

Lab03: Git Tutorial

ESM2014-41

객체지향프로그래밍 및 실습

SKKU 시스템경영공학과

조영일

Git

- Git은 전 세계에서 가장 많이 사용되는 소스코드 버전 관리 도구(VCS, Version Control System)이다.
- 버전 관리 도구(VCS) : 버전 관리 시스템은 파일의 변화를 시간에 따라 기록하여 과거 특정 시점의 버전을 다시 불러올 수 있는 시스템이다.
- VCS를 사용하면 개별 파일 혹은 프로젝트 전체를 이전 상태로 되돌리거나 시간에 따른 변경 사항을 검토할 수 있다.
- 문제가 되는 부분을 누가 마지막으로 수정했는지, 누가 언제 이슈를 만들어냈는지 등을 알 수 있다.
- 또한 파일을 잃어버리거나 무언가 잘못되어도 대개 쉽게 복구할 수 있다.

Git

- 소스코드는 수시로 변경되지만, 변경이 누적되면 개발자 스스로도 그 변화 내역을 추적하기 어렵다.
- 또한, 여러명이 동시에 하나의 프로그래밍 프로젝트를 개발할 경우 소스코드의 변경은 더더욱 추적하기가 어려워진다.
- 따라서 규모가 큰 프로젝트 혹은 공동작업을 할 때 Git을 사용하는 것은 필수 요건

Setting Git

- 최신 버전의 Git을 다운로드 하여 설치 : <https://git-scm.com/downloads>
- Git 에서 사용할 Username과 Email 설정
 - 시작 > “cmd” 실행
 - `git config --global user.name "영문이름"`
 - `git config --global user.email "example@skku.edu"`(Github 계정과 동일한 이메일 주소)
 - `git config --global credential.helper wincred`

Git Init

- Git Repository : Git을 통해 버전 관리되는 하나의 디렉토리(프로젝트를 의미함)
- git init : 특정 디렉토리에서 해당 명령어를 실행할 경우 그 디렉토리를 Git Repository로 초기화함

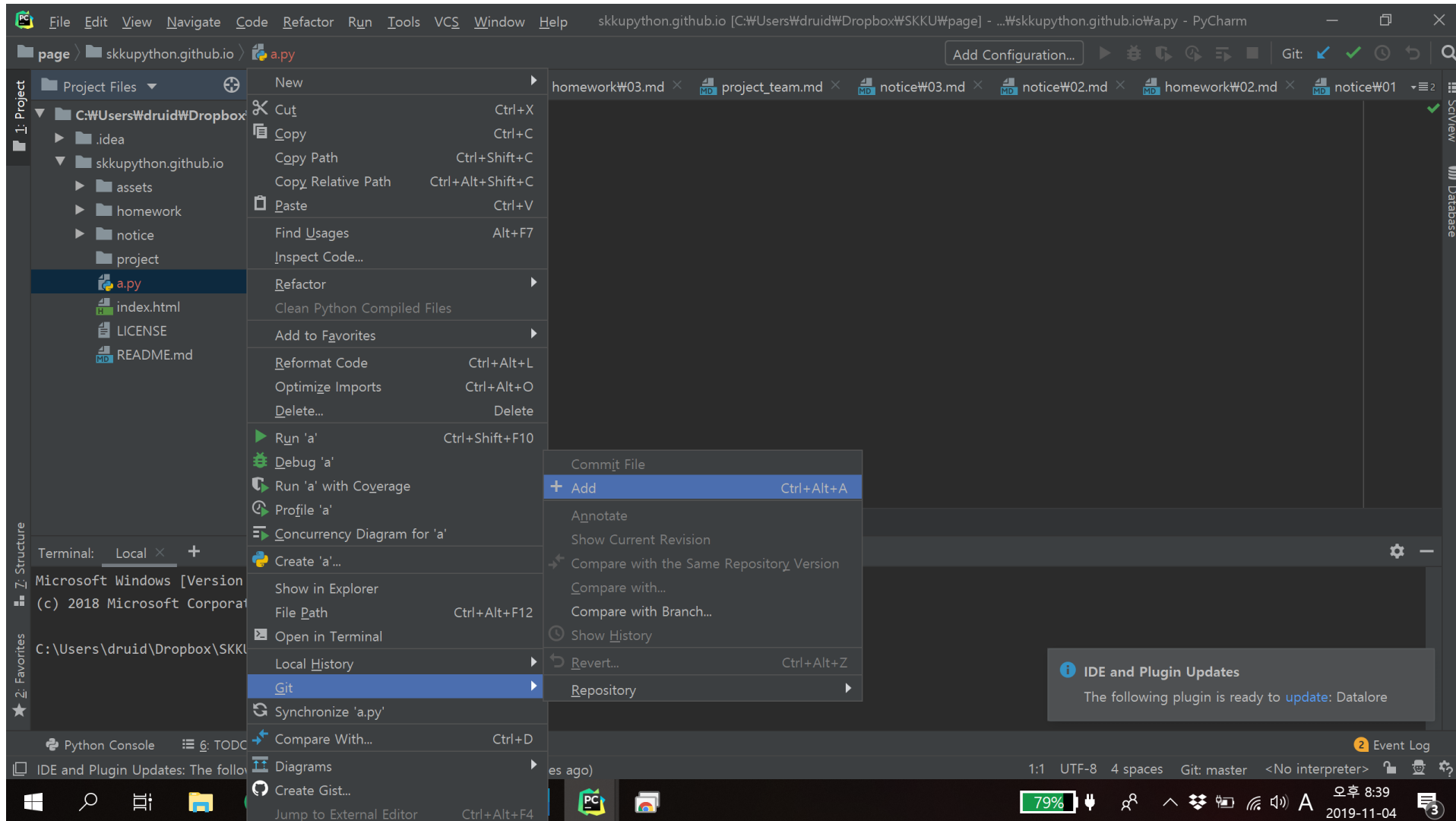
Git Add & Commit

- `git add` : Git repository에서 변경된 파일을 staging 상태(임시 변경된 상태)로 변경
 - `Git add *` : 디렉토리 내의 모든 파일을 staging 상태로 변경한다.
 - `Git add [파일경로/파일명]` : 디렉토리 내에서 특정 파일만을 staging 상태로 변경한다.
- `git commit` : staging 상태의 모든 변경을 확정하여 Repository에 반영한다.
 - `git commit -m` “변경한 내용에 대한 메모”

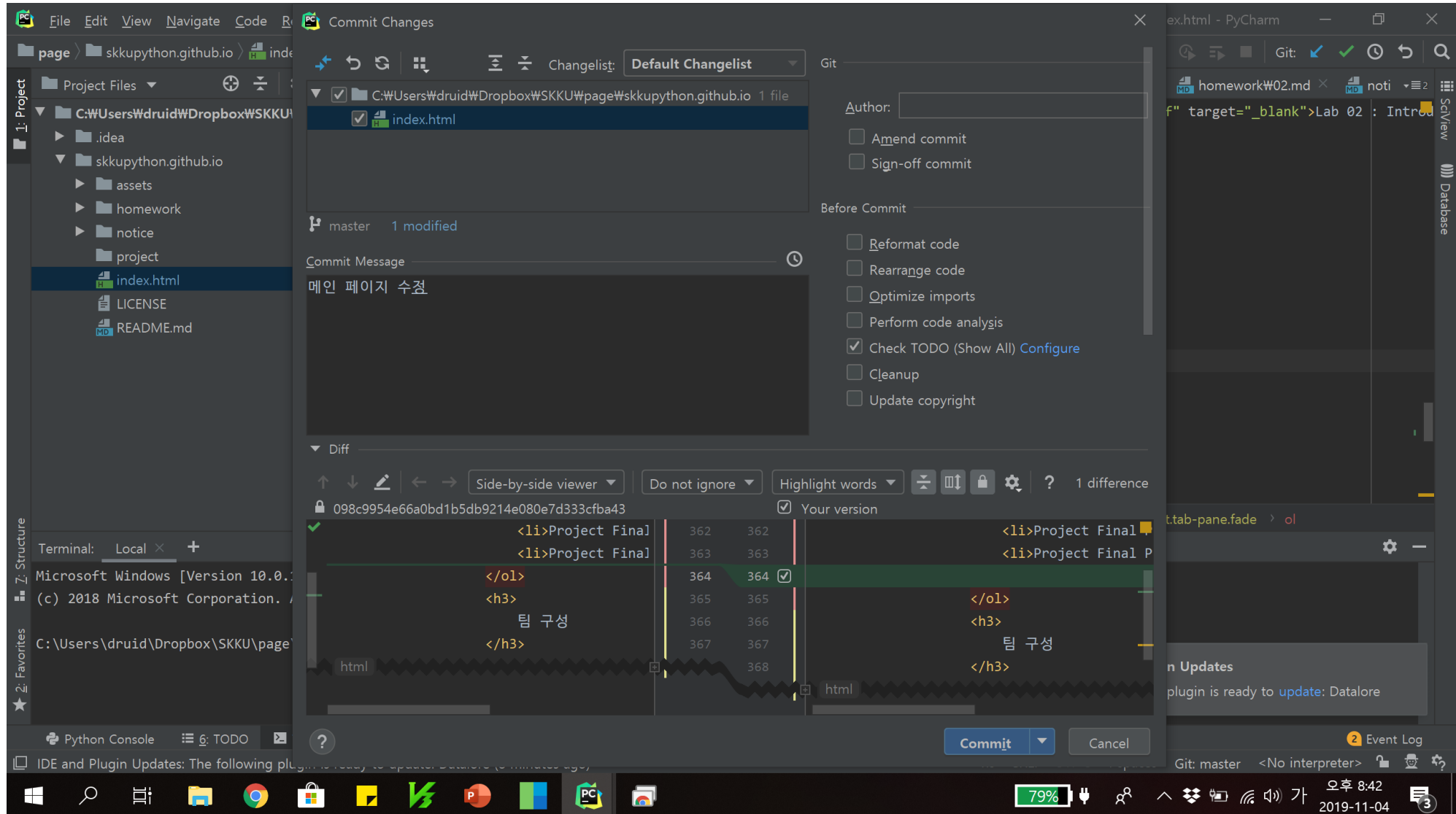
Git Add & Commit(PyCharm)

- PyCharm에는 VCS에 대한 지원도 기본 포함되어 있음
- PyCharm에서 편집하는 프로젝트가 Git Repository일 때: PyCharm VCS 기능을 통해 Add & Commit 가능


Git Add & Commit(PyCharm)



Git Add & Commit(PyCharm)

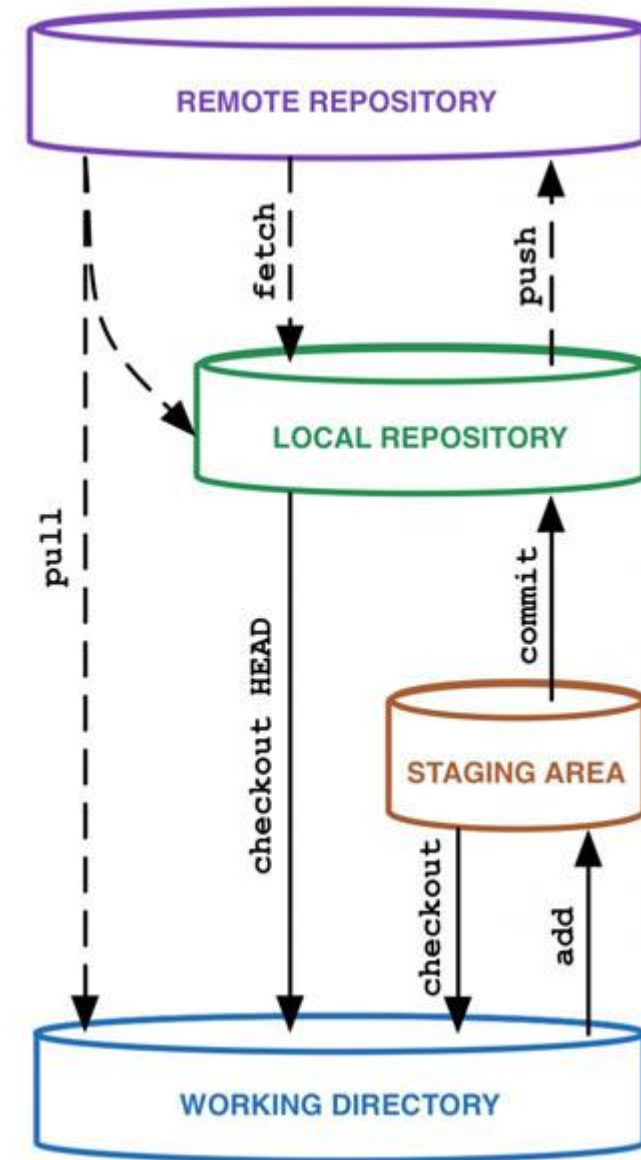
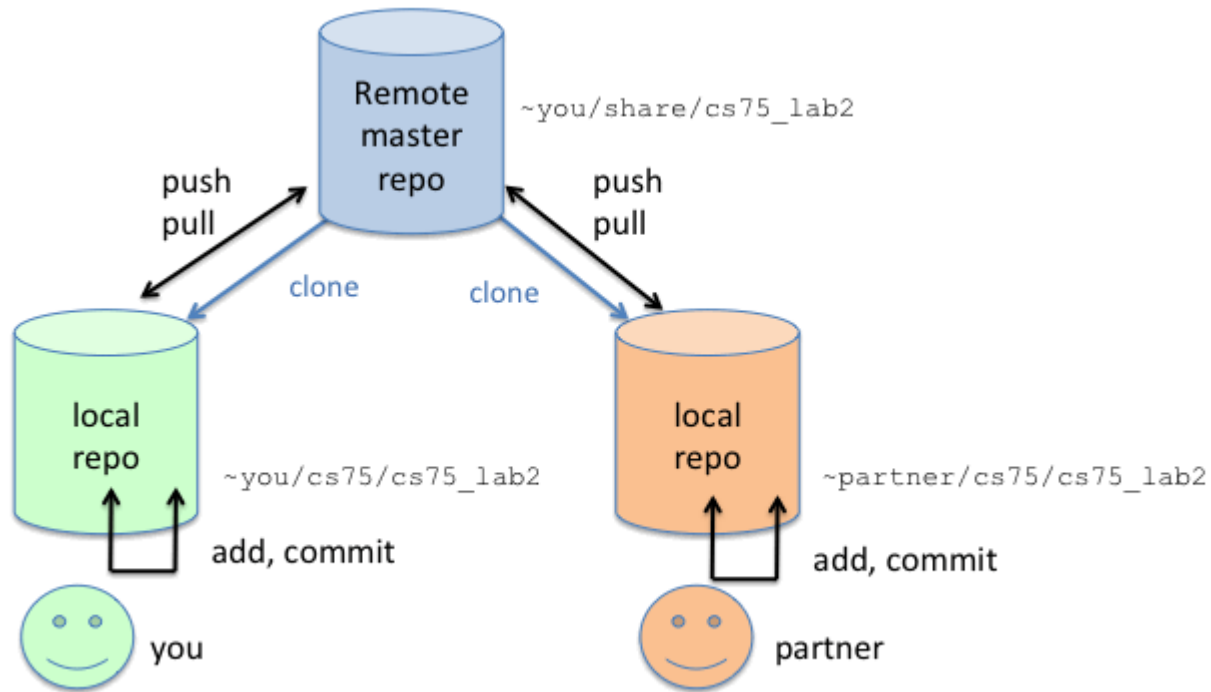


Git Commit History

인테이크 마이페이지 주문 조회에서 인테이크 제품 정보가 없는 제품이 포함된 주문의 주문 상세를 조회할수없던 문제 수정 youngilcho committed on 10 Jul 2017	 029eb76 
Merge branch 'master' of https://bitbucket.org/intakefoods/intakescm  Dong Hwi Jung authored and Dong Hwi Jung committed on 10 Jul 2017	 3c8ee41 
공급사 라이브스토어 제품 등록 시 카테고리 설정 관련 오류 수정  Dong Hwi Jung authored and Dong Hwi Jung committed on 10 Jul 2017	 e300882 
멤버십 쿠폰 만료 시각 변경 youngilcho committed on 10 Jul 2017	 3350f94 
라이브스토어 일괄 입력기능 개발 youngilcho committed on 10 Jul 2017	 c3be091 
인테이크 하단 푸터 연락처 추가 youngilcho committed on 10 Jul 2017	 d376ce9 

- 한번 Commit을 할 때마다 모든 파일/코드의 변경 내역과 Commit Message, 변경한 사용자가 함께 기록됨
- <https://github.com/skkuppython/skkuppython.github.io/commits/master>

Github

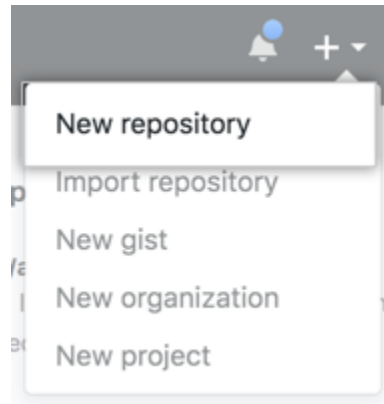


Github



- 세계 최대의 Git Remote Repository 서비스
- 공동작업을 위한 무료 Git Remote Repository를 제공함
- Github에 Remote Repository를 생성함으로써 다수의 사용자가 공동 코드 작성 작업을 할 수 있음.

Github Repository 생성



- Github에 로그인 한 이후 아무 페이지에서나 우측 상단 +버튼 클릭 > “New Repository” 클릭

Github Repository 생성

Create a new repository

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

Owner



Repository name

hello-world



Great repository names are short and memorable. Need inspiration? How about **potential-eureka**.

Description (optional)

Owner



Repository name

hello-world



Great repository names are short and memorable. Need inspiration? How about **potential-eureka**.

Description (optional)

My first repository on GitHub

☒ **Public**

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

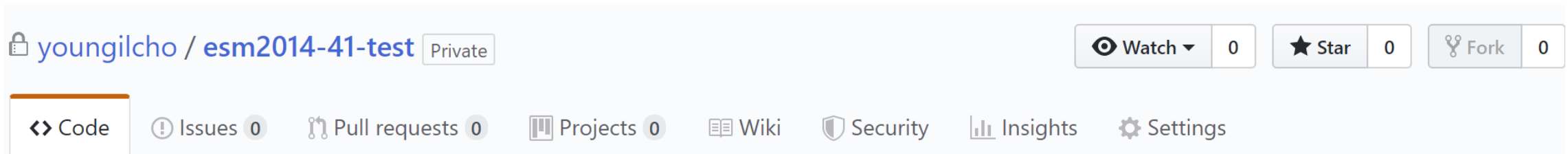
☐ **Private**

You choose who can see and commit to this repository.


- Repository 이름 설정(프로젝트 명)
- 공개 여부 설정 : Public – Github 사용자라면 누구든지 프로젝트 코드를 볼 수 있음
/ Private – 생성자와 초대된 Github 사용자만이 프로젝트 코드를 볼 수 있음
- “Create Repository” 클릭

Github Repository Clone

- `git clone` : Remote Repository(Github)의 내용을 그대로 local 로 가져오는 명령어
- 프로젝트 파일을 가져오고자 하는 위치에서 아래 명령어를 입력할 경우 Remote Repository 의 내용을 Local 로 복제함
- `git clone` “저장소 주소”



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

 Set up in Desktop

or

HTTPS

SSH

`https://github.com/youngilcho/esm2014-41-test.git`



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

Push an existing repository to Github

- 이미 Local 에 존재하는 Git Repository를 생성된 Github Repository로 전송하고자 하는 경우
- `Git remote add origin “저장소 주소”`
- `git push -u origin master`(Local Repository의 Commit 내역을 Github Repository로 전송함)

Git Conflict

- Git은 대부분의 경우에 같은 파일에 대한 여러 작업자의 수정이 있을 경우에도 알아서 코드를 합쳐 줌
- 일부 상황에서 공동작업자가 나와 같은 파일, 같은 부분을 수정 하였을 경우 : 내 Local 의 Commit 내역과 Remote Repository에 존재하는 Commit 내역이 서로 충돌하는 상황 발생
- 이러한 상황을 코드간 충돌(Conflict)이라고 부르며, 충돌이 발생하였을 경우 반드시 직접 수정 후 다시 Commit 하여야 다음 Commit을 진행 할 수 있음

Git Conflict

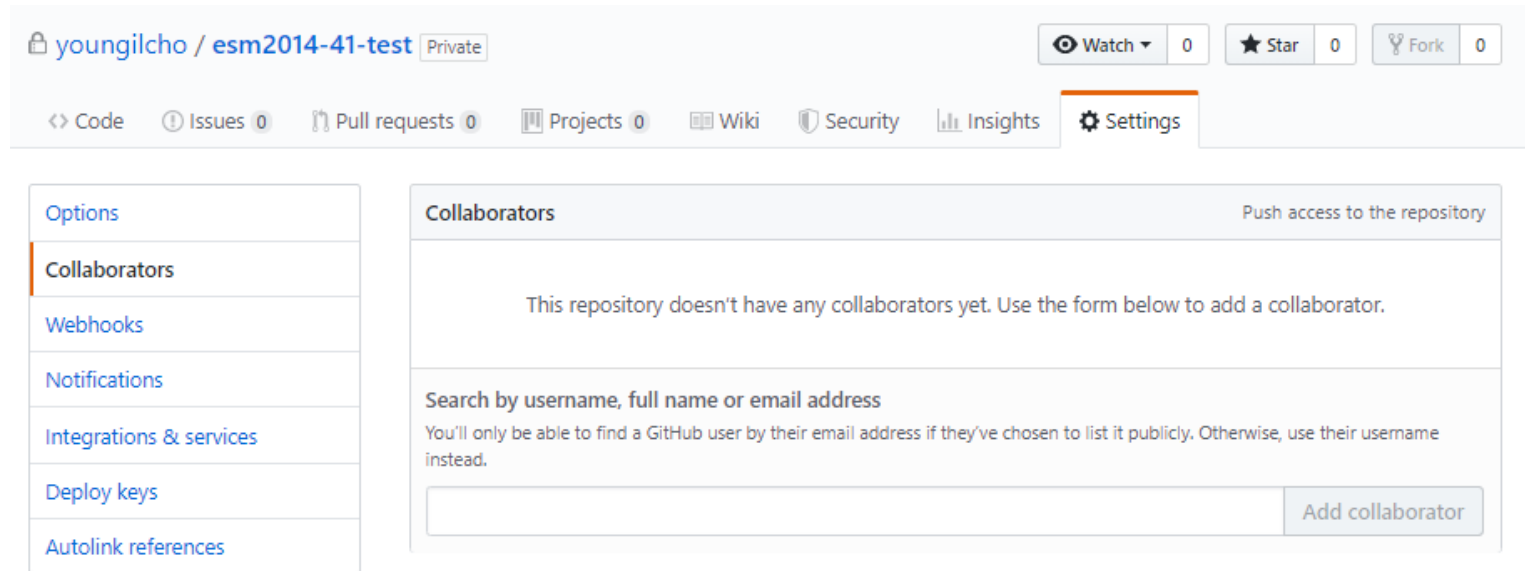
RD Merge Revisions for D:\Projects\Git\Demo\Nancy\src\Nancy\DependencyContextAssemblyCatalog.cs

5 changes. 1 conflict

Local Changes (Read-only)	Result	Changes from Server (revision 9d4b6795...)
using System.Reflection; using Microsoft.Extensions...	7 7 8 8	using System.Reflection; using Microsoft.Extensions...
/// <summary> /// Default implementation /// retrieving <see cref="IReadOnlyCollection{T}> /// </summary> public class DependencyContext	9 9 10 10 11 11 12 12 13 13 14 14	/// <summary> /// Default implementation /// retrieving <see cref="IReadOnlyCollection{T}> /// </summary> public class DependencyContext
{ private static readonly DependencyContext	15 15 16 16 17 17 18 18	{ private static readonly DependencyContext
/// <summary> /// Initializes a new instance of the <code>DependencyContext</code> class. /// using <see cref="IReadOnlyCollection{T}> /// </summary> public DependencyContext	19 19 20 20 21 21 22 22 23 23	/// <summary> /// Initializes a new instance of the <code>DependencyContext</code> class. /// </summary> public DependencyContext
: this(AssemblyName)	24 24	: this(AssemblyName)
{ }	25 25 26 26	{ }
/// <summary> /// Initializes a new instance of the <code>DependencyContext</code> class. /// using <paramref="T"> /// </summary> public DependencyContext	27 27 28 28 29 29 30 30 31 31	/// <summary> /// Initializes a new instance of the <code>DependencyContext</code> class. /// using <paramref="T"> /// </summary> public DependencyContext
{ dependencyContext	32 32 33 33 34 34	{ this.depender
}	35 35	}

Accept Left Accept Right Apply Abort

Github Collaborator



- Github Private Repository일 경우 Settings > Collaborators 메뉴를 통해 Collaborator를 추가해야 공동 작업자 역시 git push 가능
- Free Plan일 경우 Collaborator 3명 제한이 있으므로, 반드시 Github Student Developer Pack 등록 필요
- Github ID(가입 당시 사용한 Email)을 통해서 추가 가능

Git을 이용한 일반적인 공동 작업 Flow

1. 작업 시작 전 “git pull origin master”(Github Repository)에 반영된 최신 Commit 내역을 가져옴
2. git conflict가 존재 할 경우 PyCharm 등의 에디터를 통해 코드를 비교 후 Conflict 부분을 해결 이후 다시 Commit
3. 코드 수정 작업 진행
4. 하나의 단일 기능을 완성 할 때마다 Commit
5. git push origin master(Local 의 Commit 내역을 Github Repository에 반영)

.gitignore

- git repository에는 프로젝트 자체와 관련된 소스코드, Static Resource 만을 반영하는 것이 원칙
- 내 컴퓨터(Local)에서만 사용되는 파일은 git 으로 관리하면 안된다 : 공동작업자의 환경에 영향을 줄 가능성
- e.g. 프로젝트 생성시 IDE(PyCharm)에 의해 생성되는 파일들 - 내 컴퓨터에 한정된 정보를 담고 있다.(편집창의 레이아웃 등)
- .gitignore : Project Root에 “.gitignore”라는 파일을 생성하고 해당 파일 내에 프로젝트에서 git으로 관리하지 않을 파일의 목록을 작성해두면 git은 해당 파일들을 add/commit 단계에서 추가하지 않는다.
- PyCharm용 .gitignore : <http://gitignore.io/api/pycharm+all>

다음시간까지 할 일

1. 각 팀별로 Github Repository 생성
(팀별 1인이 대표 진행 Repository 명칭 - esm201441-2019-teamN / e.g. esm201441-2019-team1)
(반드시 Private Repository로 생성할 것)
2. Repository에 PyCharm용 .gitignore add, commit : <http://gitignore.io/api/pycharm+all>
3. “Django Tutorial I”에 따라서 Tutorial Project 생성 이후 Commit
4. Collaborator 에 팀원 및 다음 아이디 추가(“druidcho@gmail.com”)
5. 각 팀원들은 해당 초대 받은 저장소를 본인 컴퓨터로 git clone 하고 PyCharm 으로 프로젝트 열어 코드 편집 가능한 상태로 만들기
6. 코드 수정 이후 Commit & Push & Pull 작업 진행 까지 연습

Study Topic

- Getting started with GitHub(<https://help.github.com/en/github/getting-started-with-github>)
- Git 간편 안내서(<https://rogerdudler.github.io/git-guide/index.ko.html>)
- Git 자세히 알아보기(<https://git-scm.com/book/ko/v2>)