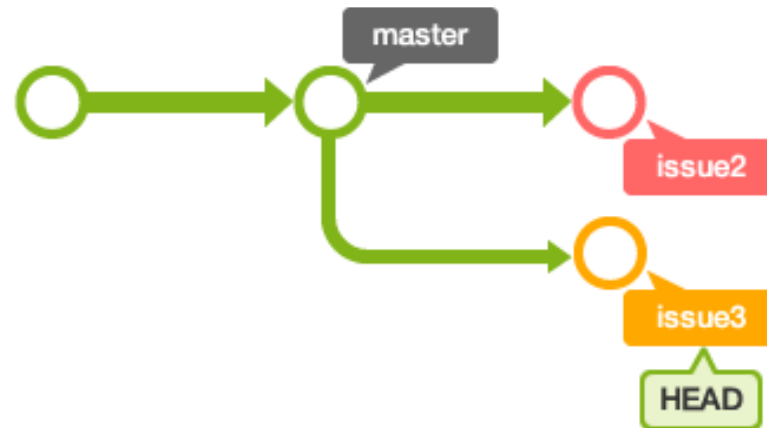
- 이번에는 pull 명령어의 설명을 추가하여 변경 사항을 커밋해 보자.
 - 쉽게 배우는 Git 명령어
 - add: 변경 사항을 만들어 인덱스 등록하기
 - pull: 원격 저장소의 내용을 가져오기
- 문장을 추가했으면 커밋해보자.

```
$ git add myfile.txt; git commit -m "pull explain plus"
```



Step2- 동시에 여러 브랜치 작업하기(5)

- 지금까지 **issue2** 브랜치에 commit에 대한 설명을
- **issue3** 브랜치에 pull에 대한 설명을 포함하여 커밋해 보았다.
- 이처럼 각각의 브랜치에서는 독립적으로 서로 다른 작업을 처리할 수 있다.



Step2- 충돌 해결하기(1)

- **issue2**, **issue3** 각 브랜치에서 변경한 부분 모두 master 브랜치로 통합해보자.
- 먼저 master 브랜치를 체크아웃하자.

```
$ git checkout master
```

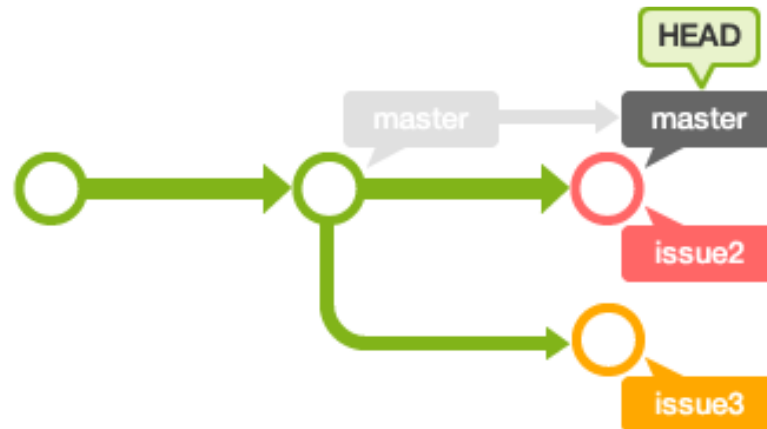
- **issue2** 브랜치를 병합한다.

```
$ git merge issue2
```

*위와 같이 하면 앞서 설명되었던 'fast-forward 병합'이 실행된다.

Step2- 충돌 해결하기(2)

- 아래 그림을 보면 master 브랜치에 **issue2** 브랜치가 병합된 것을 확인할 수 있다.



- 이번에는 **issue3** 브랜치를 병합한다.

```
$ git merge issue3
```

Auto-merging myfile.txt

CONFLICT (content): Merge conflict in myfile.txt

Automatic merge failed; fix conflicts and then commit the result.

Step2- 충돌 해결하기(3)

- conflict 발생에서 알 수 있듯이 자동 병합에 실패했다.
- 앞서 각각의 브랜치에서 변경한 내용이 myfile.txt의 **같은 행**에 포함되어 있기 때문이다.
- 실제로 myfile.txt의 내용을 확인해 보면 다음과 같이 변경되어 있다.

쉽게 배우는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

<<<<<<HEAD

commit: 인덱스 상태를 기록하기

=====

pull: 원격 저장소의 내용을 가져오기

>>>>>>issue3

Step2- 충돌 해결하기(4)

- 충돌이 있는 부분에 Git이 자동으로 이전 페이지와 같이 충돌 정보를 포함하여 파일 내용을 변경한다.
- 이 내용을 통해 어떤 브랜치에서 어떤 부분이 충돌 되었는지 확인할 수 있다.
- 충돌이 일어난 부분은 일일이 확인해서 수정해 줘야 한다.

- 쉽게 배우는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

commit: 인덱스 상태를 기록하기

pull: 원격 저장소의 내용을 가져오기

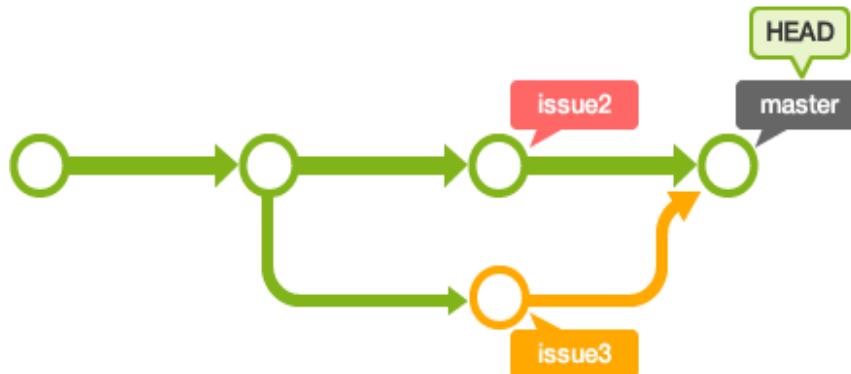
Step2- 충돌 해결하기(5)

- 충돌 부분을 모두 수정했으므로 다시 커밋해보자.

```
$ git add myfile.txt
```

```
$ git commit -m "issue3 branch merge"
```

- 이 시점까지의 이력을 보면 그림과 같다. 이번 병합은 충돌 부분을 수정했기 때문에 그 변화를 기록하는 병합 커밋이 새로 생성되었다.
- 그리고 master 브랜치의 시작('HEAD')이 그 위치로 이동해 있는 것을 확인 할 수 있다. 이와 같은 방식을 'non fast-forward 병합'이라고 한다.



Step3- Github 원격저장소 활용 방법

■ 현재 참여하고 있는 프로젝트 동기화

- (1) 원격저장소에서 PC로 동기화 하는 방법
(원격저장소와 PC가 처음 동기화 할 때)

```
$ git clone 원격저장소 링크
```

- (2) 기존에 등록된 원격저장소에서 PC로 동기화 하는 방법
(단 원격저장소 링크가 등록된 경우에만 가능)

```
$ git pull
```

- (3) PC에서 원격저장소로 동기화 하는 방법

```
$ git push
```

Step3- Github 원격저장소 활용 방법

■ 다른 오픈소스 프로젝트 살펴보기

- (1) 관심있는 프로젝트 원격저장소 방문 하여 fork 하기
- (2) 내 저장소로 fork해온 후 clone 하여 로컬 저장소(PC)로 가져오기

```
$ git clone 원격저장소링크 생성파일이름
```

- (3) 원격저장소에서 받아온 파일 확인하기

```
$ ls -al
```

Step3- Github 원격저장소 활용 방법

■ 다른 오픈소스 프로젝트에 기여하기(1)

(1) 관심있는 프로젝트 원격저장소 방문 하여 fork 하기

(2) 내 저장소로 fork해온 후 clone 하여 로컬 저장소(PC)로 가져오기

```
$ git clone 원격저장소링크 생성파일이름
```

(3) 토픽 브랜치로 변경하기

```
$ git checkout -b 토픽브랜치이름
```

(4) 코드에 기여하기(리팩토링, 디버그, 기능 추가 등등)

```
$ git add 변경파일; git commit -m 변경사항
```

(5) fork 해온 내 저장소에 push하기

```
$ git push
```

(6) 나의 원격저장소로 이동

Step3- Github 원격저장소 활용 방법

■ 다른 오픈소스 프로젝트에 기여하기(2)

★나의 프로필에서 fork해서 만들어진 프로젝트 페이지로 이동

enouo / git--training
forked from skkusosc/git--training

Unwatch 1 Star 0 Fork 1

Code Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

No description, website, or topics provided. [Add topics](#) [Edit](#)

1 commit **2 branches** 0 releases 1 contributor

Your recently pushed branches:

develop (less than a minute ago) [Compare & pull request](#)

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is even with skkusosc:master. [Pull request](#) [Compare](#)

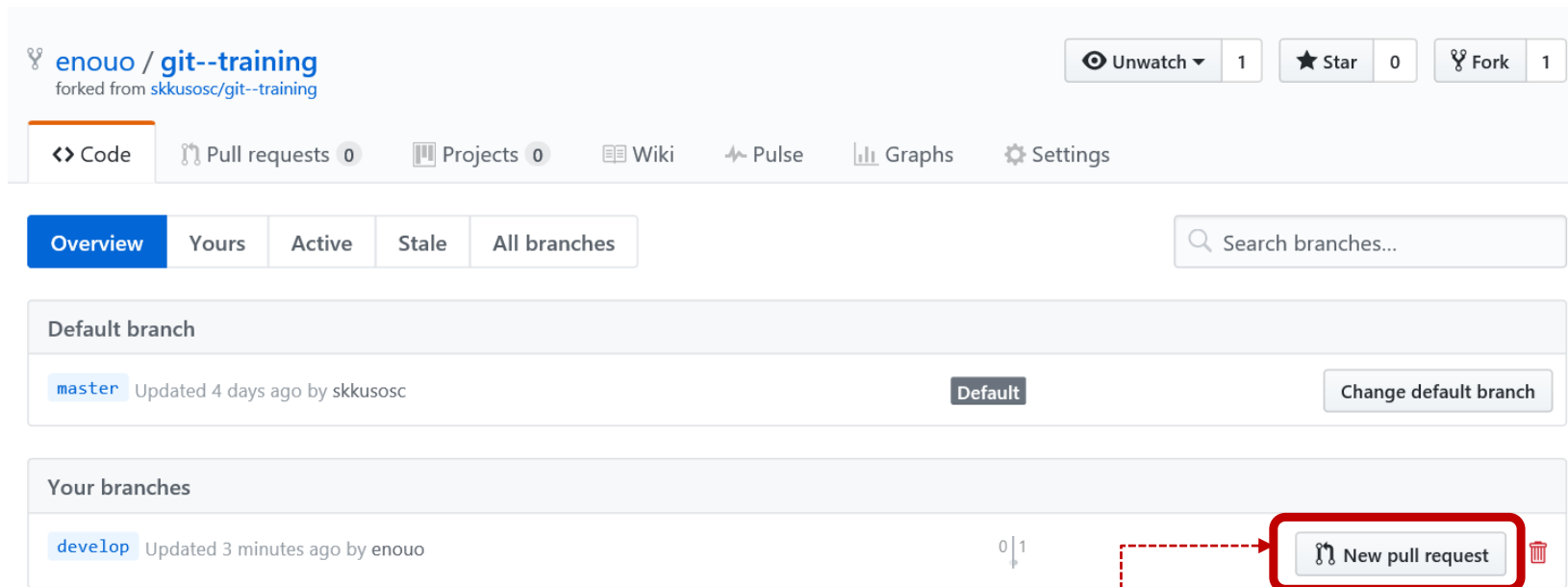
skkusosc git-training Latest commit 07674ff 4 days ago

packing_knapsack git-training 4 days ago

방법1-1) 방금 push 했던 브랜치를 확인하기 위해서 Branch 탭 클릭

Step3- Github 원격저장소 활용 방법

■ 다른 오픈소스 프로젝트에 기여하기(3)



방법1-2) Pull-request하려는 브랜치에서 New pull-request 버튼 클릭

Step3- Github 원격저장소 활용 방법

▪ 다른 오픈소스 프로젝트에 기여하기(4)

skkusosc / git--training

Watch 1 Star 0 Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base fork: skkusosc/git--training base: master ... head fork: enouo/git--training compare: develop

✓ Able to merge. These branches can be automatically merged.

test pull request #1
Signed-off-by: enouo skyup706@gmail.com

View pull request

test pull request

Write Preview

Signed-off-by: enouo <sosc@gmail.com>
오픈소스SW실습 이름:

Attach files by dragging & dropping or [selecting them](#).

☒ Allow edits from maintainers. [Learn more](#)

Create pull request

Pull-request 보내기

* pull-request 의 단위는 branch

Step3- Github 원격저장소 활용 방법

■ 다른 오픈소스 프로젝트에 기여하기(5)

*skkusc/git--training(본래 프로젝트)에서 만들어진 pull-request 확인하기
*주의) 나의 프로필에서 fork해서 만들어진 프로젝트 페이지에서 확인 하는게 아님

Contents

I
Git/GitHub
Intro

II
Hands-on
(Basic)

III
Hands-on
(Advanced)

IV
팀프로젝트
준비

실습

- 실습 원격저장소를 Fork 해온 다음 자신의 파일 추가하여 pull-request 해보기
- 실습 원격저장소 주소 : <https://github.com/soscskku/OSS>
- 본인 이름(영문)으로 txt 파일을 만들어 내용 안에 학번 및 이름 적고
실습 원격저장소에 pull request
- 아이캠퍼스 과제는 안 만들 예정이며 pull-request의 내역을 확인하여
제출 여부 확인 예정
※ 그러므로 본인 이름(영문).txt 파일 안에 학번 및 이름 기재 확실히 할 것

Contents

I
Git/GitHub
Intro

II
Hands-on
(Basic)

III
Hands-on
(Advanced)

IV
팀프로젝트
준비

팀프로젝트- 사전 준비(1)

- 각 조 팀장은 “20-1-SKKU-OSS 조직”(Organization)”에 합류하자
- 합류한 팀장은 ‘New team’을 생성하자



<팀장 역할>



Flexible repository access

You can add repositories to your teams with more flexible levels of access (Admin, Write, Read).



Request to join teams

Members can quickly request to join any team. An owner or team maintainer can approve the request.



Team mentions

Use team @mentions (ex. @github/design for the entire team) in any comment, issue, or pull request.

[Learn more](#)

🔍 Find a team...

New team

팀프로젝트- 사전 준비(2)

- Team name은 각 팀별로 작성하자
- 팀명 : 2020(1)OSS-(1,2..)
- Team visibility는 Visible 선택하고 생성하자

Create new team

Team name

test



Mention this team in conversations as @18-1-SKKU-OSS/test.

Description

What is this team all about?

Parent team

Select parent team

Team visibility

☒ Visible Recommended

A visible team can be seen and @mentioned by every member of this organization.

☐ Secret

A secret team can only be seen by its members.

Create team



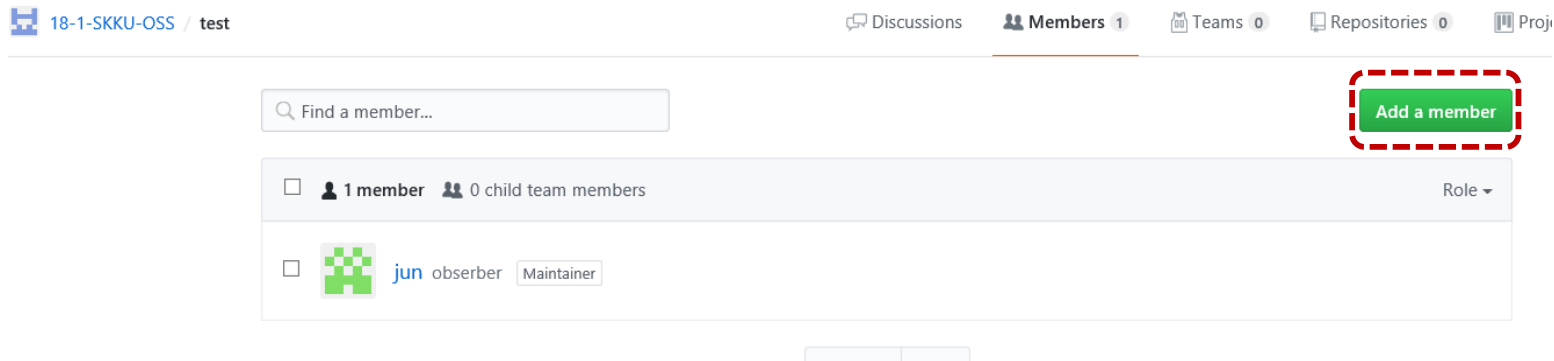
<팀장 역할>

팀프로젝트- 사전 준비(3)



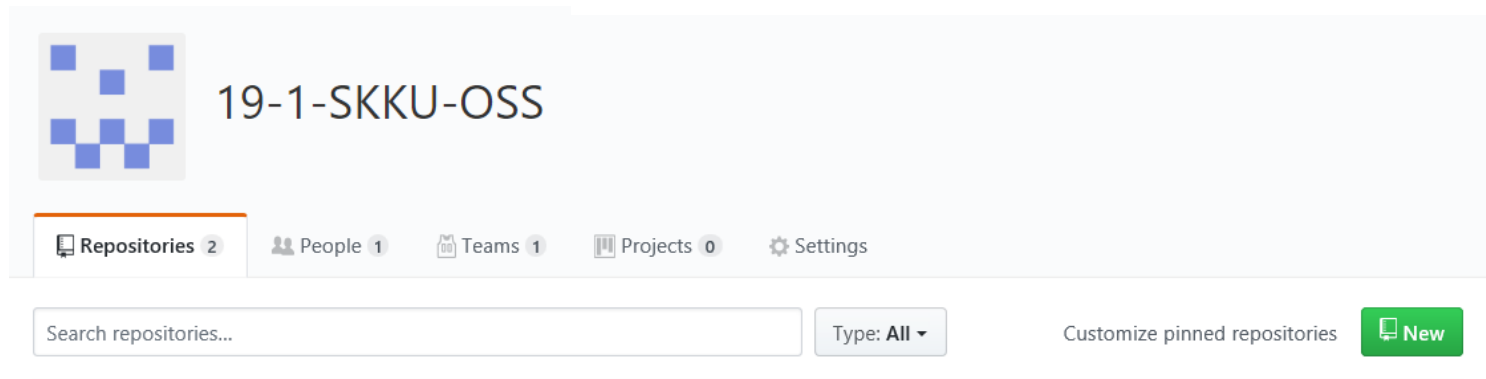
<팀장 역할>

- 이제 해당 팀으로 팀원들을 초대하자



팀프로젝트- 사전 준비(4)

- 우리 팀 Repository 를 생성하자



<팀장 역할>

팀프로젝트- 사전 준비(5)

- 우선 ‘우리팀(명)’ 으로 된 **Repository**를 생성해보자
- Repository 명은 추후 설정에서 변경 가능하다

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

18-1-SKKU-OSS ▼

Repository name

/ test

Great repository names are short and memorable. Need inspiration? How about [legendary-rotary-phone](#).

Description (optional)

☒ **Public**

Anyone can see this repository. You choose who can commit.

☐ **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▼

Add a license: **None** ▼



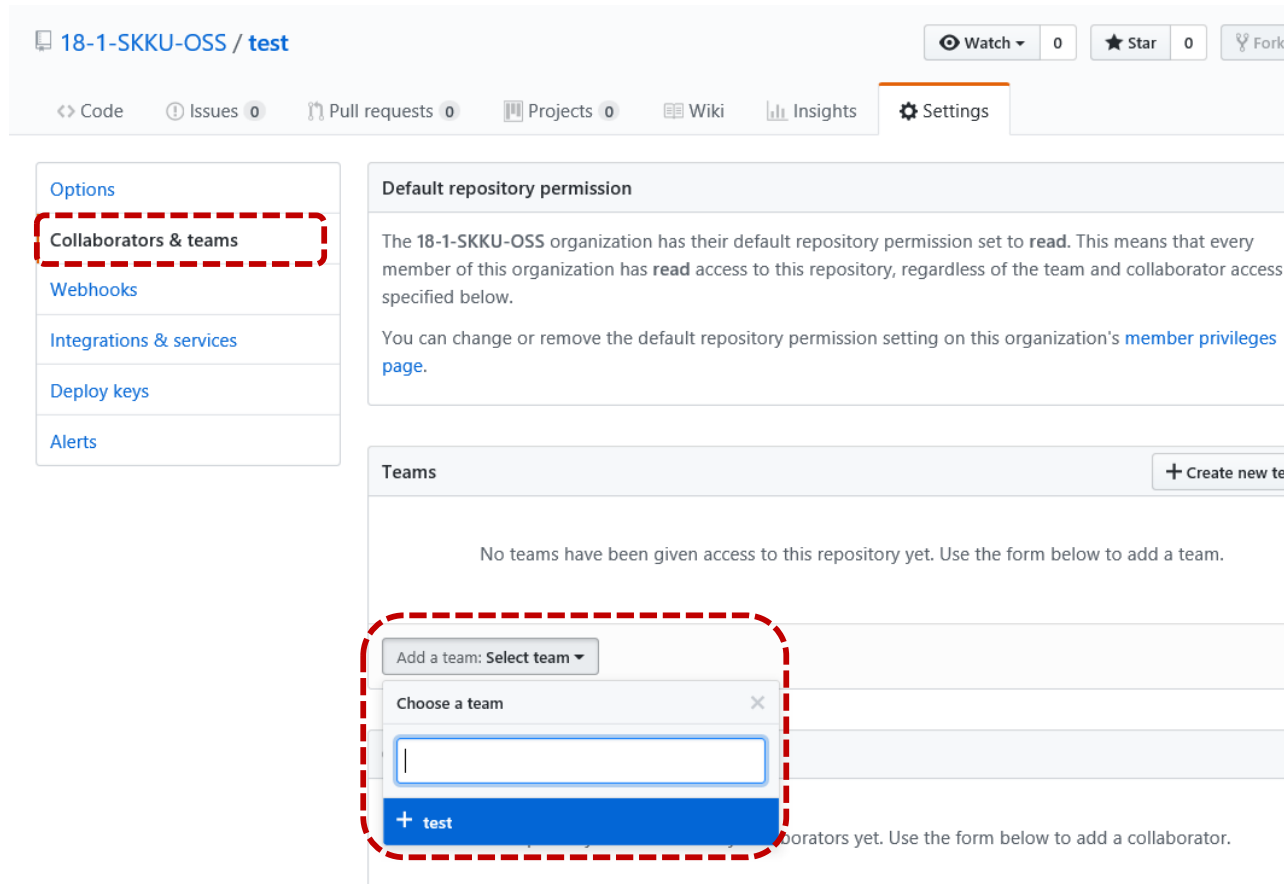
Create repository



<팀장 역할>

팀프로젝트- 사전 준비(6)

- Collaborators&teams-> Choose a team: (우리팀명) 선택



18-1-SKKU-OSS / test

Watch 0 Star 0 Fork

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Options

Collaborators & teams

Webhooks

Integrations & services

Deploy keys

Alerts

Default repository permission

The 18-1-SKKU-OSS organization has their default repository permission set to **read**. This means that every member of this organization has **read** access to this repository, regardless of the team and collaborator access specified below.

You can change or remove the default repository permission setting on this organization's [member privileges page](#).

Teams + Create new team

No teams have been given access to this repository yet. Use the form below to add a team.

Add a team: Select team

Choose a team

+ test



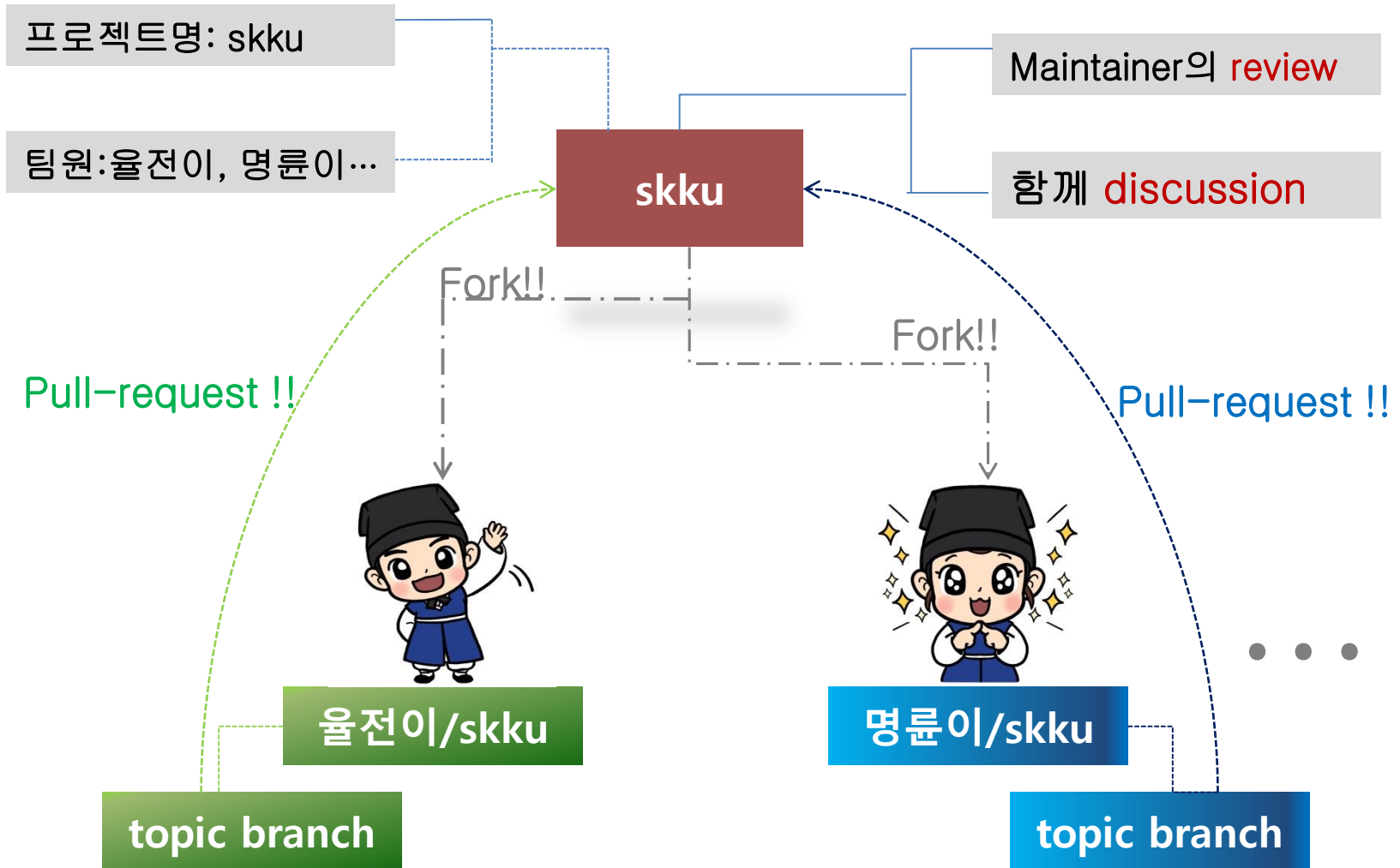
<팀장 역할>

팀프로젝트—사전 준비(7)

- 이번 학기 실습은 GitHub Organization -> Team에서 활동을 해보자
- GitHub의 다양한 서비스 기능들을 팀원들과 적극적으로 사용해보자



우리 프로젝트와 Git 운용 전략



참여와 공유

