

Git & GitHub 소개

최영원

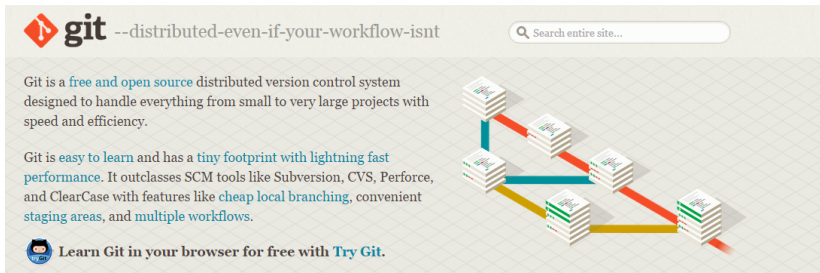
서울대학교 통계학과

2018년 9월 10일

Git과 GitHub 소개

Git?

- 분산형 소스 코드 버전 관리 시스템(DVCS)
- 리눅스에서 사용하던 Bitkeeper의 라이선스 이슈로, Linus Torvalds가 직접 개발
- GitHub의 등장과 함께 폭발적으로 성장

A screenshot of the Git website banner. It features the Git logo (a red diamond with a white 'G') and the text 'git --distributed-even-if-your-workflow-isnt'. Below this, there is a search bar with the placeholder text 'Search entire site...'. The main text describes Git as a 'free and open source' distributed version control system designed for speed and efficiency. It also mentions that Git is 'easy to learn' and has a 'tiny footprint with lightning fast performance', outclassing other SCM tools like Subversion, CVS, Perforce, and ClearCase. The text highlights features like 'cheap local branching', 'convenient staging areas', and 'multiple workflows'. At the bottom left, there is a 'Try Git' button with a small Git logo. On the right side, there is a 3D illustration of several stacks of papers or documents connected by colored lines (red, blue, yellow) in a network-like structure, representing a distributed version control system.

git --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

Try Git Learn Git in your browser for free with **Try Git**.

버전 관리란?

- 동일한 정보에 대해 여러 버전을 관리하는 것
- 버전을 통해 시간적으로 변경사항과 그 내용을 변경한 작업자를 추적할 수 있다.



DVCS (Cont'd)

- 수정이 있었다.



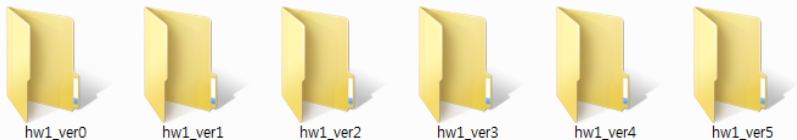
hw1_ver0



hw1_ver1

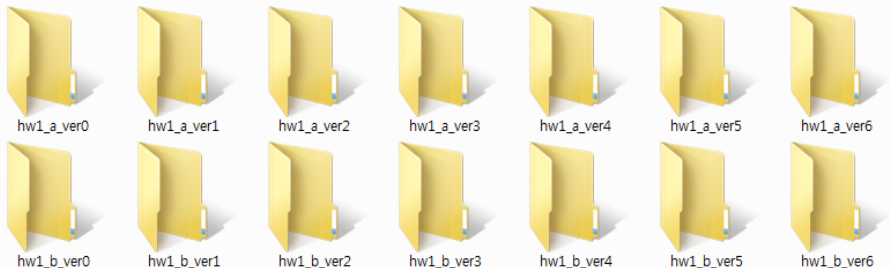
DVCS (Cont'd)

- 많은 수정이 있었다....



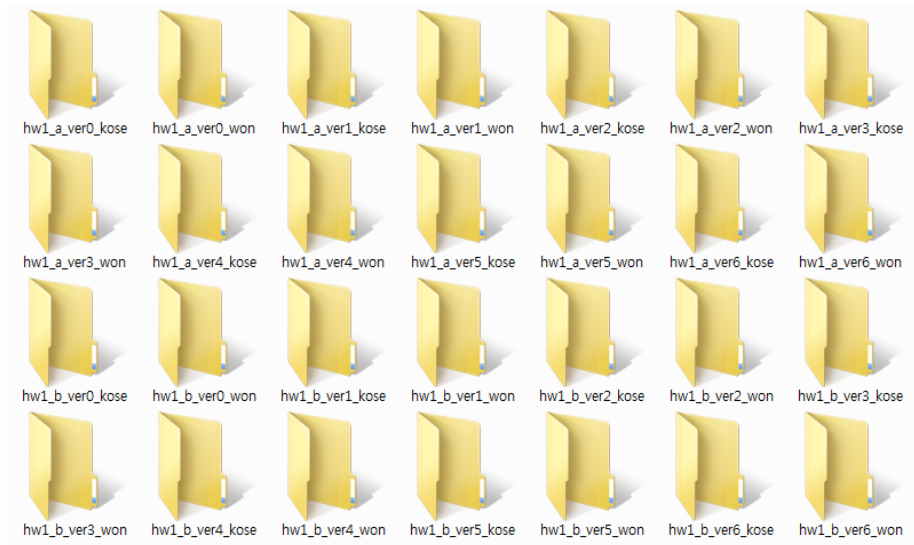
DVCS (Cont'd)

- 숙제를 위한 코드가 하나가 아니라면?



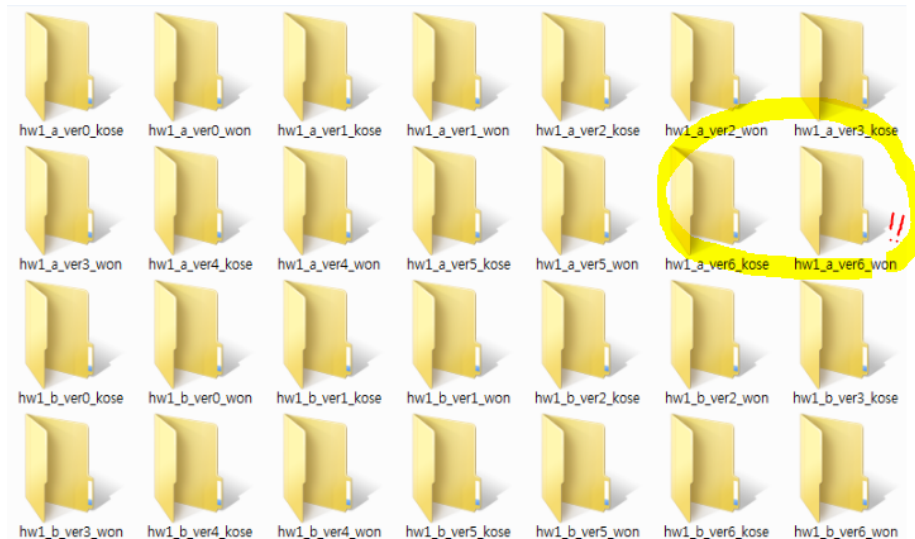
DVCS (Cont'd)

- 팀 프로젝트라면?usb?



DVCS (Cont'd)

- 합쳤는데 버그가!!



DVCS (Cont'd)

- 버전 관리를 전문적으로 해주는 소프트웨어가 있다.

어떤 소프트웨어가 있나?

많다



git



Bazaar



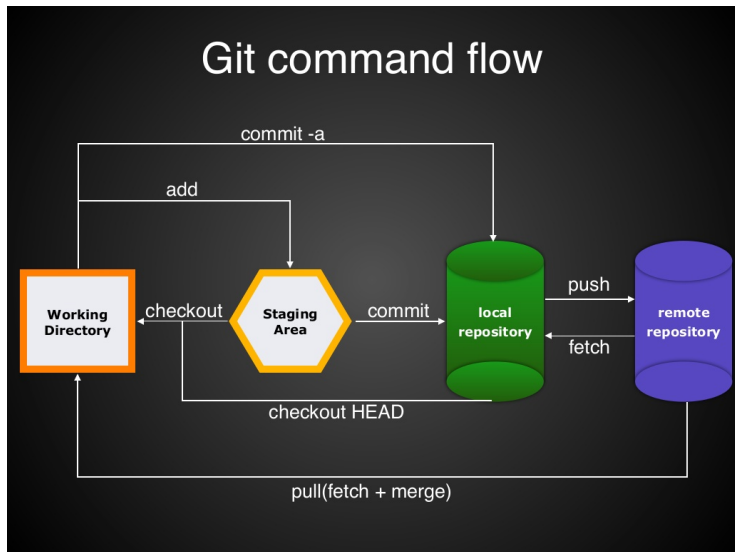
source: [http:](http://www.slideshare.net/ibare/dvcs-git?qid=b93af986-f188-4813-abd4-eb13631bf51f&v=default&b=&from_search=2)

[//www.slideshare.net/ibare/dvcs-git?qid=b93af986-f188-4813-abd4-eb13631bf51f&v=default&b=&from_search=2](http://www.slideshare.net/ibare/dvcs-git?qid=b93af986-f188-4813-abd4-eb13631bf51f&v=default&b=&from_search=2)

DVCS (Cont'd)

distributed version control system: centralized version control과 상반되는 개념으로, 각 개발자가 중앙 서버에 접속하지 않은 상태에서도 코드 작업을 할 수 있어 버전 역사와 같은 정보를 로컬 컴퓨터에서 빠르게 확인할 수 있다.

Git의 개념 및 특징



Git의 개념 및 특징 (Cont'd)

- 소스 코드의 변경 이력을 관리
 - ▶ 특정 시점 및 변경 추적
 - ▶ 원하는 시점으로 파일의 상태 변경이 가능 (유연성을 제공)
- 분산형 소스 코드 버전 관리 시스템
 - ▶ 지역 저장소와 원격 저장소로 구분
 - ▶ 지역 저장소에서 개별 버전 관리
 - ▶ 원격 저장소로 내용 공유
 - ▶ 인터넷 연결이 되지 않아도 버전 관리가 가능
 - ▶ 원격 저장소 장애로 내용물 유실시에도 지역 저장소에서 복구 가능

Git의 개념 및 특징 (Cont'd)

- 브랜치 모델

- ▶ 비 선형적 개발 지원
- ▶ 병행개발을 통한 지속적이고 안정적인 소프트웨어 개발
- ▶ 다양한 버전의 시도 및 관리
- ▶ 신규기능, 버그수정, 실험적 작업을 위해 사용

- 데이터 무결성

- ▶ 모든 변경 사항에 대한 스냅샷을 저장

Git 기본 용어

- 작업 공간(Working Directory)

- ▶ 내 컴퓨터(local)의 소스 코드 작업이 이루어지는 폴더
- ▶ 작업 공간에서 Git 초기화를 하면 저장소가 만들어진다.

- 저장소(Repository)

- ▶ 프로젝트가 거주(live)하는 저장 공간. 종종 'repo'라고 불린다.
- ▶ 컴퓨터의 로컬 폴더일 수도 있고, 온라인 호스트(ex. GitHub)의 저장 공간일 수도 있다.
- ▶ Git 정보가 저장되어 있다(보이지는 않는다).
- ▶ 작업 공간에서 저장소로 의미 단위의 작업 스냅샷을 전달해(커밋, Commit) 버전을 관리한다.

- 스테이징 영역(Staging Area)

- ▶ 커밋될 예정인 파일의 내용을 준비하는 곳
- ▶ 쇼핑몰에서 물건을 구매할 때 먼저 장바구니에 넣는 것과 유사

Git 기본 용어 (Cont'd)

- 지역 저장소(Local Repository)

- ▶ 작업이 실제로 이루어지는 컴퓨터의 저장소
- ▶ remote를 통해 원격 저장소와 연결

- 원격 저장소(Remote Repository)

- ▶ Git server를 구축해서 쓰는 경우 해당 server에 만들어진 저장소
- ▶ 온라인 호스트(ex. GitHub)를 이용하는 경우, 호스트에서 제공하는 온라인 상의 저장 공간

- 커밋(Commit)

- ▶ 그 시점의 지역 저장소의 '스냅샷'을 찍어, 프로젝트를 이전의 어떤 상태로든 재평가하거나 복원할 수 있는 체크포인트를 만든다.
- ▶ 사용자가 의미 있는 작업 뭉치를 판단해 커밋을 명령한다.
- ▶ 커밋 시 작업 내용을 간략하게 comment로 남긴다.

GitHub이란?

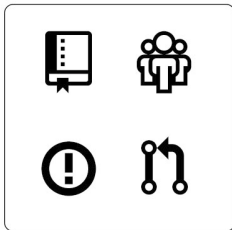
분산 버전 관리 시스템(DVCS) “Git” 위에서 만들어진 협업 플랫폼



GitHub이란? (Cont'd)

- Git 프로젝트를 호스팅하는 기능 뿐만 아니라, 협업을 위한 여러 기능을 갖추고 있음
 - ▶ Issues
 - ★ 기능과 버그에 관한 논의
 - ▶ Pull Requests
 - ★ 프로젝트에 추가/변경하고자 하는 기능에 대한 논의와 리뷰

GitHub



GitHub이란? (Cont'd)

“프로그래머의 페이스북 같은 존재”

- 많은 오픈 소스 패키지들이 GitHub을 사용하고 있음
 - ▶ 프로그래밍 언어: Scala, Julia
 - ★ 2016년 1월 1일: Python이 GitHub로 이주하기로 결정
 - ▶ 과학계산: NumPy, SciPy
 - ▶ 기계학습: scikit-learn, Apache Mahout
 - ▶ 딥러닝: Theano, TensorFlow, Caffe, Keras, Neon
 - ▶ 수업: 많은 coursera, edx 등 오픈 코스 강의들
 - ★ 2015년 2학기부터: 전산통계 및 실험, 통계계산에서 사용 중

GitHub의 주요 기능

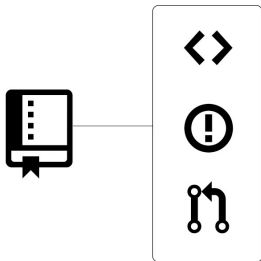
- 이슈
- 풀 리퀘스트
- 커밋 역사 확인
- 그래프
- 변경 내역 확인
- 이메일 알림
- 멘션
- 간단한 웹사이트 호스팅
 - ▶ https://won-j.github.io/326_212-2018fall/ (2018년 전산통계 및 실험)

저장소 둘러보기

<https://github.com/snu-stat/sc18-commonplace>

- 저장소(repository)

- ▶ 프로젝트가 담겨 있는 디렉토리 또는 저장 공간. 줄여서 repo라고도 한다.
- ▶ 저장소는 개별 컴퓨터의 폴더 안에 있을 수도 있고 GitHub나 다른 온라인 호스트의 공간에 있을 수도 있다.
- ▶ 숨김 파일의 형태로 Git 정보가 저장되어 있다.
- ▶ 코드 파일, 텍스트 파일, 이미지 파일 등등을 저장할 수 있다.



저장소 둘러보기 (Cont'd)

- Code

- ▶ repo에 있는 파일들을 확인할 수 있다.
- ▶ 프로젝트, 문서, 다른 중요한 파일 등등
- ▶ 프로젝트의 root라고도 한다.
- ▶ 변경 사항은 Git의 버전 관리에 의해 추적된다.

- README.md

- ▶ 각 저장소에 기본적으로 포함하는 것을 권장함.
- ▶ GitHub에서는 이 파일의 내용을 repository 아래에 표시한다.
- ▶ 프로젝트를 설명하고 프로젝트 내의 도움이 되는 정보들을 가리켜야 함.

저장소 둘러보기 (Cont'd)

이슈

- Issues

- ▶ 버그 추적과 기능 요청에 사용
- ▶ 특정 팀 멤버들에게 할당될 수 있으며 활발한 논의와 협업을 위해 만들어짐

저장소 둘러보기 (Cont'd)

풀 리퀘스트

- Pull Request

- ▶ 작성자가 저장소에 주고자 하는 파일 추가, 변경, 삭제 등의 변화를 나타냄
- ▶ Issue를 해결하기 위해 사용

GitHub 작업의 흐름

GitHub 작업의 흐름

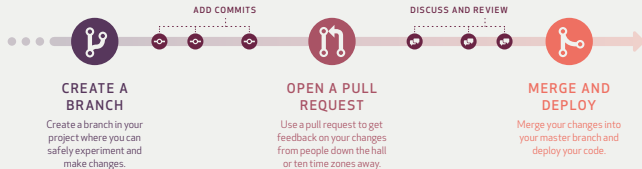
새로운 아이디어를 프로젝트를 망칠 걱정 없이 안전하게 시험하기 위한 과정

- 브랜치 만들기
- 커밋
- 풀 리퀘스트
- 병합

GitHub 작업의 흐름 (Cont'd)

WORK FAST WORK SMART THE GITHUB FLOW

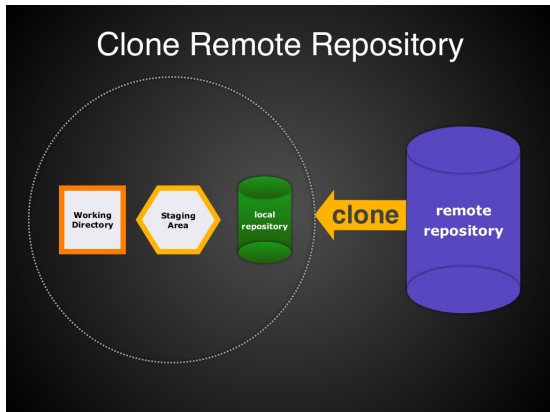
The GitHub Flow is a lightweight, branch-based workflow that's great for teams and projects with regular deployments. Find this and other guides at <http://guides.github.com/>.



GitHub is the best way to build software together.

GitHub provides tools for easier collaboration and code sharing from any device. Start collaborating with millions of developers today!

저장소 복제(clone)하기

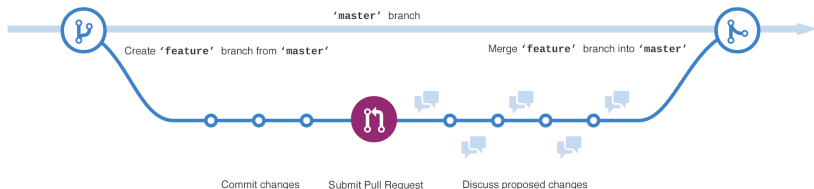


- 복제 (clone)

- ▶ 저장소의 복제본을 로컬 컴퓨터에 저장하는 과정
- ▶ 역사를 포함한 그 저장소의 모든 것을 복제함 (DVCS의 이점)

브랜치 선택 혹은 만들기

- 브랜치(branch): 병행 개발을 위한 분기. GitHub에 있는 모든 것은 각각의 브랜치에 속하게 된다.
 - ▶ master: 관례적으로 모든 작업의 기준이 되는 브랜치.
 - ▶ feature branch: 기능을 추가하기 위해 master에서 분기된 브랜치



갱신 (pull)

- Pull : 로컬 저장소를 원격 저장소에 맞춰 갱신
 - ▶ Github의 저장소의 최신 내용을 로컬 컴퓨터에 반영하고 싶을 때 사용.
 - ▶ 원격 저장소의 변경 내용이 로컬 작업 디렉토리에 받아지고(fetch), 병합(merge) 됨.

로컬 파일 편집하기

- 복제된 저장소가 있는 공간에서 파일을 원하는 대로 편집
- 2단계(two-stage) 커밋(commit)
 - ▶ 커밋(commit) : 저장소의 “스냅샷”
 - ▶ 프로젝트를 이전의 어떤 상태로든 재평가하거나 복원할 수 있는 체크포인트를 만든다.

2단계(two-stage) 커밋

- 스테이징 (add)
 - ▶ 커밋될 예정인 파일의 내용을 준비
 - ▶ 쇼핑몰에서 물건을 구매할 때 먼저 장바구니에 넣는 것과 유사
- 커밋(commit)
 - ▶ 저장소의 “스냅샷”
 - ▶ 프로젝트를 이전의 어떤 상태로든 재평가하거나 복원할 수 있는 체크포인트를 만든다.



working



staging



history

변경된 내용 동기화(Synchronize)하기

- 컴퓨터 내에서 변경한 내용을 GitHub 서버에 업데이트
- 방법
 - ① 풀 리퀘스트 & 병합
 - ② Push

풀 리퀘스트 (pull request)

- 프로젝트 파일의 변경을 제안하기 위해 사용
- 이슈를 해결하기 위한 행위
- 풀 리퀘스트는 병합되기 전까지 '진행중인 작업'으로 간주되며, 커밋을 추가할 수 있음

병합 (merge)

- master 브랜치에 변경된 내용을 병합할 준비가 끝났을 때
- 병합하고자 하는 기능 브랜치의 내용과 역사를 master 브랜치의 내용과 역사에 추가함
- 병합이 완료된 브랜치는 삭제할 수 있음
- 그룹에서 누가 풀 리퀘스트를 병합할 지에 대해서는 내부적으로 약속을 해 두는 것이 좋다.

Push

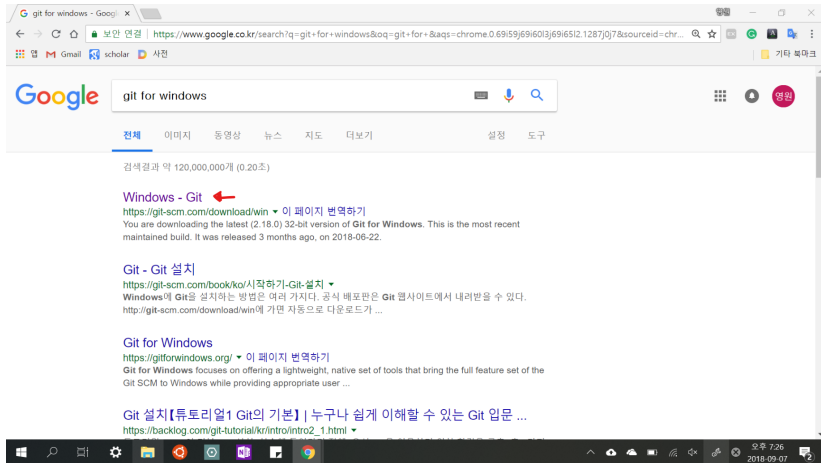
- 로컬 컴퓨터에서 작업하고 커밋을 완료한 내용을 Github 서버에 업데이트
- Clone 한 리모트 저장소에 쓰기 권한이 있고, clone 하고 난 이후 아무도 Github 저장소(repo)에 push 하지 않았을 때만 사용할 수 있다.
- 다른 사람이 push 한 후에 push 하려고 하면 충돌이 발생하므로, push 가 불가능하다. 이 경우 먼저 다른 사람이 작업한 것을 가져와서 merge 한 후에 push 할 수 있다.

실습

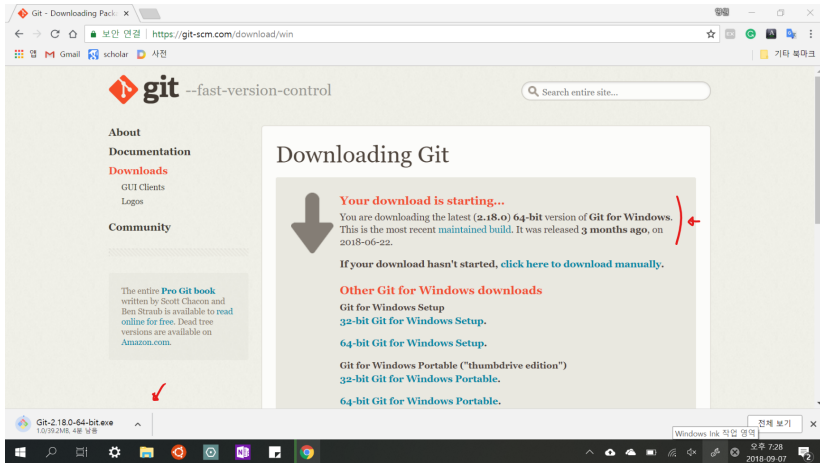
실습 목표

- ① Git 설치하기
- ② Github 가입하기
- ③ Github Desktop 설치하기
- ④ 숙제 repository 설정하기
- ⑤ 풀 리퀘스트로 공지된 새 숙제 merge 하기
- ⑥ 숙제 작업 내용을 커밋하고 Github에 push하기
- ⑦ Github 반영 내용 확인하기

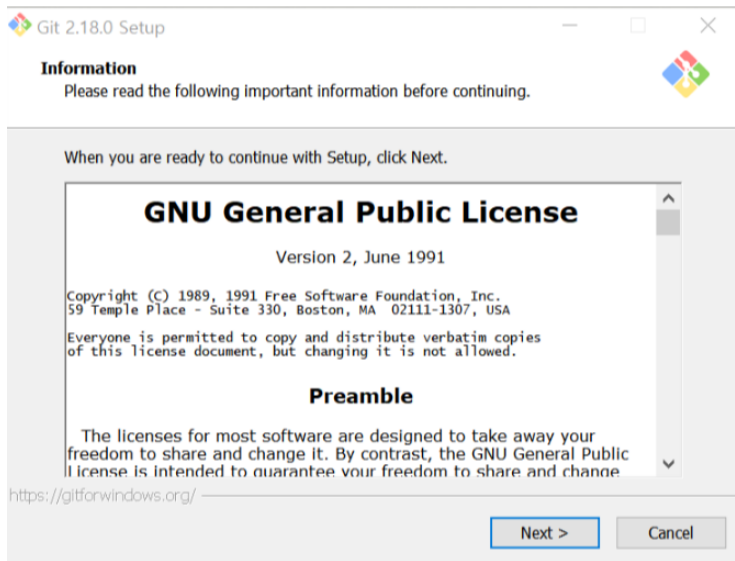
Git 설치하기



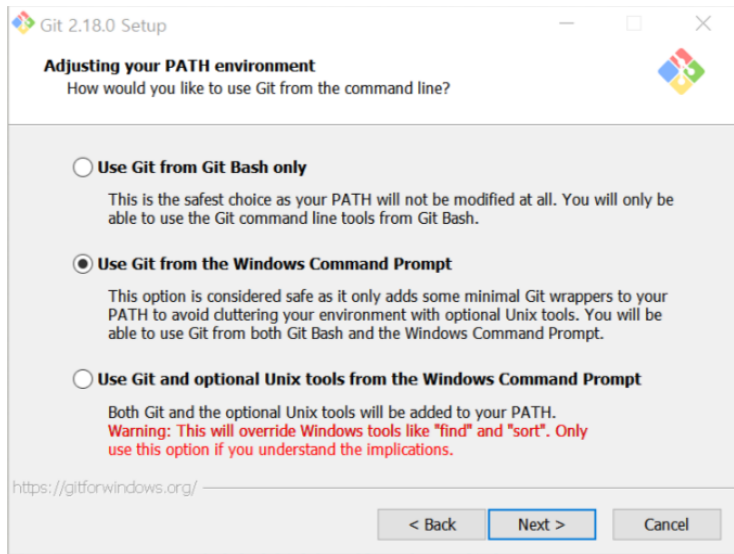
Git 설치하기 (Cont'd)



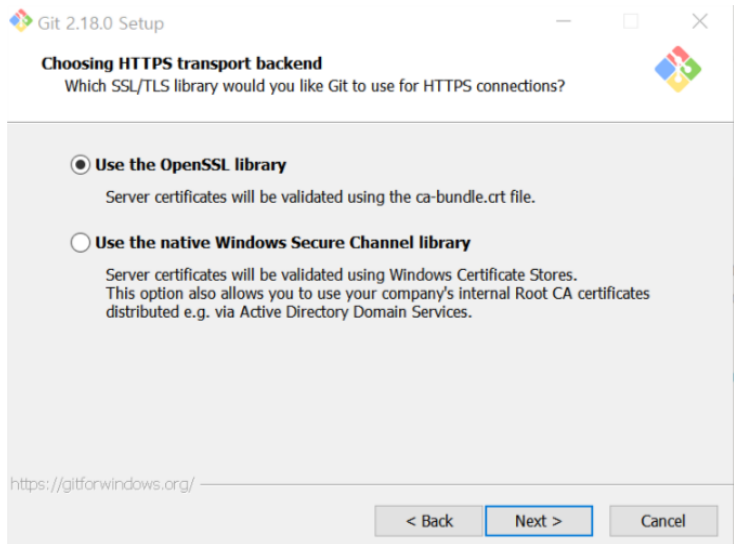
Git 설치하기 (Cont'd)



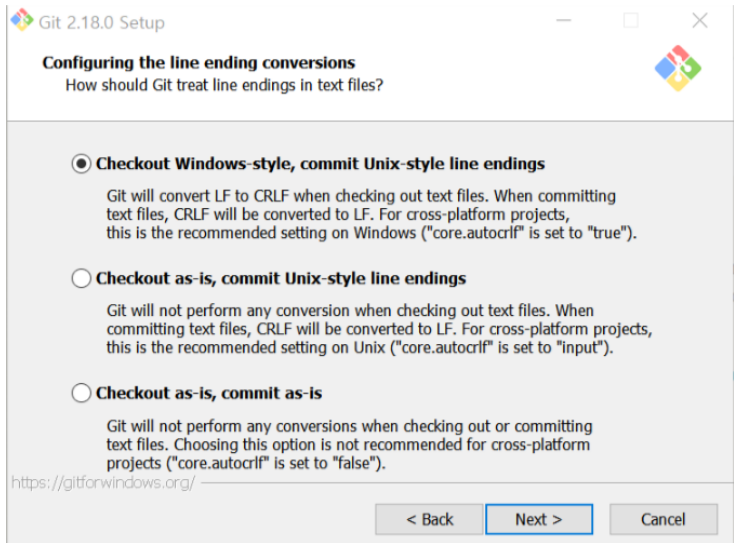
Git 설치하기 (Cont'd)



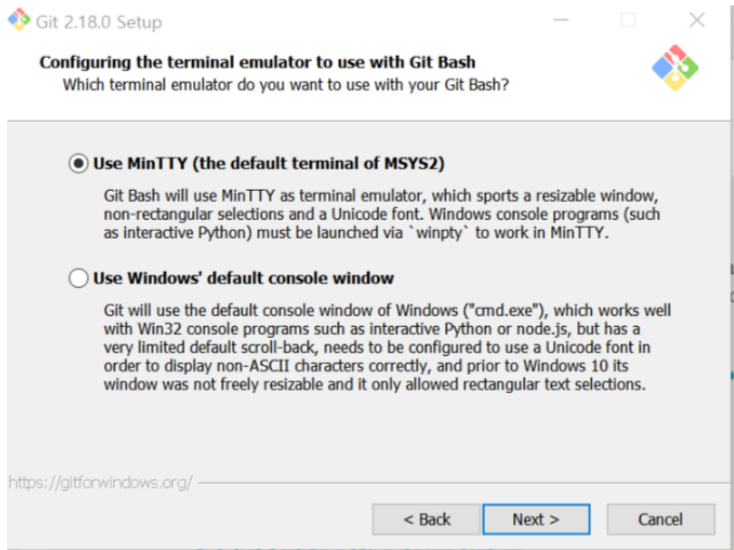
Git 설치하기 (Cont'd)



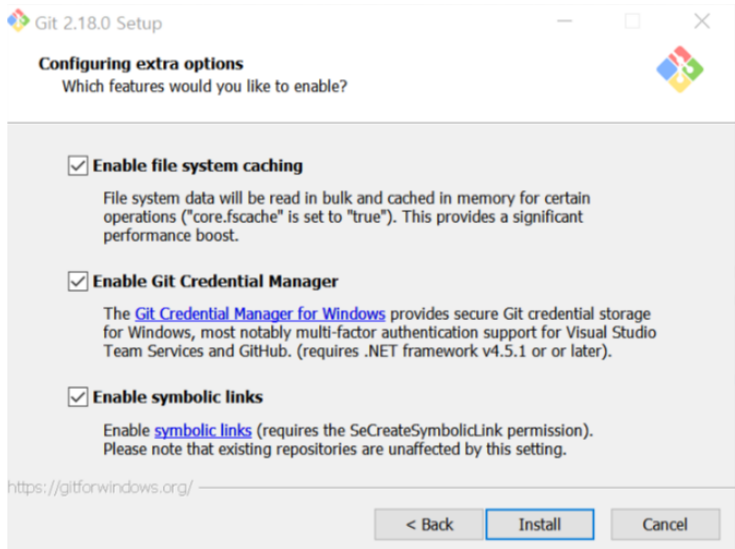
Git 설치하기 (Cont'd)



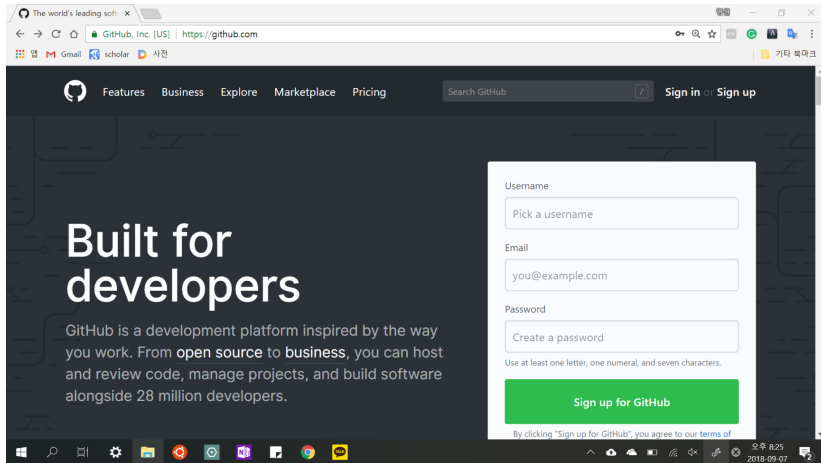
Git 설치하기 (Cont'd)



Git 설치하기 (Cont'd)



Github 시작하기

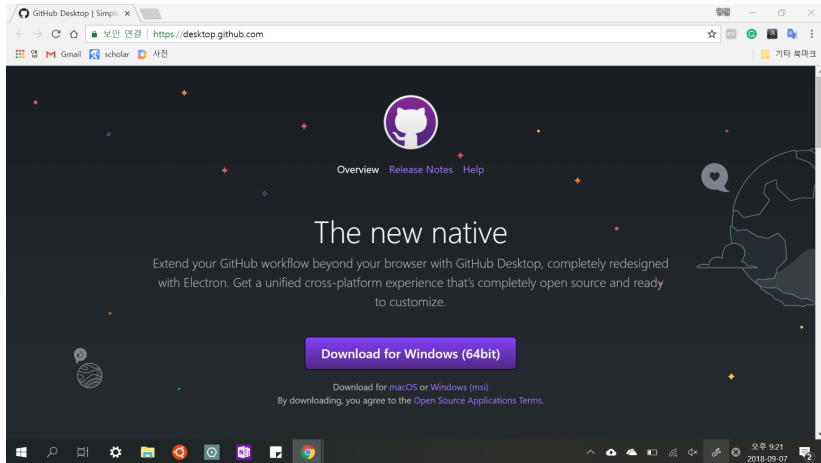


GitHub Desktop

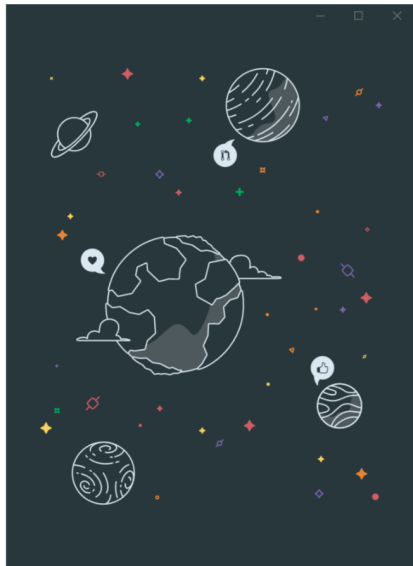
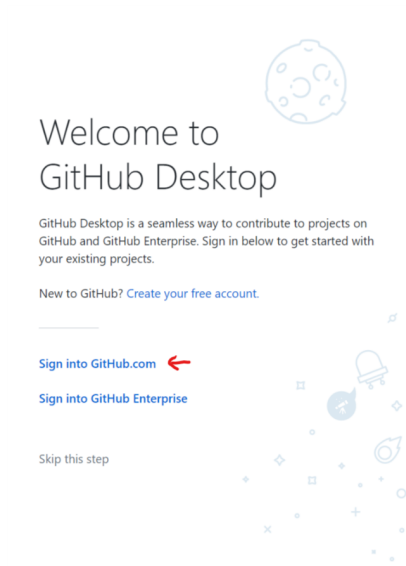
<https://desktop.github.com/>

- Git 명령어를 커맨드라인에서 사용하지 않고도 손쉽게 로컬 컴퓨터에서 GitHub 작업을 할 수 있게 해 주는 도구
- 단순화된 환경
 - ▶ 커밋 되지 않은 파일을 자동으로 스테이징
 - ▶ Pull과 push을 묶어 sync라고 하며, sync 버튼으로 remote와의 상호작용을 할 수 있음
 - ▶ Pull request를 프로그램 내에서 직접 생성 가능

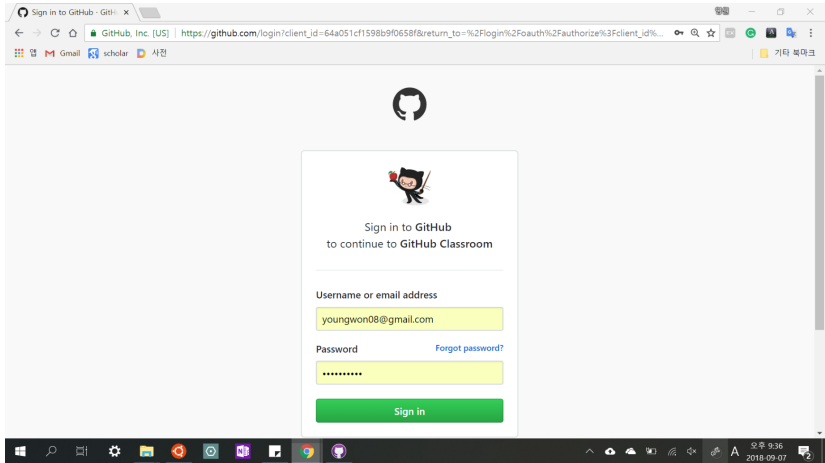
Github Desktop



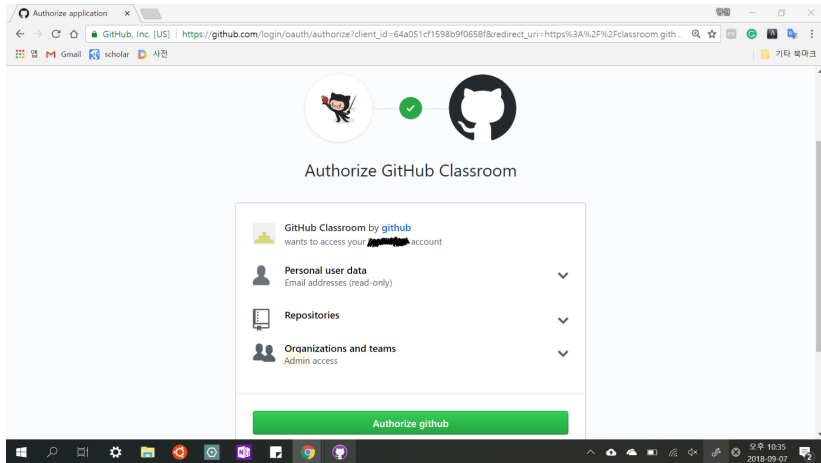
Github Desktop (Cont'd)



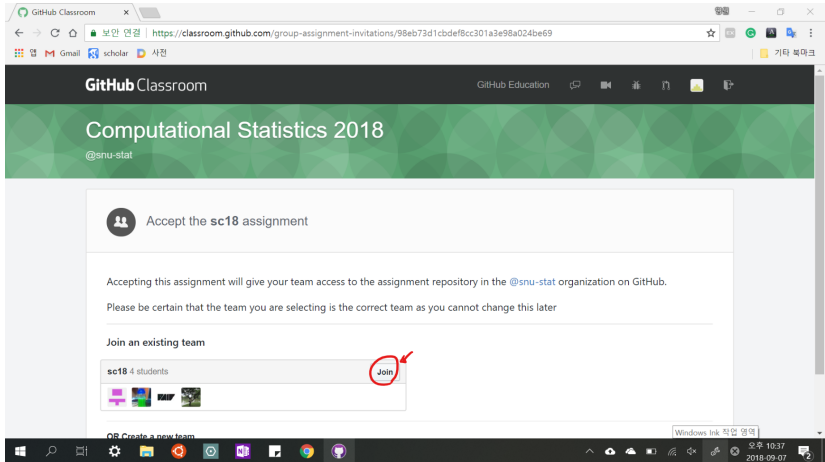
Github Group 가입하기



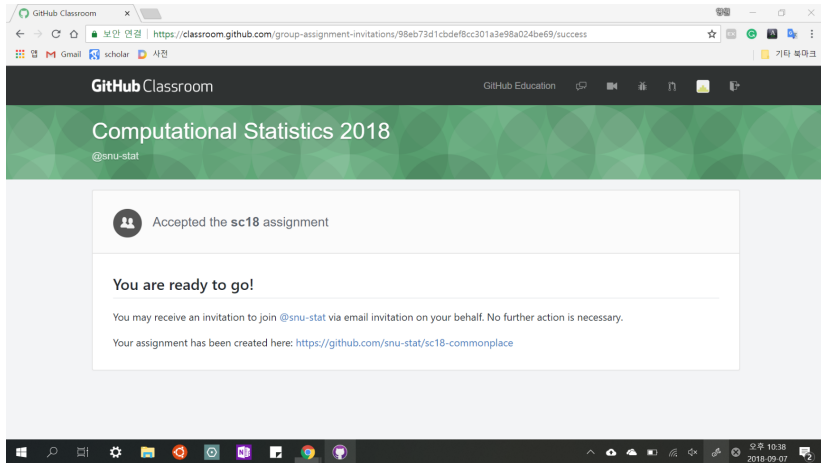
Github Group 가입하기 (Cont'd)



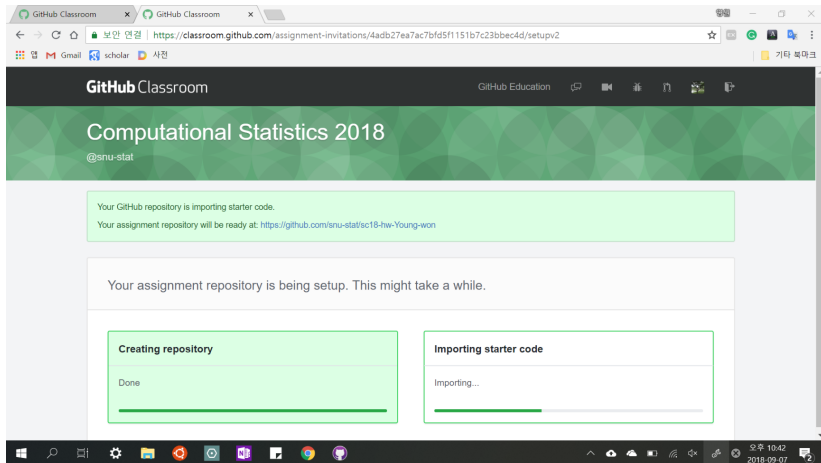
Github Group 가입하기 (Cont'd)



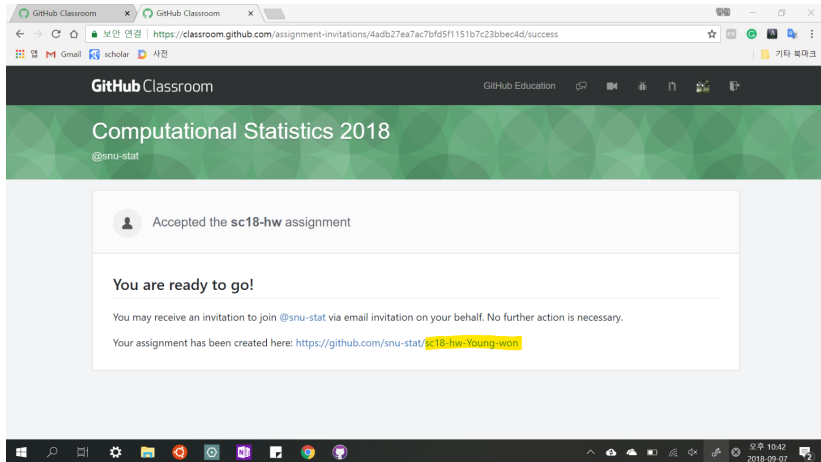
Github Group 가입하기 (Cont'd)



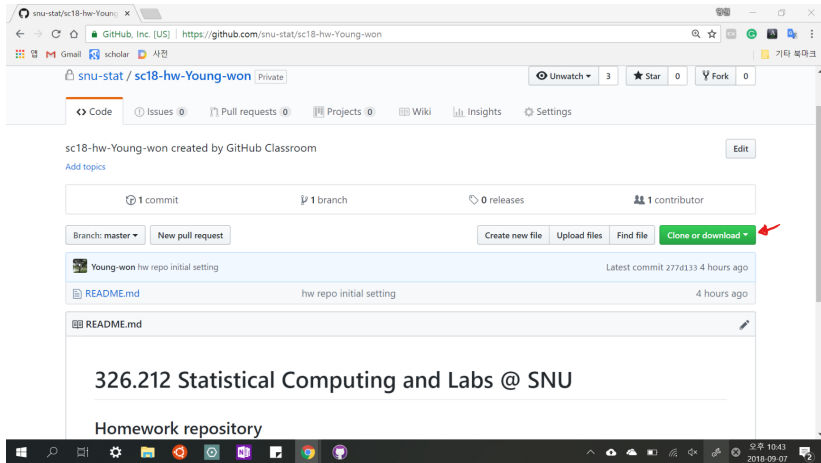
Github 속제 repository 만들기



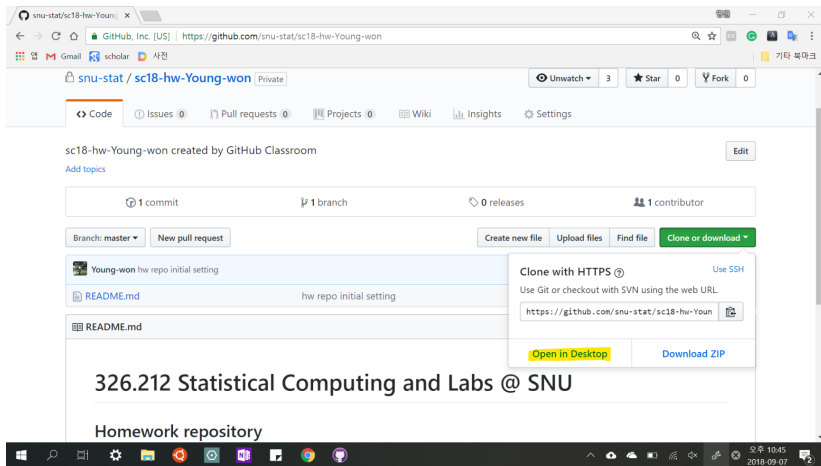
Github 속제 repository 만들기 (Cont'd)



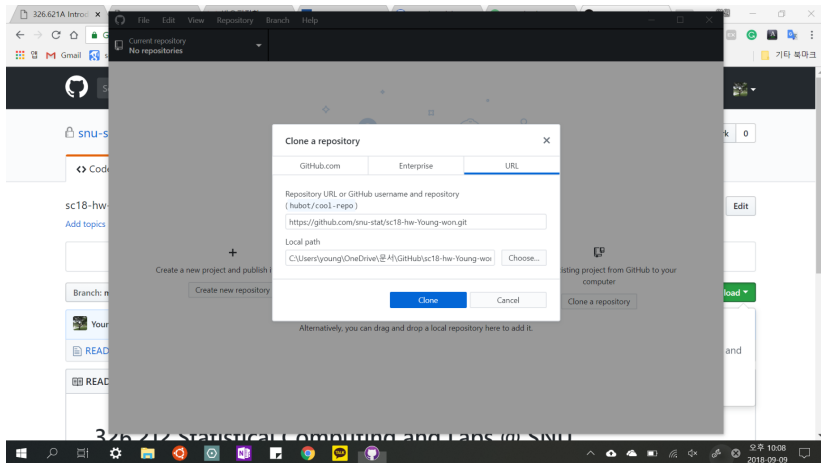
Github 속제 repository를 로컬 저장소에 복제 (Clone)



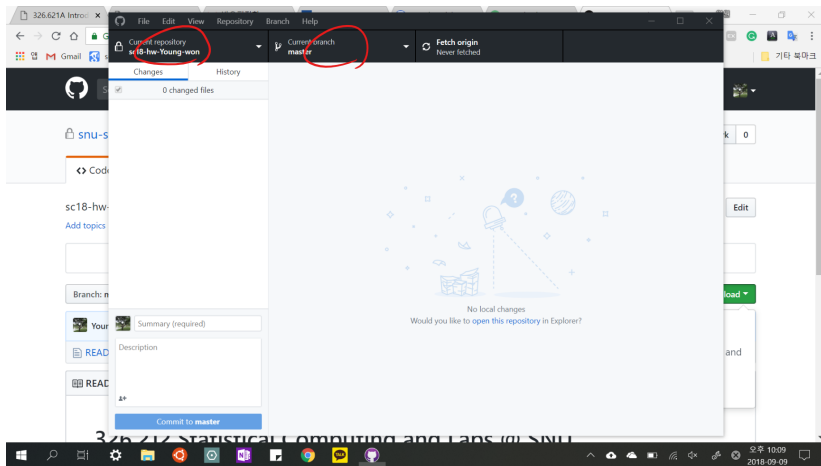
Github 속제 repository를 로컬 저장소에 복제 (Github Desktop 이용)



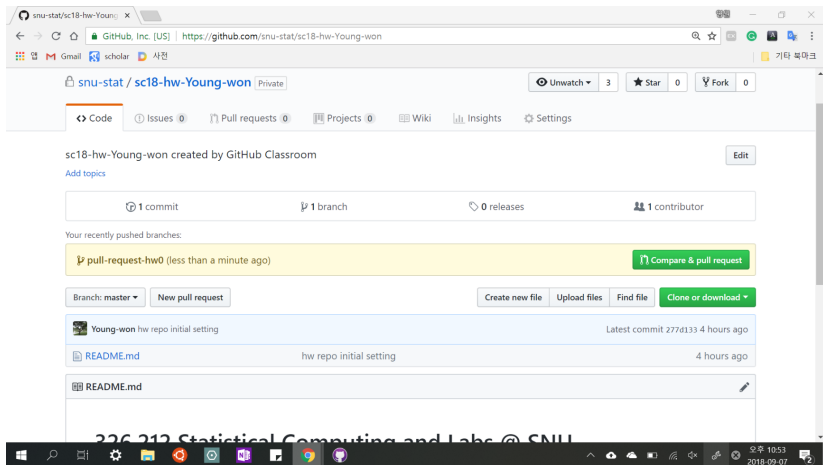
Github 속제 repository를 Github Desktop에서 열기



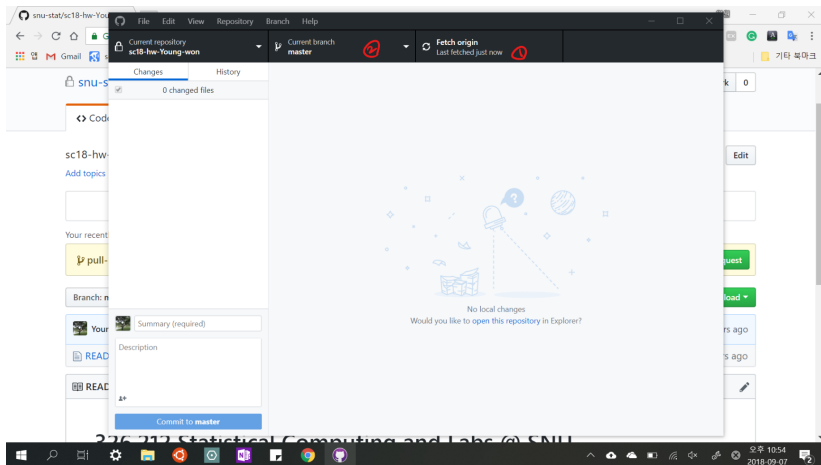
Github 속제 repository를 Github Desktop에서 열기 (Cont'd)



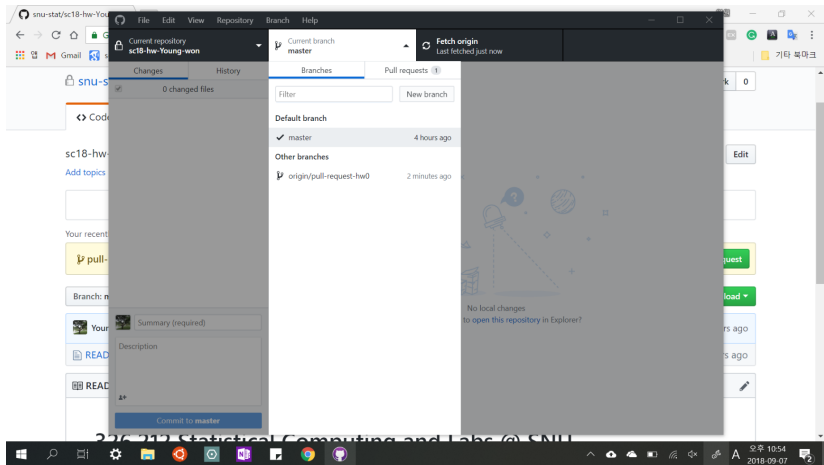
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기



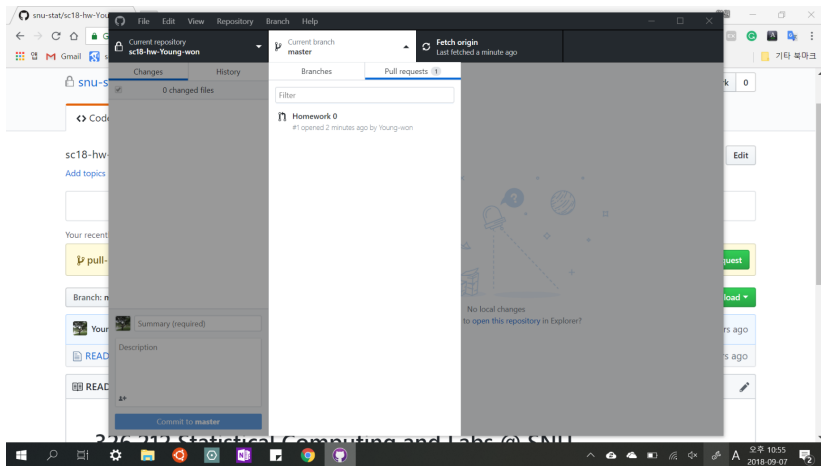
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기 (Cont'd)



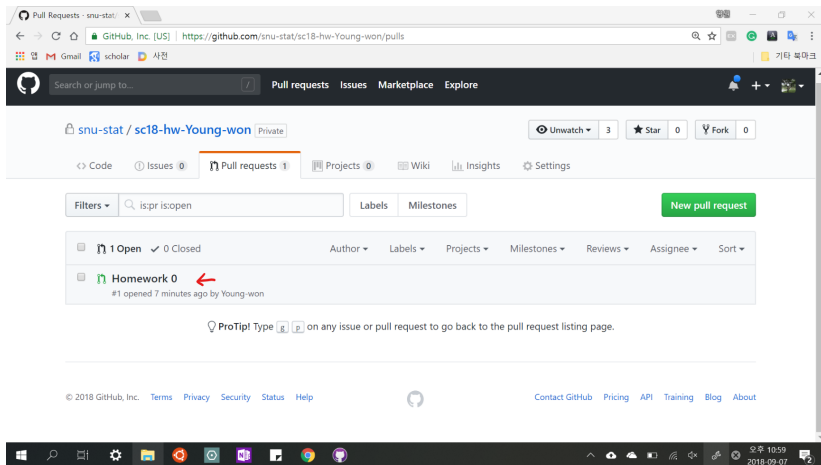
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기 (Cont'd)



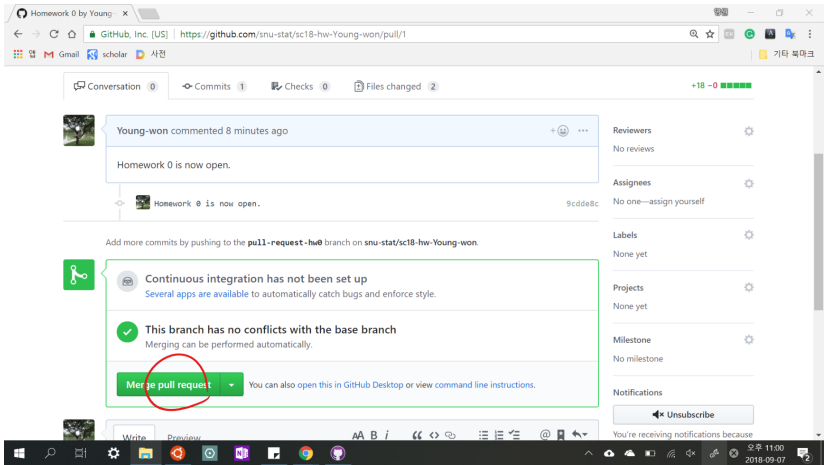
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기 (Cont'd)



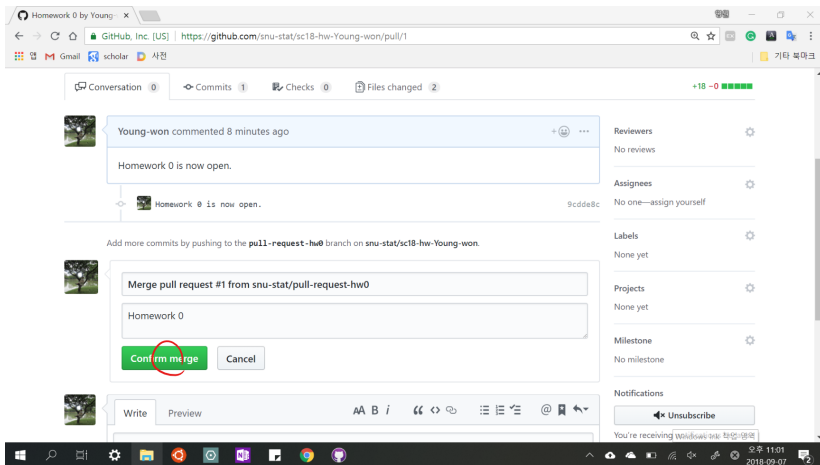
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기 (Cont'd)



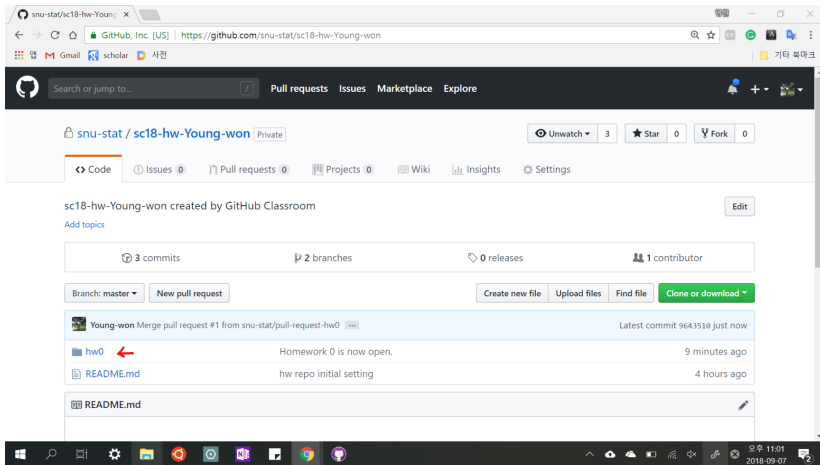
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기 (Cont'd)



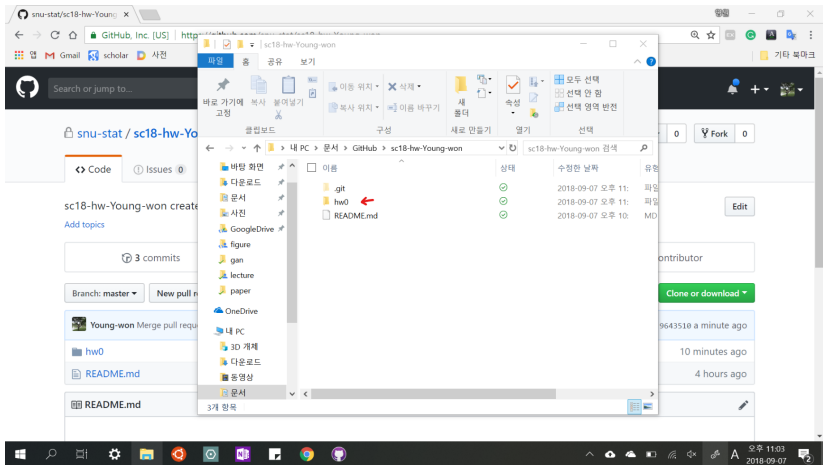
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기 (Cont'd)



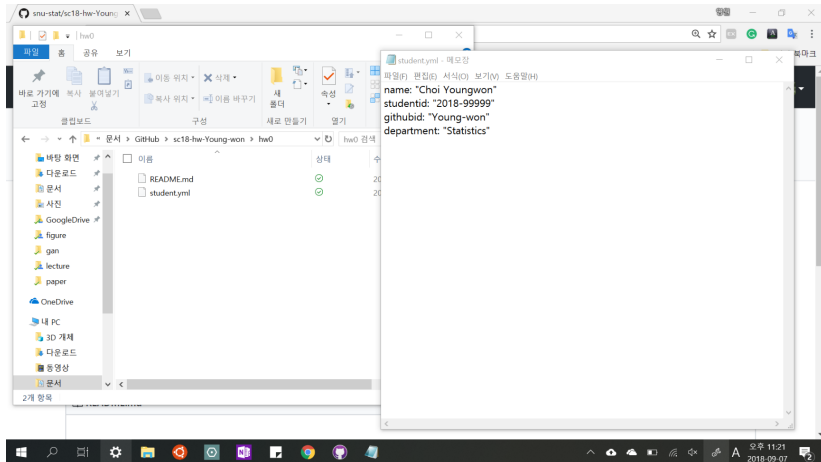
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기 (Cont'd)



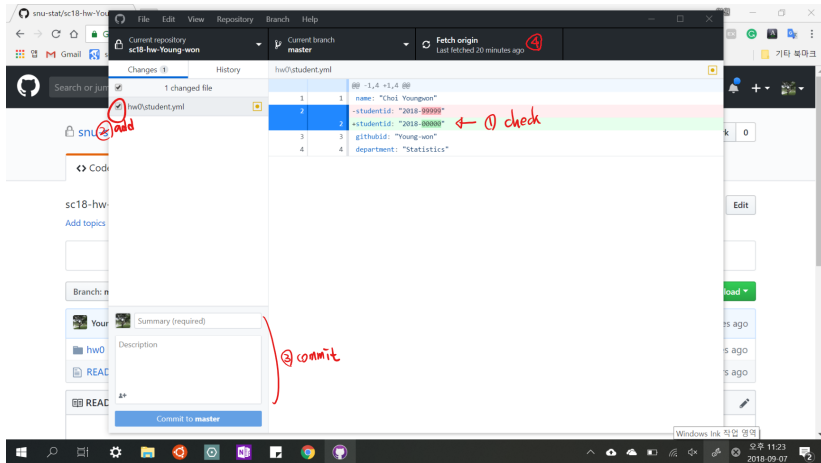
Github에 풀 리퀘스트로 올라온 숙제 병합 및 로컬 저장소에 반영하기 (Cont'd)



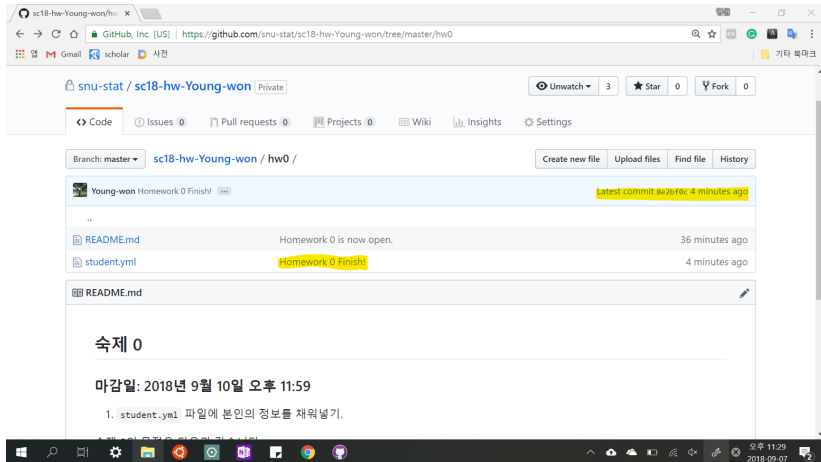
숙제 0 작업하기



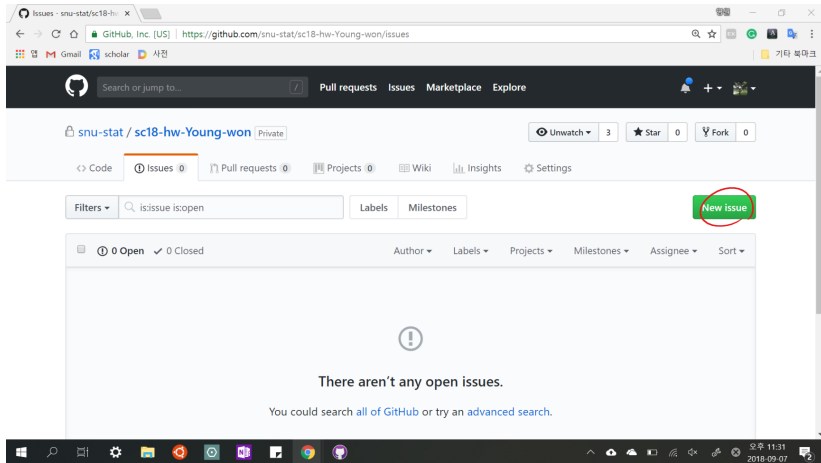
숙제 0 커밋 및 푸쉬하기



Github 변경 내역 확인하기



Github 에서 issue로 질문하기



Github 에서 issue로 질문하기 (Cont'd)

The screenshot shows a web browser window displaying a GitHub issue page. The address bar shows the URL: <https://github.com/snu-stat/sc18-hw-Young-won/issues/2>. The page title is "hw0 - githubid #2". Below the title, there is a green "Open" button with a red circle around it, and text indicating "Young-won opened this issue 2 minutes ago · 0 comments".

The main content area shows a comment from "Young-won" stating: "#1 hw0의 githubid가 무엇을 의미하나요? @snu-stat/young-won". Below the comment, there is a text input field with the placeholder "Leave a comment" and a "Comment" button. To the right of the comment input, there is a red circle with the number "1".

On the right side of the page, there are several settings sections: "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Notifications" (Unsubscribe).

The bottom of the page shows a Windows taskbar with various application icons and the system clock displaying "오후 11:35 2018-09-07".

Github 에서 issue로 질문하기 (Cont'd)

hw0 - githubid #2

Closed Young-won opened this issue 3 minutes ago · 1 comment

Young-won commented 2 minutes ago

#1 hw0의 githubid가 무엇을 의미하나요? @snu-stat/young-won

Young-won commented just now

github 내의 user id를 말합니다.

Young-won closed this just now

Assignees: No one—assign yourself

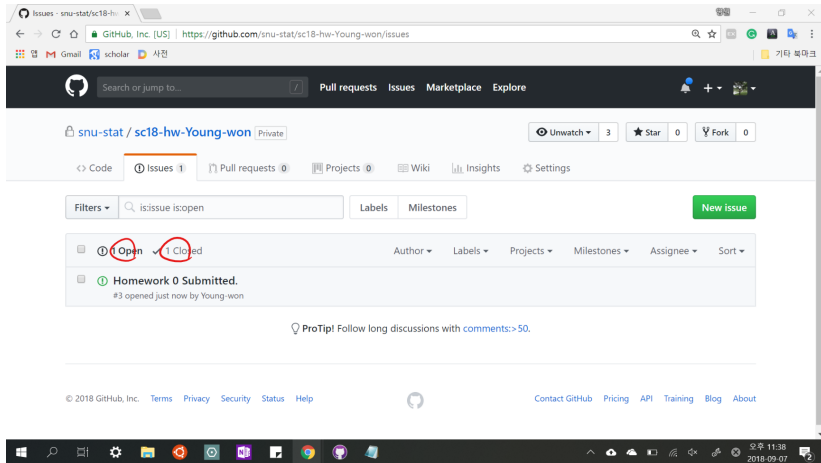
Labels: None yet

Projects: None yet

Milestone: No milestone

Notifications: Unsubscribe

Github 에서 issue로 숙제 제출 결과 확인하기



Github 에서 issue로 숙제 제출 결과 확인하기 (Cont'd)

The screenshot shows a web browser window displaying a GitHub issue page. The browser's address bar shows the URL: `https://github.com/snu-stat/sc18-hw-Young-won/issues/3`. The page header includes the GitHub logo, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header, the repository path is `snu-stat / sc18-hw-Young-won`, marked as Private. It shows 3 Unwatched issues, 0 Stars, and 0 Forks. The 'Issues' tab is selected, showing 1 issue. The issue title is 'Homework 0 Submitted. #3'. A green 'Open' button is next to the title. The issue was opened by 'Young-won' a minute ago and has 0 comments. A comment from 'Young-won' states: 'Your commit `8e2bf8c` submitted for grading.' Below this, there is a red handwritten note: 'check commit id'. The right sidebar shows 'Assignees' (No one—assign yourself), 'Labels' (None yet), and 'Projects' (None yet). The bottom of the screen shows a Windows taskbar with various application icons and the system clock showing 11:39 PM on 2018-09-07.

Github 에서 issue로 숙제 제출 결과 확인하기 (Cont'd)

The screenshot shows a GitHub web interface for a repository named 'snu-stat / sc18-hw-Young-won'. The commit being viewed is titled 'Homework 0 Finish!' and was made by 'Young-won' 15 minutes ago. The commit message is 'Finish!!!'. The diff shows a change to the file 'hw0/student.yml'. The original content (left) has a red background and shows a deletion of the 'studentid' field. The new content (right) has a green background and shows the addition of a new 'studentid' value. The diff is summarized as 'Showing 1 changed file with 1 addition and 1 deletion.' The bottom of the page shows the Windows taskbar with various application icons and the system clock indicating 11:40 AM on 2018-09-07.

Homework 0 Finish!

Finish!!!

master

Young-won committed 15 minutes ago

Showing 1 changed file with 1 addition and 1 deletion.

File	Line	Original	Modified
hw0/student.yml	2	- studentid: "2018-99999"	+ studentid: "2018-00000"

0 comments on commit 8e2bf0c

정리

- Git : 협업을 위한 분산 버전 관리 시스템
- GitHub : “Git” 위에서 만들어진 협업 플랫폼
- 작업 과정
 - ▶ 설정하기
 - ▶ 저장소 복제하기
 - ▶ 브랜치 선택하기
 - ▶ 로컬 파일 편집하기
 - ▶ 커밋
 - ▶ 동기화
 - ▶ (필요할 경우) 풀 리퀘스트 및 병합

참고자료

- Github Guides
- 완전 초보를 위한 깃허브
- GitHub Training Kit: GitHub for Everyone
- Git 관련
 - ▶ Pro Git 오픈 소스북, 3장 까지만 읽으면 기본적인 내용을 익힐 수 있다.
 - ▶ Try Git Code School과 GitHub에서 제공하는 튜토리얼
 - ▶ Git 입문자를 위한 가이드
 - ▶ 버전 관리를 들어본 적 없는 사람을 위한 DVCS - Git(SlideShare)

Q & A

감사합니다!