



Git / GitHub

(1주차)

Contents

I
Git/GitHub
Intro

II
Hands-on
(Basic)

III
Hands-on
(Advanced)

IV
Git/GitHub
Review

Contents

I
Git/GitHub
Intro

II
Hands-on
(Basic)

III
Hands-on
(Advanced)

IV
Git/GitHub
Review

오픈소스PJT 기여 과정



설치&사용

Pull-request
보내기

- 오타수정
- 단순 디버깅
- Etc.

Project
개선

- New feature
- Refactoring
- Etc.

Git/GitHub Start

MongoDB, NoSQL, SQL, Linux, Java.. 등 무수히 많은 **프로그램**
무수히 많은 **프로젝트**

어디서부터 시작할지 **고민하지 말고**

우선 **Git/Github** 부터 알아보자



What is Git?

- Formal **version control system**
- Developed by Linus Torvalds(developer of Linux)
 - used to manage the source code for Linux
- Tracks any content(but mostly plain text files)
 - source code
 - data analysis projects
 - websites
 - presnetations



Why use Git?

- It's **fast**
- You don't need access to a server
- Amazingly good data merging simultaneous changes
- **Everyone's** using it



- ❖ Git is fast, you can use it locally on your own computer, it's amazingly good at merging changes, and there are lots of people using it.

What is GitHub?

- GitHub.com is a site for **online storage** of Git repositories.
- **Many open source projects** use it,
 - such as the Linux kernel.
- You can get **free space** for open source projects or you can pay for private projects.



Why use GitHub?

- It takes care of the server aspects of Git
- Graphical user interface for git
 - Exploring code and its history
 - Tracking issues
- Facilitates:
 - Learning from others
 - Seeing what people are up to
 - Contributing to others code



- ❖ GitHub is great for browsing others code, for learning; you don't even have to download it to your computer. And it's really easy to contribute to others' code

GitHub



Contents

I
Git/GitHub
Intro

II
**Hands-on
(Basic)**

III
Hands-on
(Advanced)

IV
Git/GitHub
Review

Git/GitHub 실습 본격적인 시작에 앞서서..

- 본 강의는 실습 90% 이론 10%
- 나중에 하는건 없다. 무조건 오늘 목표는 이루고 가자
- 내 손으로 명령어를 직접 입력해보자
- 내 옆에 숨은 고수들이 많다 (조장 및 조교들에게 적극 도움을 요청하자)
- 처음 다루는 팀원들을 도와주며 친해지자 (팀 프로젝트 진행 사전 대비)
- 터미널 환경 경험하자
- GUI(Graphic User Interface)대신 CLI(Command Line Interface)경험하자
- Vi에디터를 사용해보자

훈련방식(Training mode)

우리가 스마트폰을(매뉴얼 없이) 사용하면서 이해하듯

Git 이란 프로그램도 일단 써보면서 이해 해보자



실습 방법, 시나리오

Git 실습 방법은 ??

직접 C프로그래밍을 짜면서 **Git을 사용한다는 시나리오**

→ report card(성적 출력하기 문제)

- 1) 코딩은 Ctrl-c,v로 하고 (예제 파일 활용)
- 2) Git은 **직접 명령어** 쳐서 (git-bash 사용)

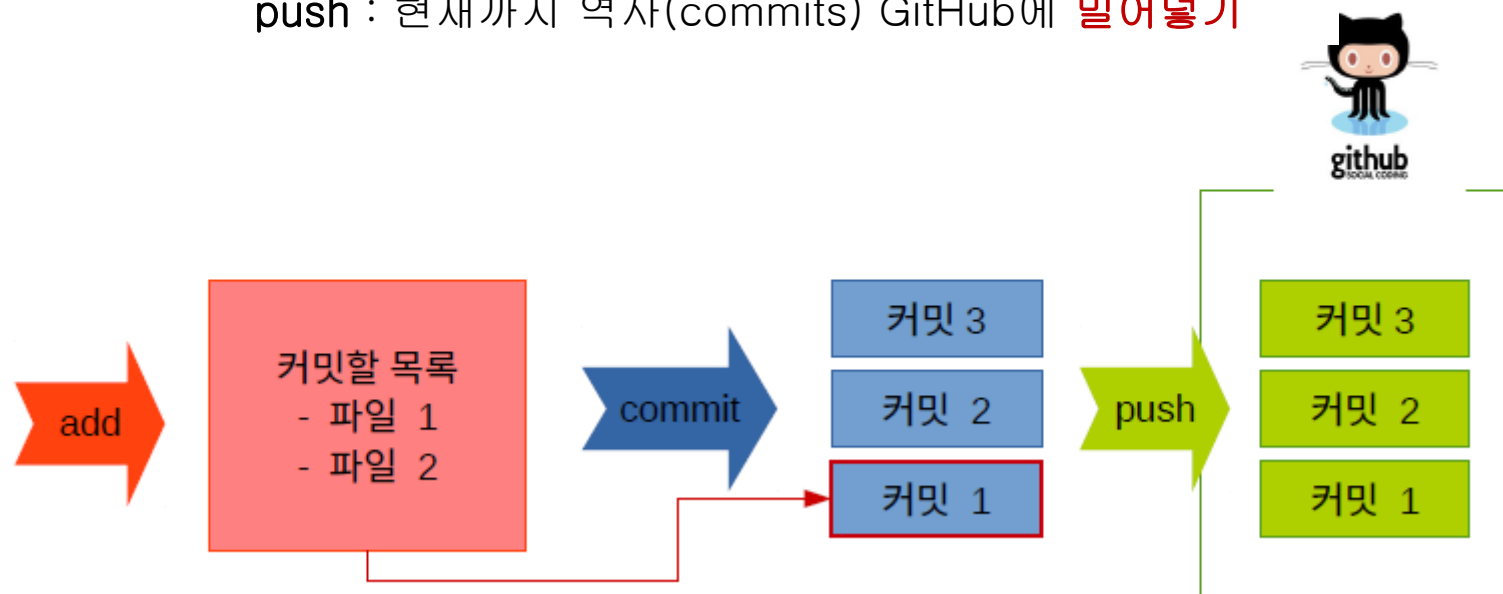


Git 필수 명령

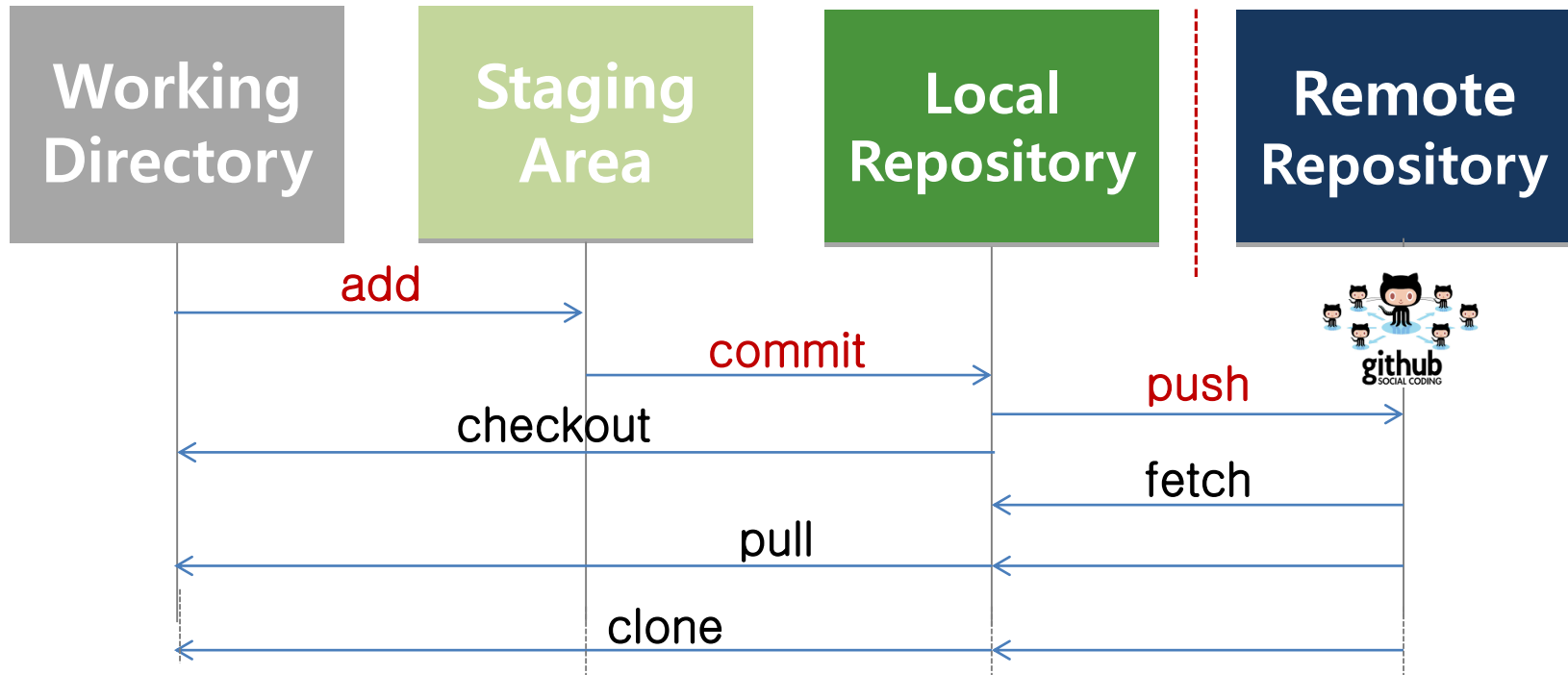
add : 커밋할 **목록에 추가**

commit : 커밋 (히스토리의 **한단위**) **만들기**

push : 현재까지 역사(commits) GitHub에 **밀어넣기**



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**

준비하기

1. 예제소스 다운받기

아이캠퍼스 - 자료실

2. Git 설치하기

<https://git-scm.com/downloads>

Downloads



Older releases are available and the Git source repository is on GitHub.



3. Editor 다운받기 (자유)


<https://atom.io/>

4. GitHub 회원가입

<https://github.com/join>


준비하기

5. example-ex.zip 압축 풀고 폴더 열어두기



Ctrl-c,v 복사, 붙여넣기 할 소스들

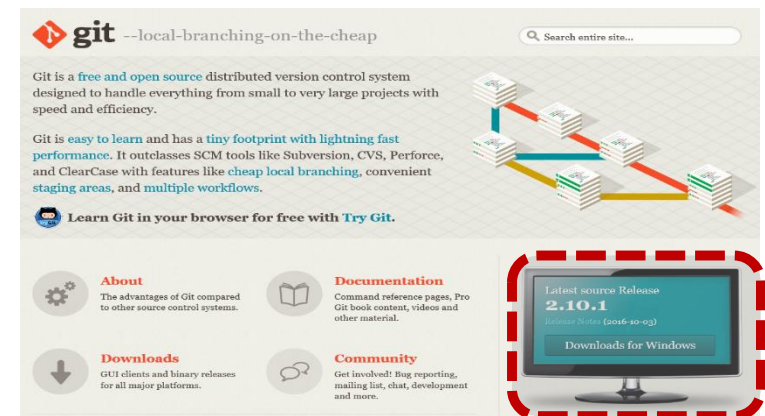
6. 편집기 (atom or sublime text) 열어두기




Ctrl-c,v 복사, 붙여넣기 용도 편집기

Git 설치

- 다운로드: <https://git-scm.com/download>
- 리눅스의 경우 배포판에 따라
 - 데비안 계열 : `$ sudo apt-get install git-all`
 - Fedora 배포판 : `$ sudo yum install git-all`
- GUI: git-gui, GitKraKen, Source Tree




GitHub 회원가입



[Personal](#)
[Open source](#)
[Business](#)
[Explore](#)
[Pricing](#)
[Blog](#)
[Support](#)

Join GitHub


The best way to design, build, and ship software.



Step 1:
Set up a personal account



Step 2:
Choose your plan



Step 3:
Tailor your experience

Create your personal account

Username

This will be your username — you can enter your organization's username next.

Email Address

You will occasionally receive account related emails. We promise not to share your email with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

You'll love GitHub

Unlimited collaborators

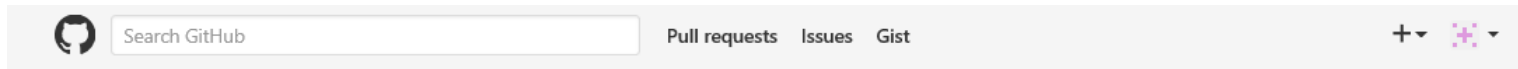
Unlimited public repositories

- ✓ Great communication
- ✓ Frictionless development
- ✓ Open source community

가입정보 기입



GitHub 회원가입



Welcome to GitHub

You've taken your first step into a larger world, @skkusosc

Completed
Set up a personal account

Step 2:
Choose your plan

Step 3:
Tailor your experience

free 선택하자

Choose your personal plan

- ☒ Unlimited public repositories for free.
- ☐ Unlimited private repositories for \$7/month. ([view in KRW](#))

Don't worry, you can cancel or upgrade at any time.

- ☐ **Help me set up an organization next**
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
[Learn more about organizations.](#)

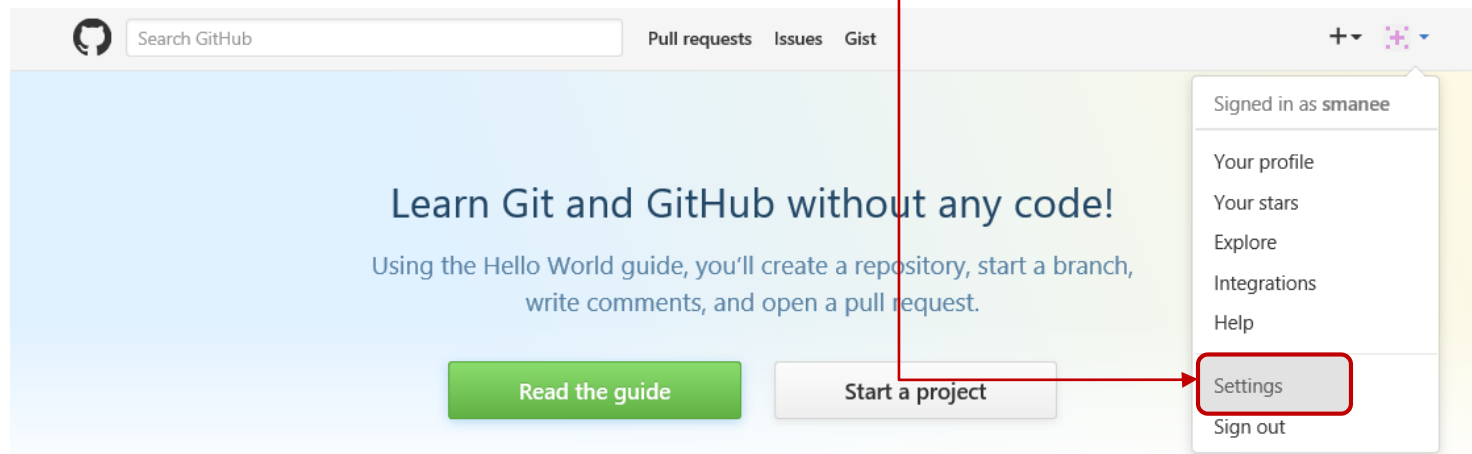
Continue

Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

GitHub 메일 인증하기

기다려도 메일이 혹시 안오면 설정버튼



Step 기본 설정

1) Git-bash 혹은 터미널 실행 후에

1-1) 미리 캐치 저장되어 있을지 모를 계정정보 삭제 (처음 설치시 생략 가능)

```
$ git config --global user.email --unset-all user.name
```

```
$ git config --global user.name --unset-all user.email
```

2) 나의 GitHub계정 이메일 (GitHub계정 이메일)과 이름 (본인 영문이름or닉네임) 을 적자

```
$ git config --global user.email "본인메일@gmail.com"
```

```
$ git config --global user.name "본인이름or닉네임 skkusosc"
```

※ git-training-ex-v2.zip 압축분 폴더 열기 (window 폴더 탐색기로)

Step1 초기화 및 첫 commit하기 (Basic)

1) 나의 GitHub 계정 이메일과 이름을 적자 (기본 설정)

```
$ git config --global user.email "본인메일@gmail.com"
```

```
$ git config --global user.name "본인이름,닉네임 skkusosc"
```

2) Git bash를 실행 (명령어칠 준비), 폴더 생성하기

```
$ mkdir report-card
```

3) 경로 이동 (pwd 명령어로 현재경로 확인하기)

```
$ cd report-card
```

4) 해당 폴더를 git 초기화 (ls-A 명령어로 생성된 .git폴더 확인하기)

```
$ git init
```

5) 프로그램 문제 PDF 파일 추가 (커밋할 목록에 추가 add) (commit1 폴더내 파일 활용)

```
$ git add report_card.pdf
```

6) 첫 commit 하기 (역사 한단위 만들기)

```
$ git commit -m "report card: Add question PDF"
```

Step1 초기화 및 첫 commit하기 (Basic)

일단 따라해보자

7) 소스코드 추가하기 (커밋할 목록에 추가 add) (commit2 폴더내 파일 활용)

```
$ git add report_card.c
```

8) commit 하기 (역사 한단위 만들기)

```
$ git commit -m "report card: Add base code"
```

Git 상태확인 명령어
(중간중간 치면서 수시로 확인하자)

```
$ git show  
$ git log  
$ git shortlog  
$ git diff  
$ git status
```

Step2 diff 사용과 추가 commit하기 (Basic)

일단 따라해보자

1) 상태를 확인한다

```
$ git status
```

2) **commit3** 폴더에 있는 report_card.c 소스 파일로 수정, 덮어쓰기 후 확인

```
$ git diff
```

3) diff를 통해서 변화분을 확인했다면 add 진행

```
$ git add report_card.c
```

4) 준비된 소스파일을 commit 한다

```
$ git commit -m "report card: Print a message of introduction"
```

5) 지금까지한 3개의 commit들을 확인해보자

```
$ git log
```

Step2 diff 사용과 추가 commit하기 (Basic)

일단 따라해보자

6) **commit4** 폴더에 있는 report_card.c 소스 파일로 수정, 덮어쓰기 후 확인

```
$ git diff
```

7) diff 를 통해서 변화분을 확인했다면 add 진행

```
$ git add report_card.c
```

8) 준비된 소스파일을 commit 한다.

```
$ git commit -m "report card: Print grades of each subject"
```

9) 지금까지한 4개의 commit들을 확인해보자

```
$ git log
```

Step3 commit하기,반복 (Basic)

일단 따라해보자

1) **commit5 폴더**에 있는 report_card.c 소스 파일로 수정, 덮어쓰기 후 확인

```
$ git diff
```

2) diff 를 통해서 변화분을 확인했다면 add 진행

```
$ git add report_card.c
```

3) 준비된 소스파일을 commit 한다.

```
$ git commit -m "report card: Show the sum of each grade"
```

Step3 commit하기,반복 (Basic)

일단 따라해보자

4) **commit6 폴더**에 있는 report_card.c 소스 파일로 수정, 덮어쓰기 후 확인

```
$ git diff
```

5) diff 를 통해서 변화분을 확인했다면 add 진행

```
$ git add report_card.c
```

6) 준비된 소스파일을 commit 한다

```
$ git commit -m "report card: Get a average of grades"
```

Step4 지금까지의 commit을 push 하자 (Basic)

일단 따라해보자

1) 상태를 확인하고 현재 브랜치명 master 를 확인하자

```
$ git status
```

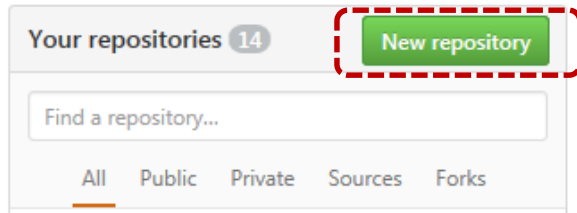
2) 지금까지한 commit들을 확인하자 (6개가 아니면 다시 확인하자)

```
$ git shortlog
```

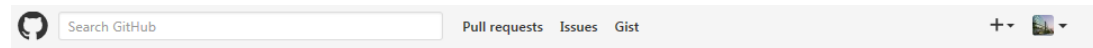
3) Github 원격저장소 URL을 등록하자

(잠깐 멈추고 <http://github.com> 켜고 repository 새로 생성하자)

잠깐, GitHub에서 원격저장소 만들기



새로운 **원격 저장소**를 생성하자



Create a new repository

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

프로젝트명 자유(ex. test)

Owner:

Repository name:

Great repository names are short and memorable. Need inspiration? How about **cuddly-tribble**.

Description (optional):

☒ **Public**
Anyone can see this repository. You choose who can commit.

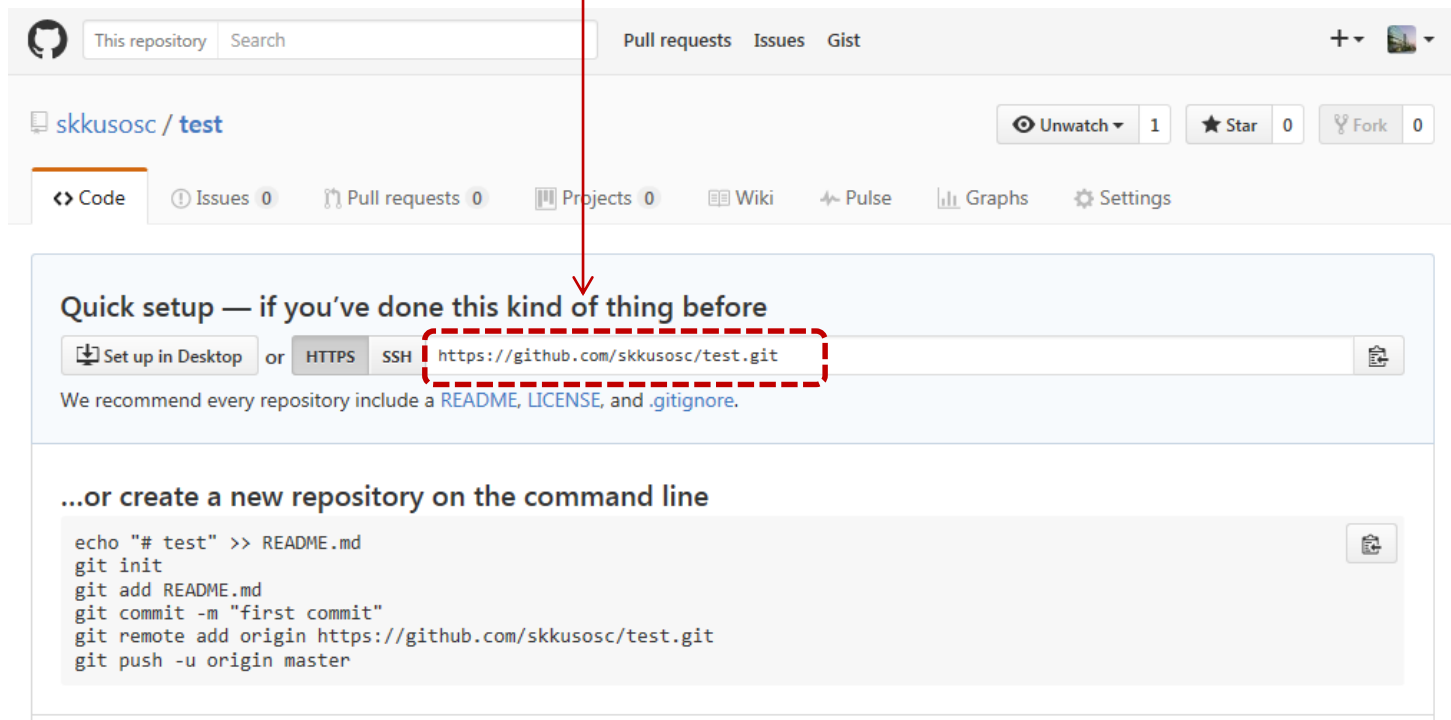
☐ **Private**
You choose who can see and commit to this repository.

☐ **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** ⓘ

잠깐, GitHub에서 원격저장소 만들기

해당 URL 복사해서 Step4의 4)으로 이어서 진행



Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/skkusosc/test.git` 

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/skkusosc/test.git
git push -u origin master
```

Step4 지금까지의 commit을 push 하자 (Basic)

일단 따라해보자

4) 방금 복사한 URL로 GitHub 원격저장소 등록하자

```
$ git remote add origin 방금복사한 URL
```

5) GitHub 원격저장소(origin)에다가 밀어 넣자

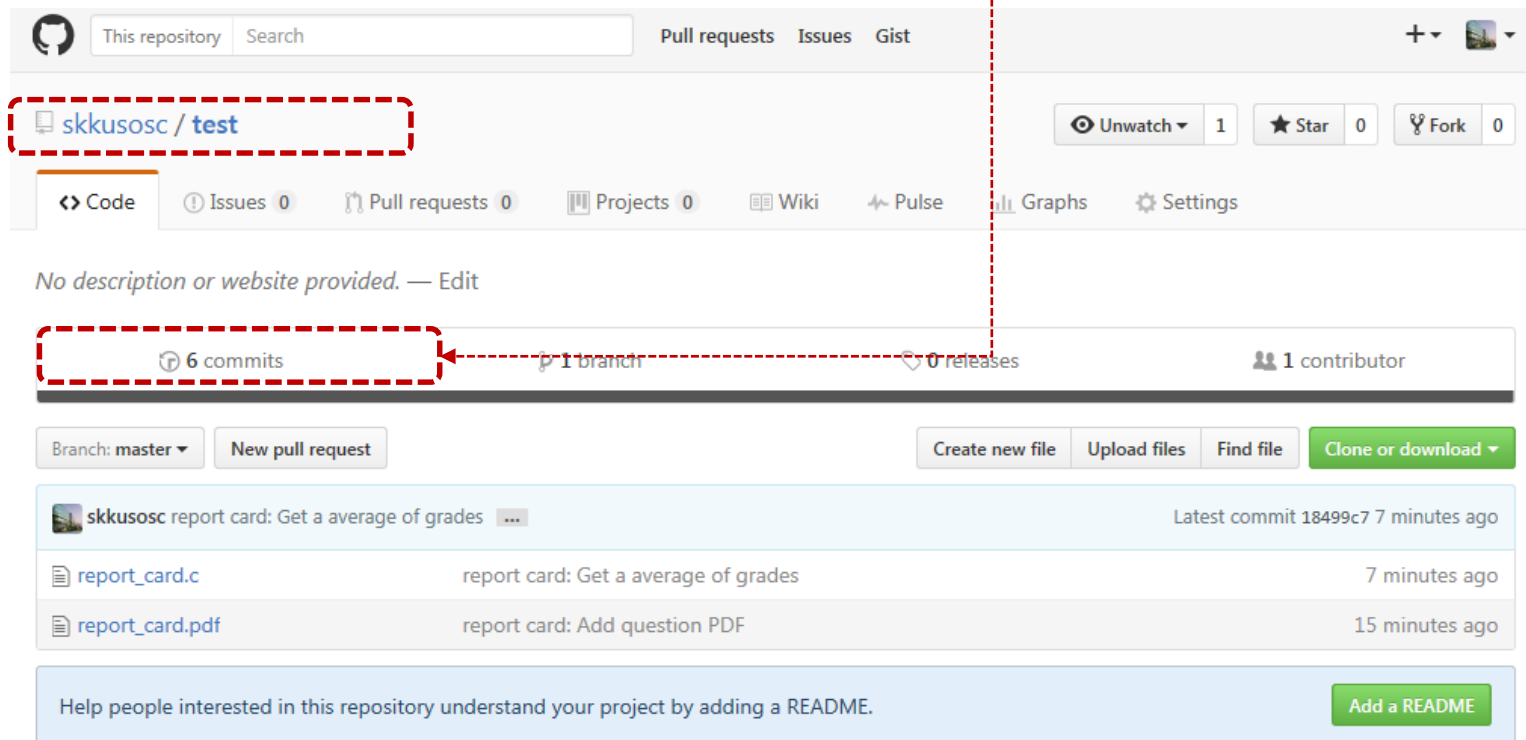
```
$ git push origin master
```

6) GitHub 웹페이지 열고 확인해보자



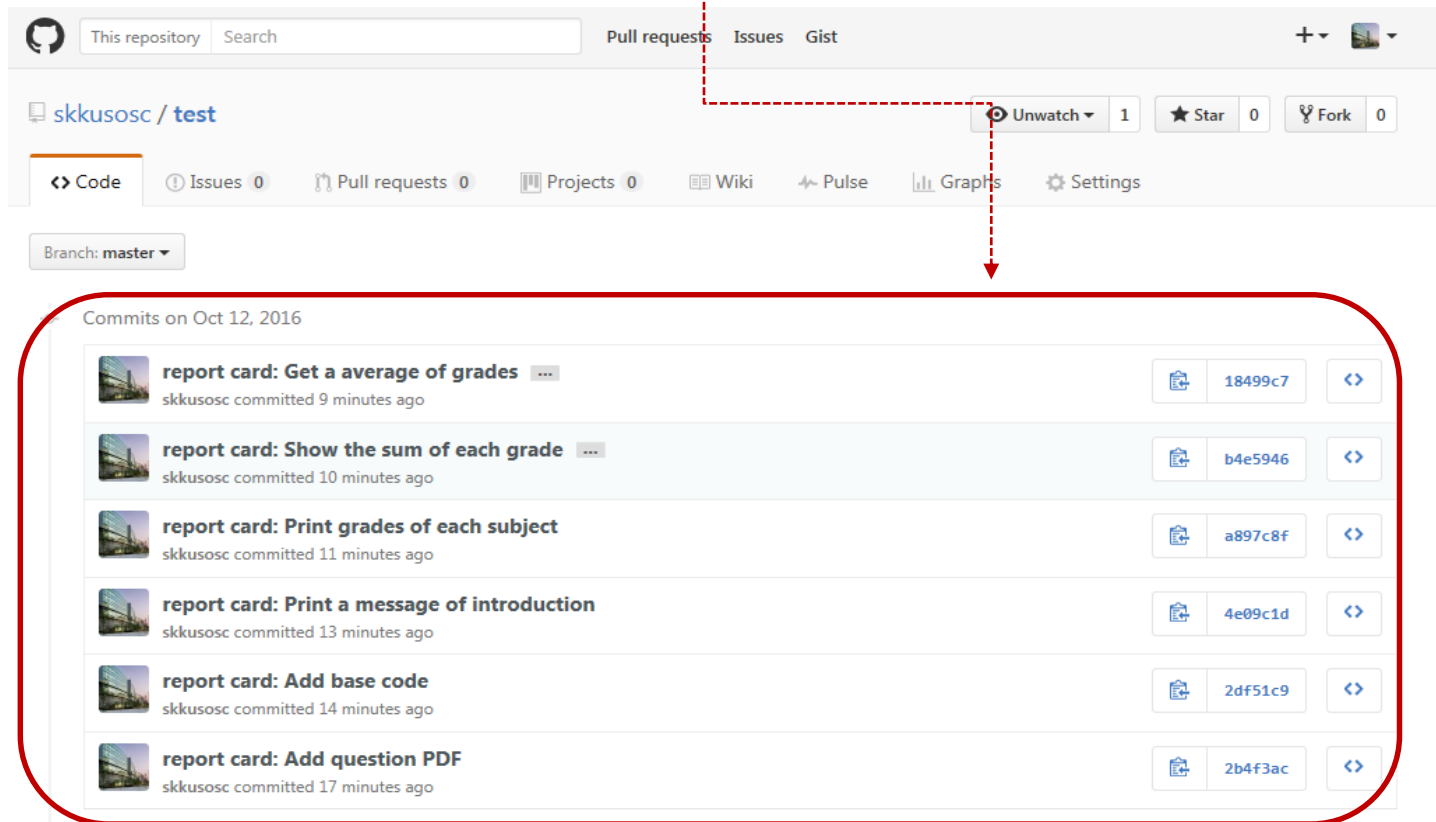
Step4 지금까지의 commit을 push 하자 (Basic)

Github 들어가서 **본인** 프로젝트의 commit 기록 눌러보자



Step4 지금까지의 commit을 push 하자 (Basic)

본인이 추가한 commit 들이 나오는걸 확인하자(본인 Github)



Step5 commit 수정하기 (Basic)

- 1) report_card.c 파일을 열어 'Mean' 변수명을 '**Average**'로 수정하자
(Editor 는 자유)

```
$ git diff
```

- 2) diff를 통해서 변화분을 확인했다면 add진행

```
$ git add report_card.c
```

- 3) 가장 위에 있는 commit을 수정하자

```
$ git commit --amend
```

- vi 에디터 또는 지정된 에디터(메모장 등)이 열릴수 있다.
Commit 메시지를 수정하거나 수정없이 에디터를 닫으면 완료
- Vi에디터는 i또는 a키를 눌러 수정모드로 변경하여 수정 후
ESC키 누르고 :wq 명령어 입력하여 Enter누려 나올 수 있다.

Step5 commit 수정하기(remote도) (Basic)

4) 바로 push 해보자 (충돌 오류발생)

```
$ git push origin master
```

5) 강제로 push 해서 수정하자

```
$ git push origin master -f
```

6) 다시 GitHub 가서 제대로 변경되었는지 확인해보자

- 4)의 충돌이유는 Local (본인 노트북,PC) 에 기록된 commit들과 GitHub에 먼저 push하여 저장된 commit들의 commit ID가 일치하지 않는 부분이 있기 때문임
- 물론 모든 commit ID가 일치한 상태에서 Local에만 새로운 추가 commit 있을때는 push 가능

Step6 add한거 취소하기 (Basic)

1) touch 로 빈파일 생성하고 add 하자 (; 로 명령어들을 연속적 실행가능)

```
$ touch test; git add test
```

2) 현재 상태 확인 해보고

```
$ git status
```

3) reset으로 add한거 취소해보자

```
$ git reset
```

4) 현재 상태 다시 한번 확인해본다

```
$ git status
```

Step7 commit한거 없애기 (Basic)

1) 아까 test파일 여전히 존재하는지 확인 (지웠으면 다시 만들기)

```
$ git status
```

2) 임의로 실수의 commit을 만들어 낸다 (; 로 명령어들을 연속적 실행가능)

```
$ git add test; git commit -sm "test"
```

3) 그리고 push 까지해서 Github에 있는 tree까지 실수 commit을 넣는다

```
$ git push origin master
```

4) Github 가서 확인해보자

Step7 commit한거 없애기 (Basic)

5) 확인 후, 지금까지 한 7개 commit을 확인하자

```
$ git shortlog
```

6) 가장 최근 commit을 지우자

```
$ git reset HEAD~1
```

7) commit이 지워졌는지 확인해보자 (commit 기록 7 -> 6)

```
$ git shortlog
```

8) 강제로 GitHub에 있는 tree 도 밀어 넣어서 수정한다

```
$ git push origin master -f
```

9) GitHub 가서 확인해보자

➤ GitHub에 있는 commit을 수정할 길은 Local에서 수정후 -f옵션으로 push 하는 방법 뿐

반복연습

Keep going

다시 Step1 부터 새롭게 스스로 **반복연습**을 해보자



Contents

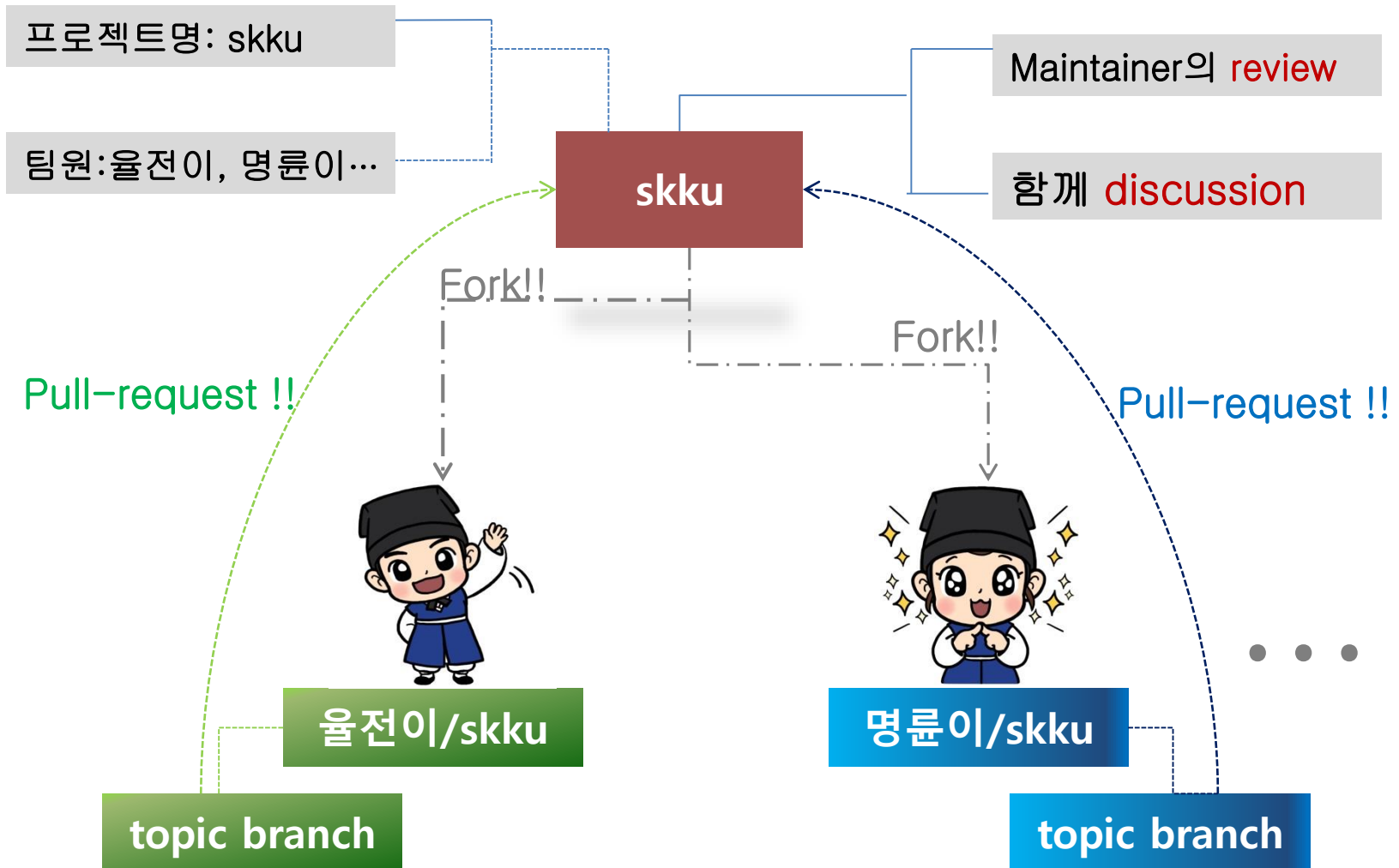
I
Git/GitHub
Intro

II
Hands-on
(Basic)

III
Hands-on
(Advanced)

IV
Git/GitHub
Review

우리 프로젝트와 Git 운용 전략



GitHub Flow

GitHub은 Pull-request가 중심인 협업 워크플로 위주로 설계 되어 있음

다른 프로젝트에 내가 만든 commit을 제출한다는 의미(실제 전송단위는 branch)

1. 기존 프로젝트(master)에서 Fork(복사) 해온다

2. Clone해서 토픽 브랜치를 만든다

3. 내가 만든 commit (원가 수정)을 보낸다

4. 자신의 GitHub 프로젝트에 브랜치를 Push 한다

5. 기존 프로젝트(master)에 Pull-request를 보낸다

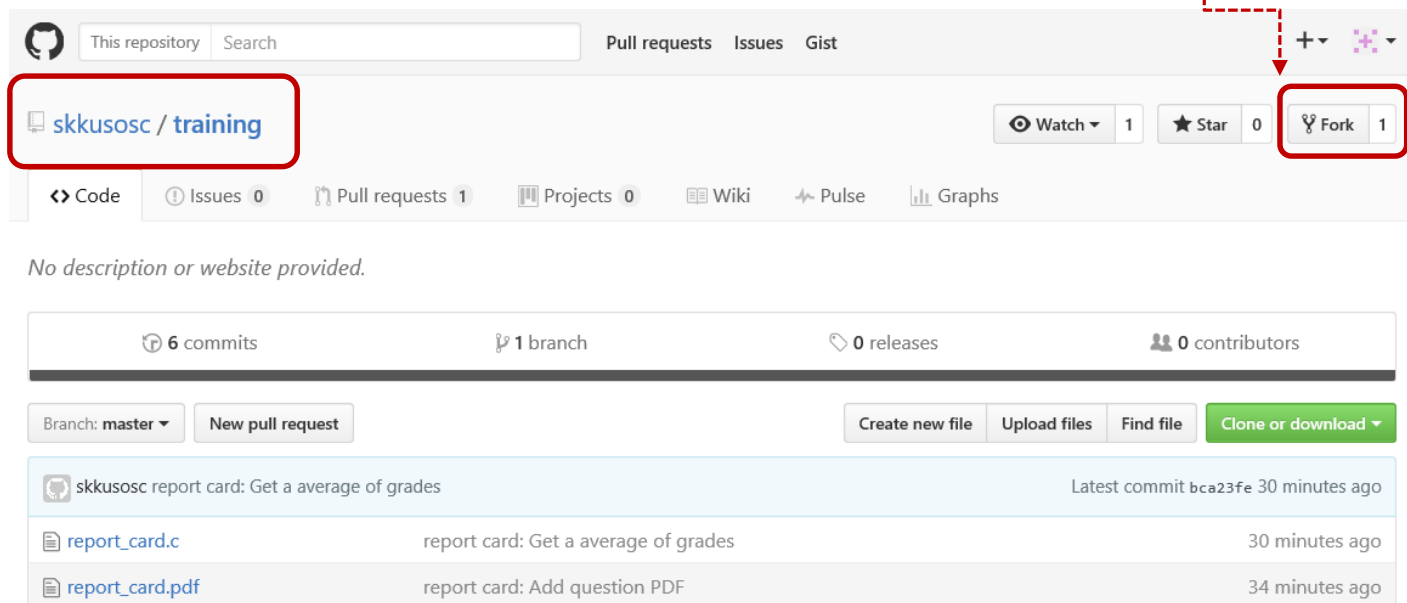
6. 토론하면서 그에 따라 계속 커밋한다

7. 기존 프로젝트(master)소유자는 Pull-reques를 검토후, Merge한다

Step8 Fork & Clone 하기

➤ 주의 fork는 본인 프로젝트를 대상으로 하는게 아니다. 아래 url 들어가자

<https://github.com/skkusosc/git--training> 가서 **Fork** 버튼 누르자



Step8 Fork & Clone 하기

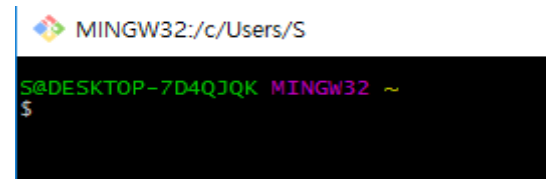
Fork 가 되면 내 원격저장소가 추가 된다

Clone or download 초록색 버튼 클릭!!
Fork해서 만들어진 본인 repo의 URL 복사
*주의)skkusosc/training 원본프로젝트 URL 사용하면 안됨

필수) forked from skkusosc/training 표시 확인하기

Step8 Fork & Clone 하기

상위폴더로 이동 한 후,



- 1) (git-bash/터미널에서) 최초 경로 HOME 경로로 이동하자
(report-card 작업하던 폴더에서 벗어나기)

```
$ cd ~
```

- 2) clone 으로 fork한 repo 받아오기 " 아까 fork한 repo에서 복사한 URL "

```
$ git clone https://github.com/skkusosc/git--training
```

- 3) clone 한 프로젝트 폴더로 이동하기 (만약 프로젝트명이 git--training 이면 그 이름으로 이동)

```
$ cd git--training
```

- 4) 작업할, 토픽 브랜치(develop) 따로 만들기

```
$ git checkout -b develop
```

*브랜치 생성이란?
간단한 비유로 말하면

- 5) pull_request_test 폴더로 이동하자

```
$ cd pull_request_test
```

“같은 폴더에 또 다른 세상 열기 ”

Step8 Fork & Clone 하기

6) 내 이름으로 된(skkusosc 대신) 폴더 만들고

```
$ mkdir yuljeon; cd yuljeon
```

7) 내 이름으로 된(skkusosc 대신) 폴더에 작업하던 report_card.c 소스 파일 또는 아무파일 복사해서 넣기

8) 추가한 폴더 (임의의 소스파일/내가 작업한 소스내용) 통째로 add

```
$ git add report_card.c
```

9) 준비된 파일 commit

```
$ git commit -sm "test pull request"
```

10) 내가 fork한 repo의 develop 브랜치로 push (주의: master 아님)

```
$ git push origin develop
```

Step8 pull-request 하기

★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](#)

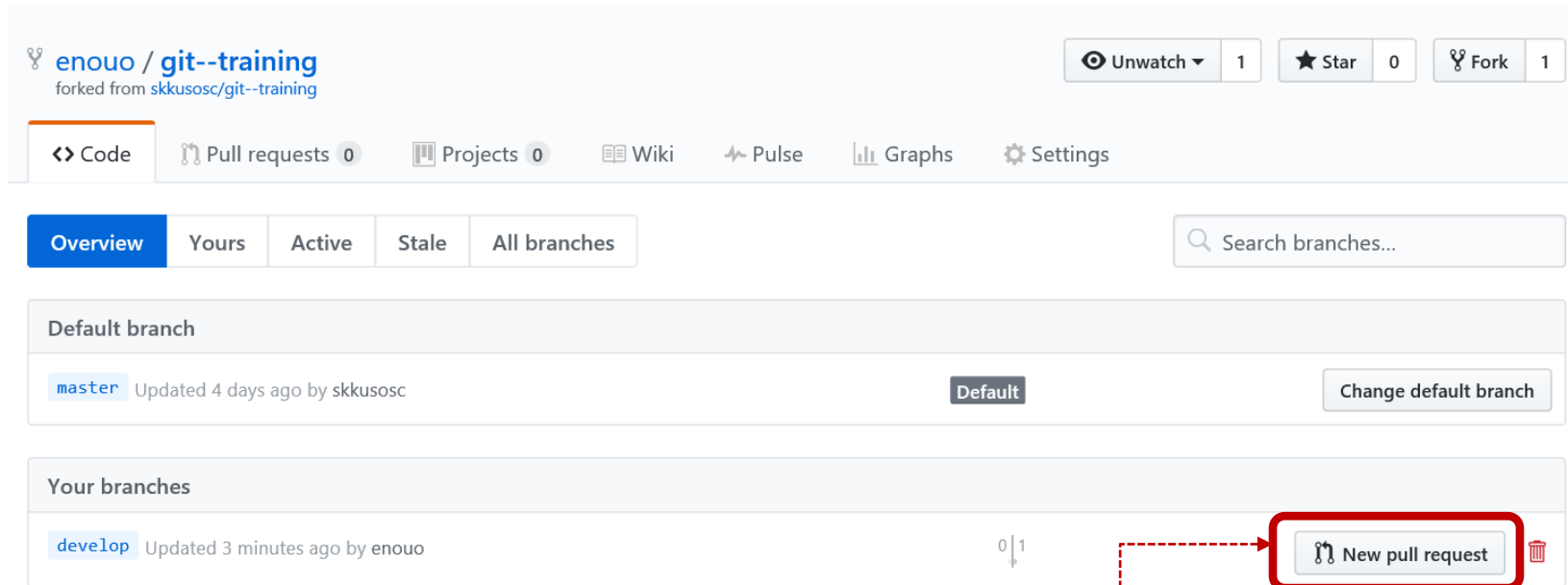
Branch: master New pull request Create new file Upload files Find file Clone or download

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

Repository	Commit	Time
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

Step8 pull-request 하기



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

Step8 pull-request 하기

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:

본인 이름: 추가 작성

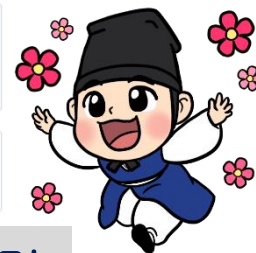
Attach files by dragging & dropping or [selecting them](#).

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

Create pull request

여기까지 성공하면
Mission 완료!

Pull-request 보내기



Step8 pull-request 하기

skkusosc / git--training

Watch 1 Star 0 Fork 1

Code Issues 0 Pull requests 1 Projects 0 Wiki Pulse Graphs

test pull request #2

Edit

Open enouo wants to merge 1 commit into skkusosc:master from enouo:develop

Conversation 0 Commits 1 Files changed 1 +21 -0

enouo commented just now

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

test pull request ... af1f6d6

Add more commits by pushing to the `develop` branch on `enouo/git--training`.

✓ This branch has no conflicts with the base branch
Only those with [write access](#) to this repository can merge pull requests.

Write Preview AA B i “ < > ↺ ⋮ ≡ ≡ ≡ ↶ @ 📌

Styling with Markdown is supported

Close pull request Comment

1 participant

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

Step9 merge로 2개 브랜치 합치기(자유 실습)

1) 방금 작업한 develop 브랜치가 현재 브랜치인지 확인하자 (status로도 확인 가능)

```
$ git branch
```

2) 추가 브랜치 만들어보자

```
$ git checkout -b test
```

3) Touch로 빈파일 하나 만들어서 commit 만들어보자 (;로 명령어들 연속적 실행 가능)

```
$ touch test; git add test; git commit -sm "test"
```

4) 현재 브랜치(develop)를 기준으로 추가 브랜치(test)을 합치자

```
$ git checkout develop; git status; git merge test
```

Contents

I
Git/GitHub
Intro

II
Hands-on
(Basic)

III
Hands-on
(Advanced)

IV
Git/GitHub
Review

Commit 단위개발의 정신

혼자 가면 빨리 가지만
함께 가면 멀리 간다



Commit 단위개발 IDEA

Commit 단위로 코딩하고 리뷰하고 토론하고 적용한다. (집단지성의 극대화)

소프트웨어의 취약점을 극복하는 전략



Git/GitHub(1) Review

오늘 이것만은 기억하자

1) Git 과 GitHub 차이는 ?

- Git 은 각 컴퓨터 (**local**) 에 설치되어 소스코드 관리가 가능한 프로그램
- GitHub 는 **remote** 저장소가 있는 외부서버를 지칭
- Git 이라는 **Source Control** 방법을 Github이 사용할 뿐

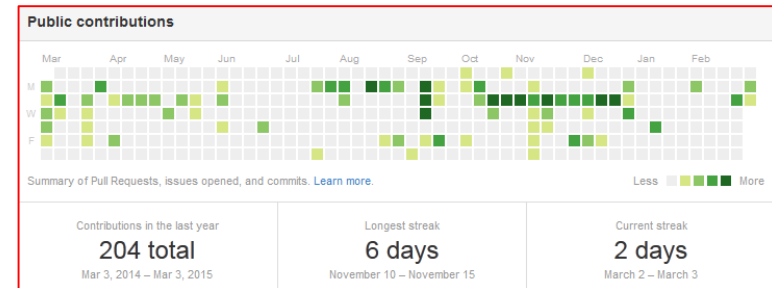
2) Commit 과 Push 차이는 ?

- commit 은 **local** 작업폴더에 history 를 쌓는 것으로 외부망(**internet**) 필요없음
- push 는 **remote** 저장소(GitHub 등) 에 history 를 쌓는 것이어서 외부망(**internet**)이 필요하다

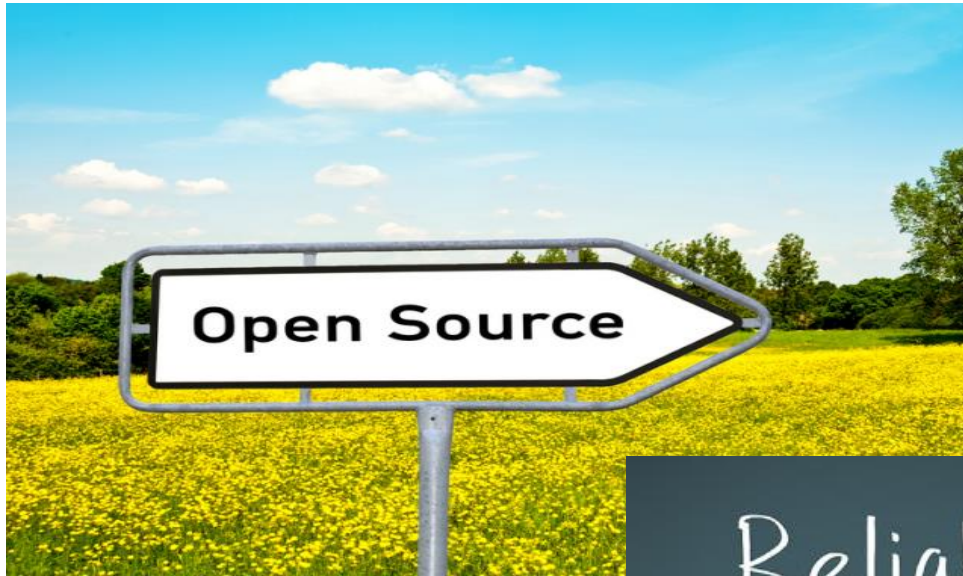
Git/GitHub(2) Review

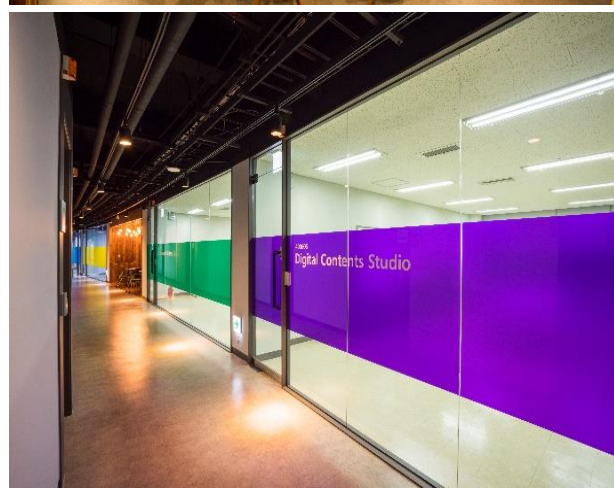
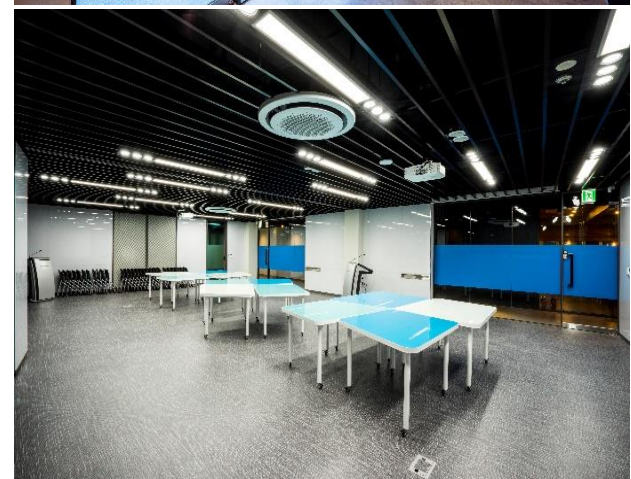
오늘 이것만은 기억하자

- GitHub을 쓸수 있다는 것은
 - Source Control
 - Issue Tracking/Control
 - 협업 도구 (Fork / pull request)
 - Statistics 등을 쓴다는 것



참여와 공유





SUNGKYUN OPENSOURCE SOFTWARE CENTER (SOSC)

Thank you.

400624

Connection cafe