

프로젝트 실행 서비스 환경 구축

프로젝트 조원끼리 소스 공유

- 1> 소스 코드 오류 발생 직전 되돌릴 수 있다
- 2> 소스 코드 개발 버전 관리할 수 있다
- 3> 소스 공유 가능하다

버전 관리의 개념

## 버전 관리 시스템 (Version Control System : VCS)

- 조직의 핵심 자산인 소스 코드의 개정과 백업 절차를 자동화하여 오류 수정 과정을 도와줄 수 있는 시스템
- 시스템 개발 중 어떤 의미 있는 변화들을 관리하는 체계 (일련의 개발 관리 활동)
- 의미 있는 변화들
  - 기능 개선, 오류 수정, 고객 요구사항 변경 등에 의한 소스코드 또는 산출물의 변경
- 수행 기능
  - 여러 개발자들이 협업하며 코드를 작성하고 관리할 수 있는 기능
  - 소스코드의 백업 및 이전 버전으로의 롤백 기능 등

## 형상 관리 도구 (1)

소스코드를 작성하거나 관리하는 모든 개발자뿐 아니라 디자이너, 기획자 등 파일에 대한 이력 관리가 필요한 경우에는 반드시 필요한 도구

### 상용 (유료)

- IBM Rational ClearCase
- Perforce
- PTC Integrity

### 오픈소스(무료)

- Subversion(SVN)
- CVS
- Mercurial
- Git

#### ■ Git – 2일 도스 코맨드 형태

- 프로그램 등의 소스 코드 관리를 위한 분산 버전 관리 시스템
- 여러 명의 개발자가 특정 프로젝트를 자신의 컴퓨터로 협업하여 개발하면서 버전을 관리할 수 있는 시스템
- Git의 버전 관리 방식
- 중앙 서버 컴퓨터와 여러 개의 컴퓨터들이 연결되어 모두 같은 버전의 데이터베이스 유지
- 업데이트될 때마다 최신 버전 자동으로 생성
- 파일들이 최신 버전으로 모든 컴퓨터에서 유지
- Github를 통해 웹 브라우저를 사용해서 누구나 Repository를 만들

어 사용할 수 있음

- 각 개발자들이 진행한 개발 변경 사항을 온라인에서 확인 가능한 서비스
- 코드를 전체 복사해서 독립적으로 개발하는 것 (자신만의 버전)

#### ■ Revision

- 저장소에 저장된 각 파일 버전
- 소스 파일을 수정하여 커밋하게 되면 일정한 규칙에 의해 숫자가 증가함

## Repository

- Local repository : 내 컴퓨터의 저장소
- Remote repository : Github 저장소

### add

- Stage에 올리는 명령. commit이 가능한 상태로 만드는 것

**commit** : 변경된 내용 확정

**push** : 원격 저장소로 업로드

### pull

- 원격 저장소로부터 필요한 파일을 로컬 저장소로 다운로드 후 Merge

## branch (가지)

- Root 프로젝트로 부터 파생된 프로젝트
- 코드를 전체 복사해서 독립적으로 개발하는 것 (자신만의 버전)
- 새로운 작업 테스트 시 사용하는 영역
- 독립된 working directory

## HEAD

- 현재 작업 중인 branch를 가리키는 포인터
- 현재 branch의 마지막 commit된 snapshot

## checkout

- 다른 branch로 이동
- 체크하기 원하는 저장소로 이동할 수 있는 탐색 명령

github와 sts 또는 eclipse 연동

-github 회원가입과 원격저장소 생성-

-github 인증-토큰 생성

-내 프로젝트 원격저장소에 올리기-

gitupload

-원격저장소에서 프로젝트 가져오기-

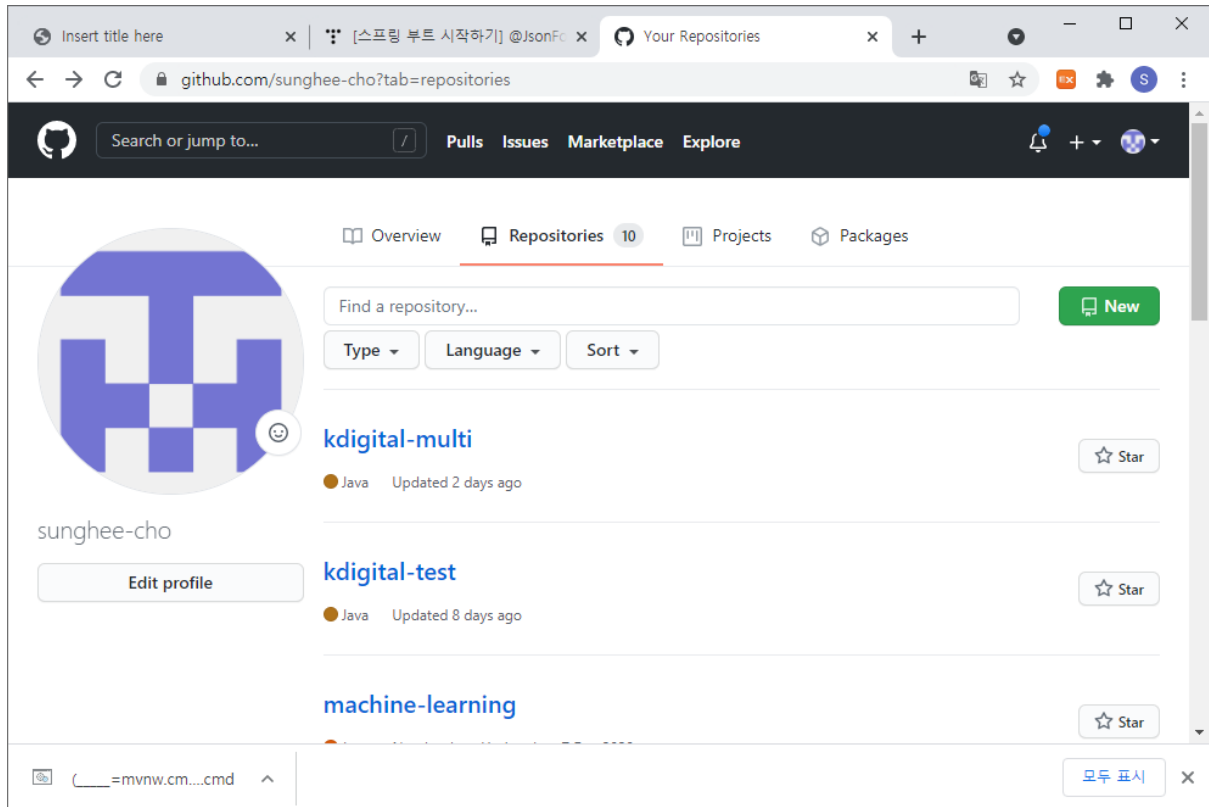
-변경 사항 만들기-

-github 회원가입과 원격저장소 생성-

1. github에 회원 가입한다.(sign up)

2. github에 로그인한다(sign in)

3. repository를 새롭게 생성한다. 오른쪽 new 버튼을 클릭한다.





4. private 선택시 무료 플랜으로 만든 비공개 저장소에서는 협업이 3명까지만 가능.

## Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*      Repository name \*

 sunghee-cho ▾ / kdigital-recent 

Great repository names are short and memorable. Need inspiration? How about **redesigned-guacamole**?


Description (optional)

- ☐  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☒  **Private**  
You choose who can see and commit to this repository.

### Initialize this repository with:

Skip this step if you're importing an existing repository.

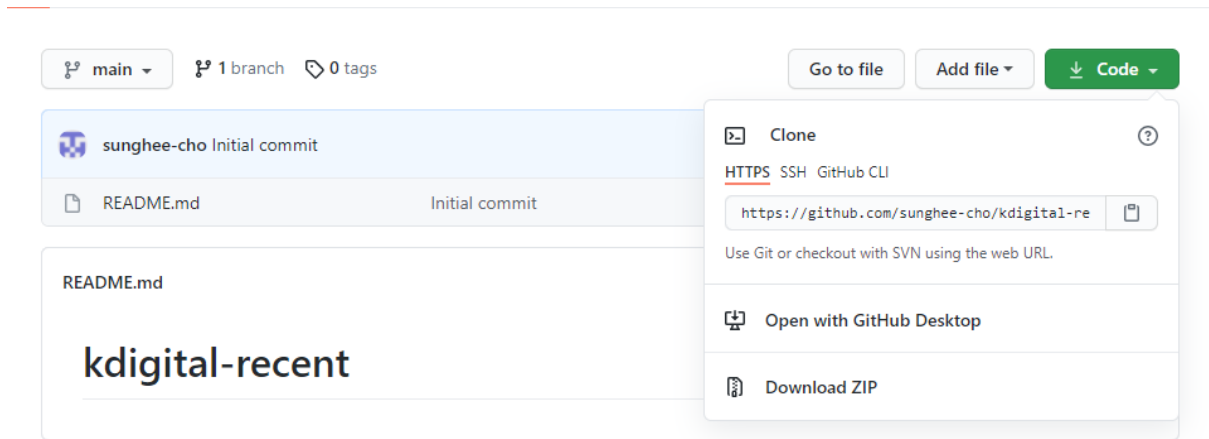
- ☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)
- ☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)
- ☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  `main` as the default branch. Change the default name in your [settings](#).

Create repository

5. 삭제시에는 첫화면에서 레포지토리 선택하고 settings 메뉴 선택 , 화면 하단으로 이동하면 danger zone에서 delete this directory를 이용한다.

6. 코드를 눌러 url를 복사한다.



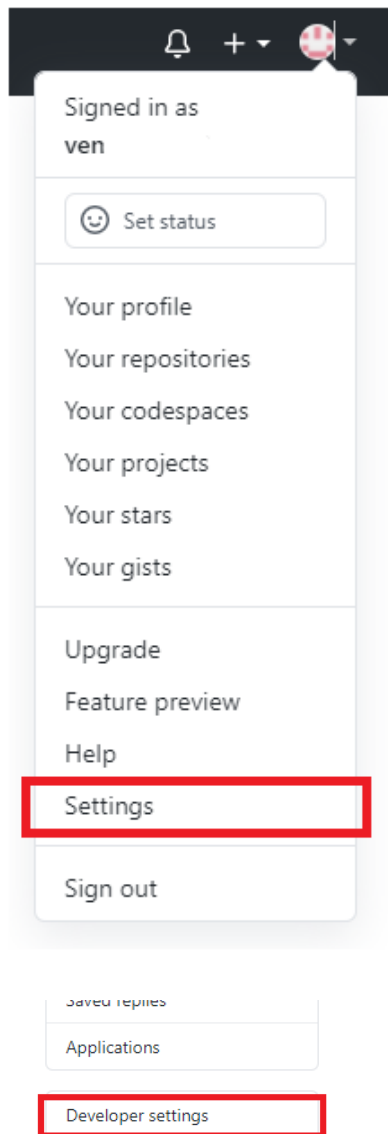
GitHub 연동 후 문제없이 사용하고 있었는데 갑자기 Commit, Push, Pull..  
아래와 같은 에러 메시지가 나왔다.

결론은 2021.08.13일부터

GitHub에서 ID/Password 인증을 없애고  
ID/Personal Access Token 방식의 Token 인증방식을 요구한다.

#Jenkins + GitHub 연동 시에도 동일하다.

GitHub 홈페이지 > Github Signed(오른쪽 상단) > Settings



Personal access tokens > Generate new token

Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

Personal access tokens

Generate new token

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).



## New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

토큰이름

What's this token for?

Expiration\*

유효기간

token will never expire!

Beep bop! Tokens that live forever are scary. Expiration dates are highly recommended!

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows

생성된 token을 저장해둔다. (최초에만 확인 가능, 나중에 다시 확인할 수 없음)

## Personal access tokens

[Generate new token](#)[Revoke all](#)

tokens you have generated that can be used to access the [GitHub API](#).

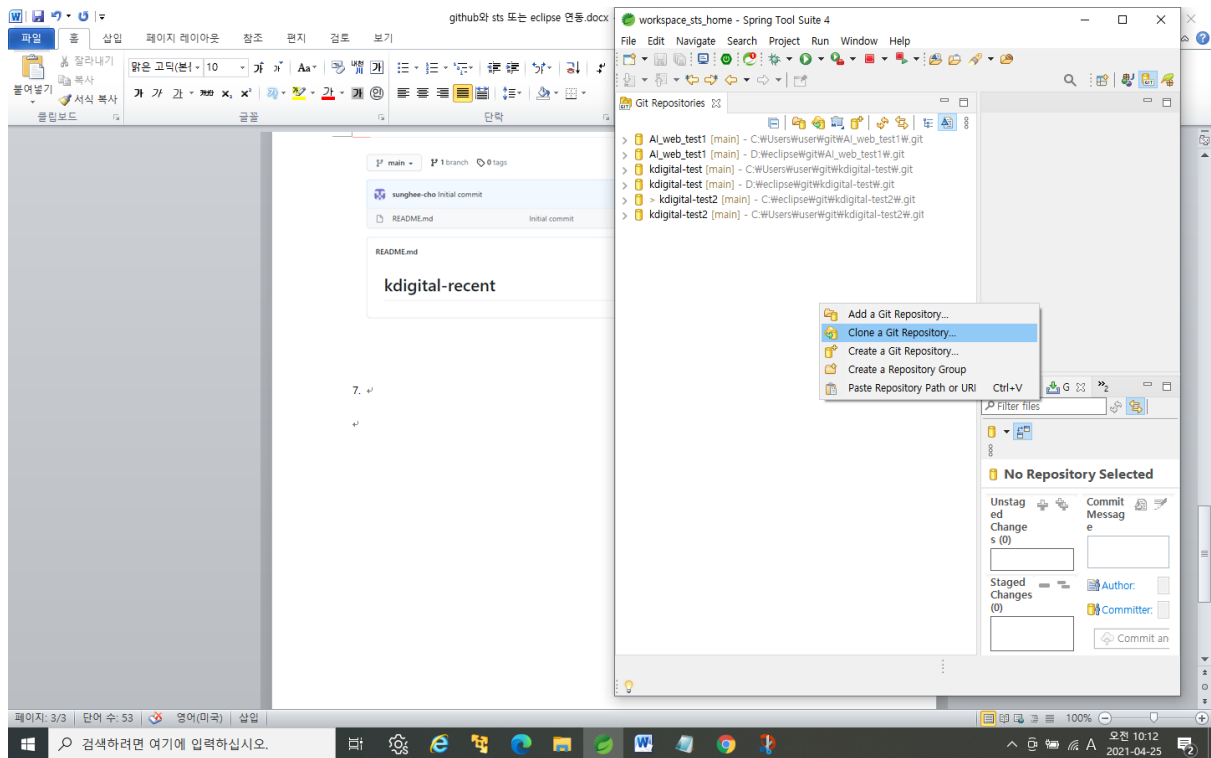
Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp\_ 

Delete

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

7. sts를 시작한다. window-perspective-git를 선택하여 project explorer 에서 마우스 우클릭한다. clone a git repository를 선택한다.



8. uri 붙여넣기, user와 password에 깃허브 정보를 입력한다

Clone Git Repository

**Source Git Repository**  
Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:


Password:

☐ Store in Secure Store

password에는 깃헙 토큰을 입력한다.


9. 다음 화면에서 main 선택한다.

10. 로컬 저장소를 지정한다.

 Clone Git Repository

**Local Destination**

Configure the local storage location for kdigital-recent.



**Destination**

Directory:

Initial branch:

☐ Clone submodules

**Configuration**

Remote name:


**Projects**

☐ Import all existing Eclipse projects after clone finishes

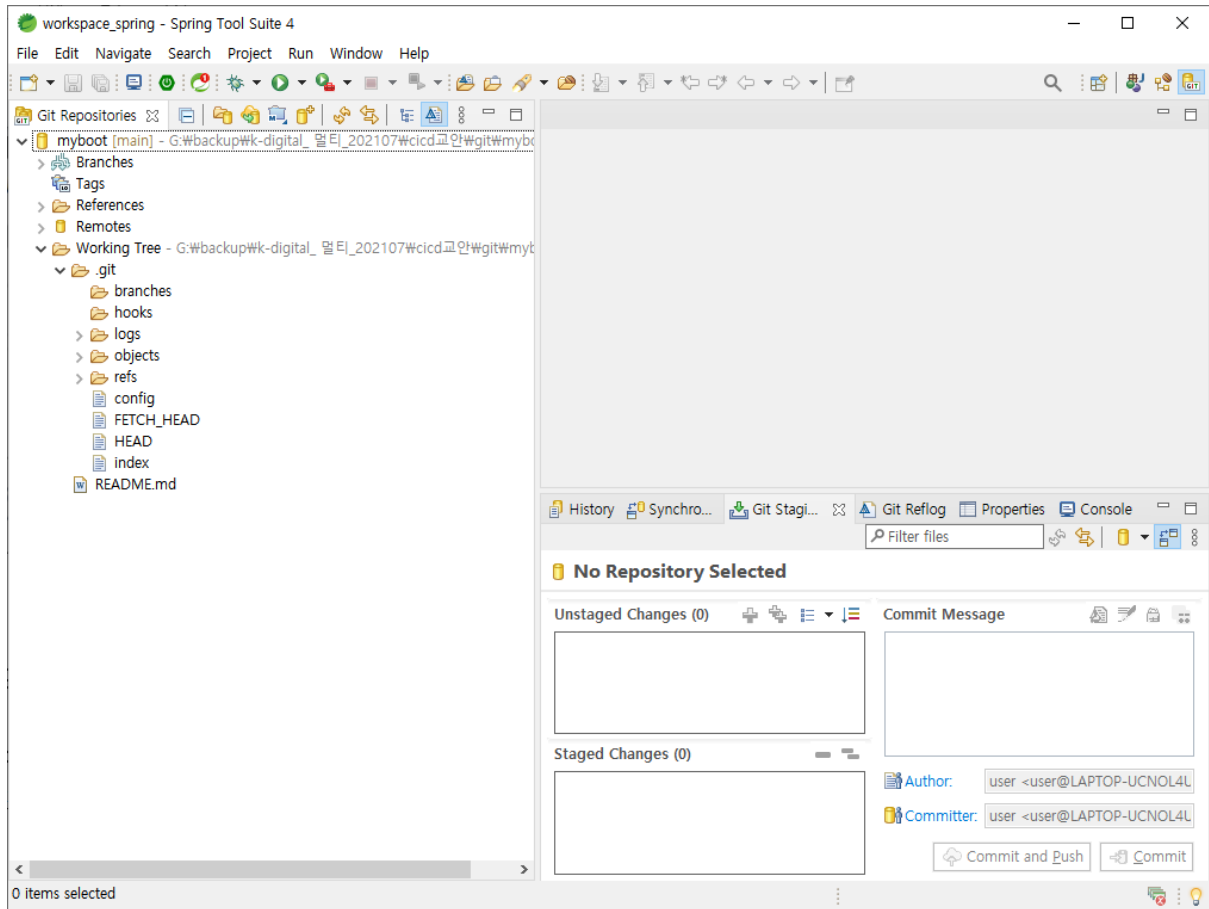
**Working sets**

☐ Add project to working sets

Working sets:

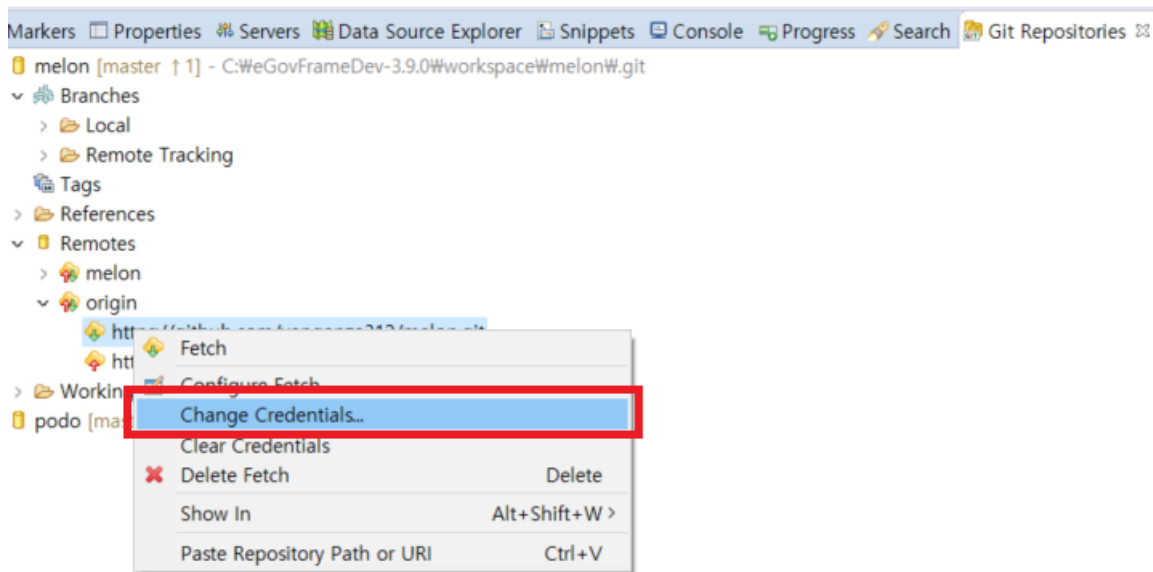


내 로컬 저장소 화면이다.



필요시 git perspective에서 다음처럼 change credential을 실행한다.

Git Repositories > Remotes > origin > 해당 Github 우 클릭 > Change Credentials...

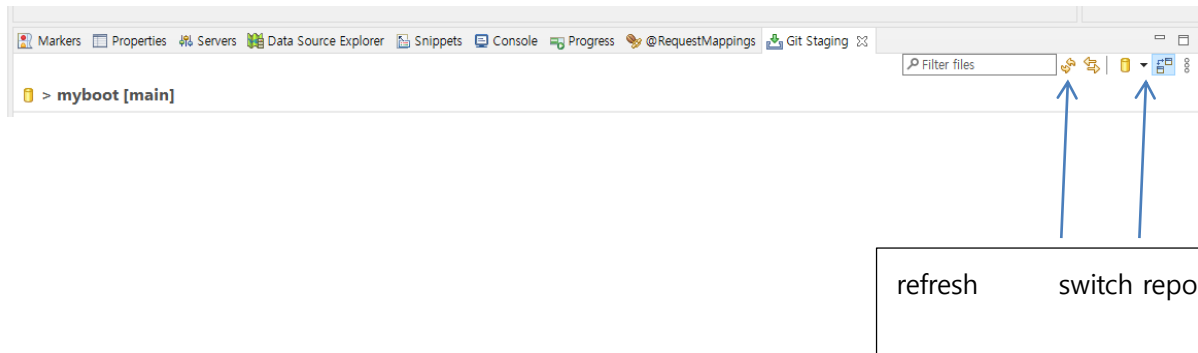


working tree에는 원격 저장소에 파일이 업로드되어 있다면 그 프로젝트도 보일 것이다.(현재는 없다)

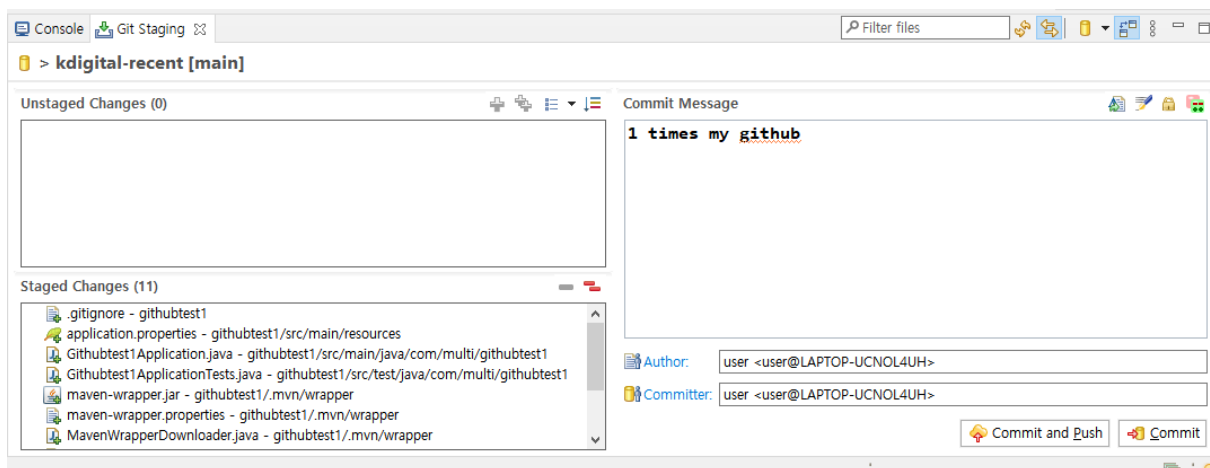
### -내 프로젝트 원격저장소에 올리기-

1. githubtest1 이라는 이름으로 스프링 부트 프로젝트를 생성한다. 만약을 위해 중요한 파일이 있는 프로젝트는 테스트용으로 올리지 않는다.
2. 프로젝트 선택 - 마우스 우클릭 - team - share project 하면 프로젝트와 파일앞에 > , ? 모양들이 생긴다.
3. 프로젝트 선택 - 마우스 우클릭 - team -add to index 한다. 이제 파일들은 github 서버에 올리기가 준비가 된다.

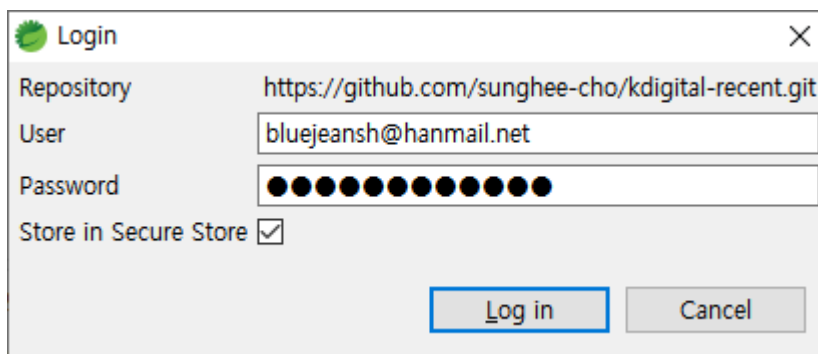
(add to index 해도 변화없으면 git staging에서 switch repository 선택 후 refresh한다.)



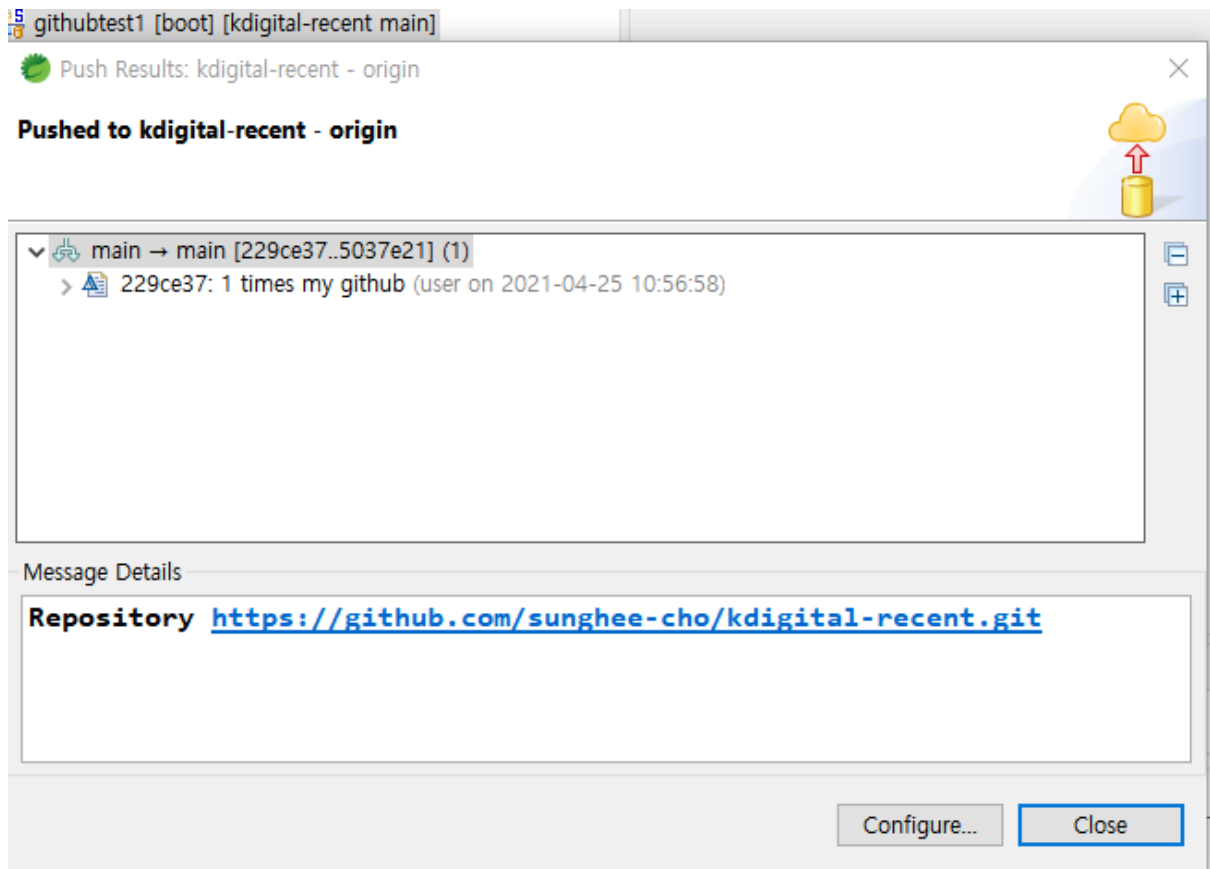
4. 프로젝트 선택 – 마우스 우클릭 – team –commit... 을 클릭한다. staged changes 에 파일 목록 출력된다. commit message는 직접 입력한다. commit and push 클릭한다.



5. 깃헙 아이디와 암호 입력한다. store in secure store에 체크하면 앞으로는 안 물어본다.

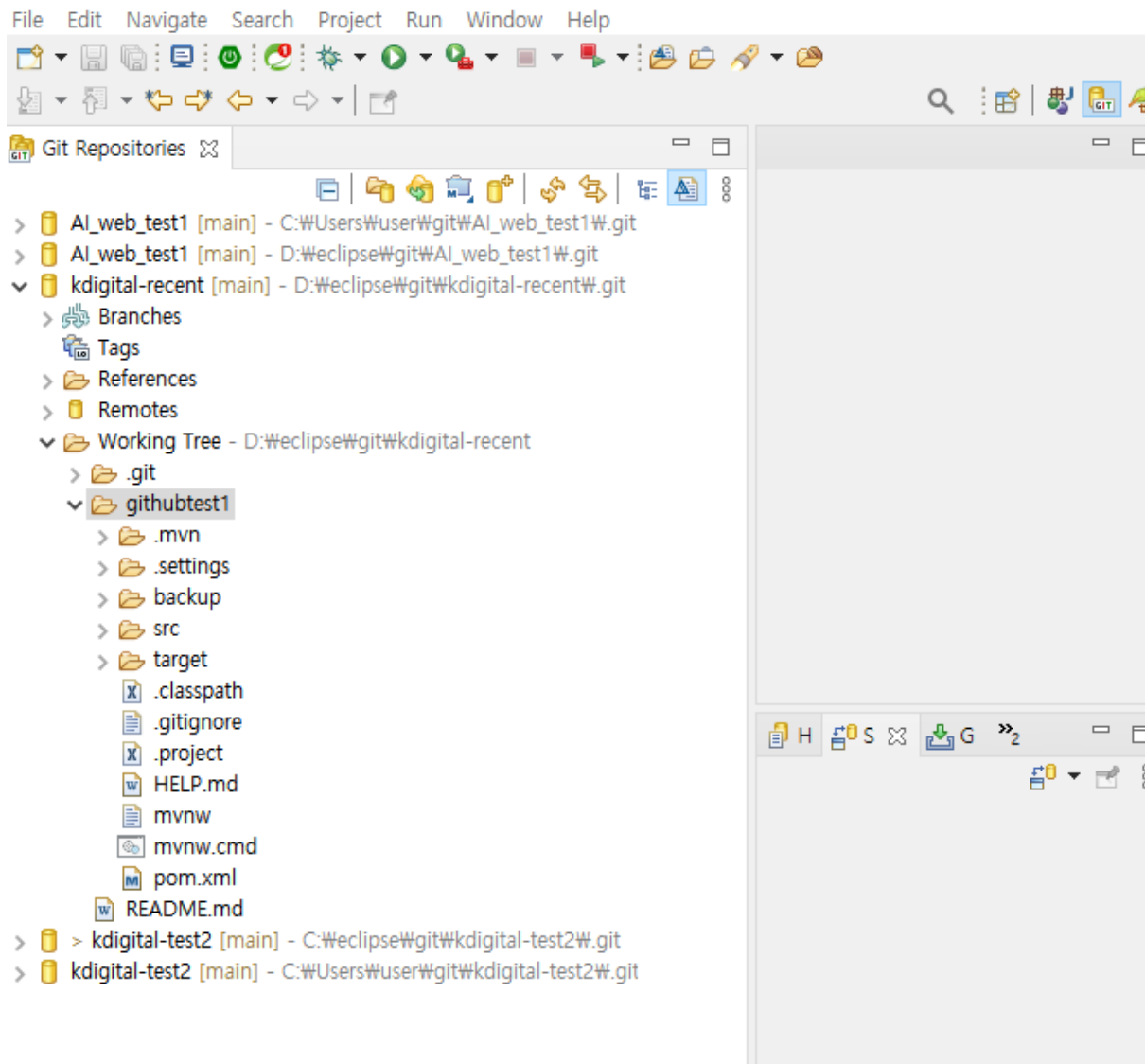


6. 아래 화면에서 close 한다

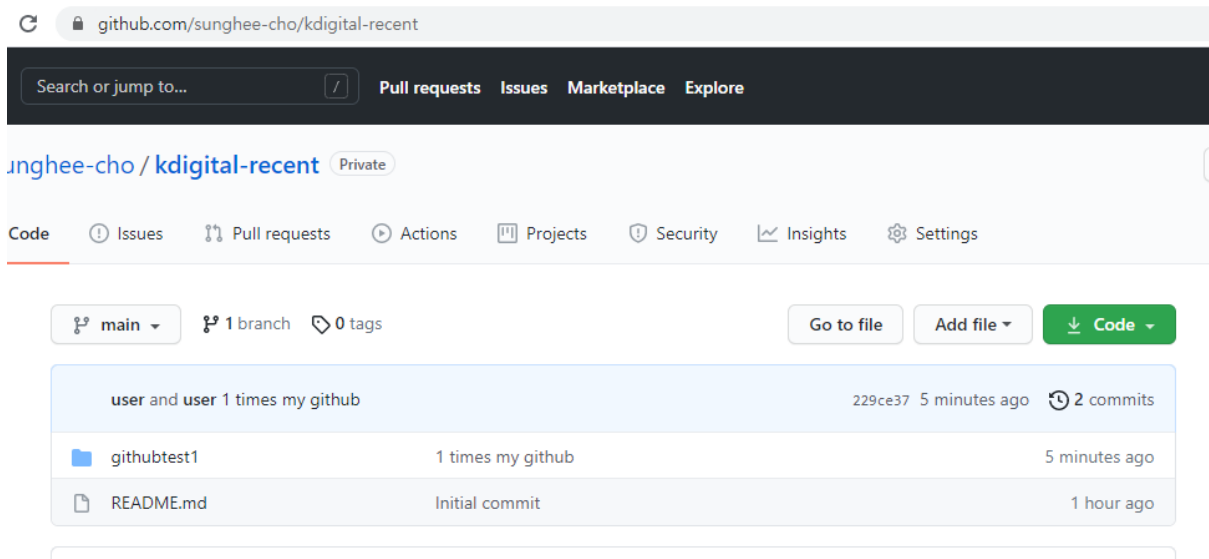


10. 로컬 저장소를 확인한다. 오른 상단 git 클릭, 레포지토리이름, working tree 확인하면 업로드한 프로젝트가 보인다.





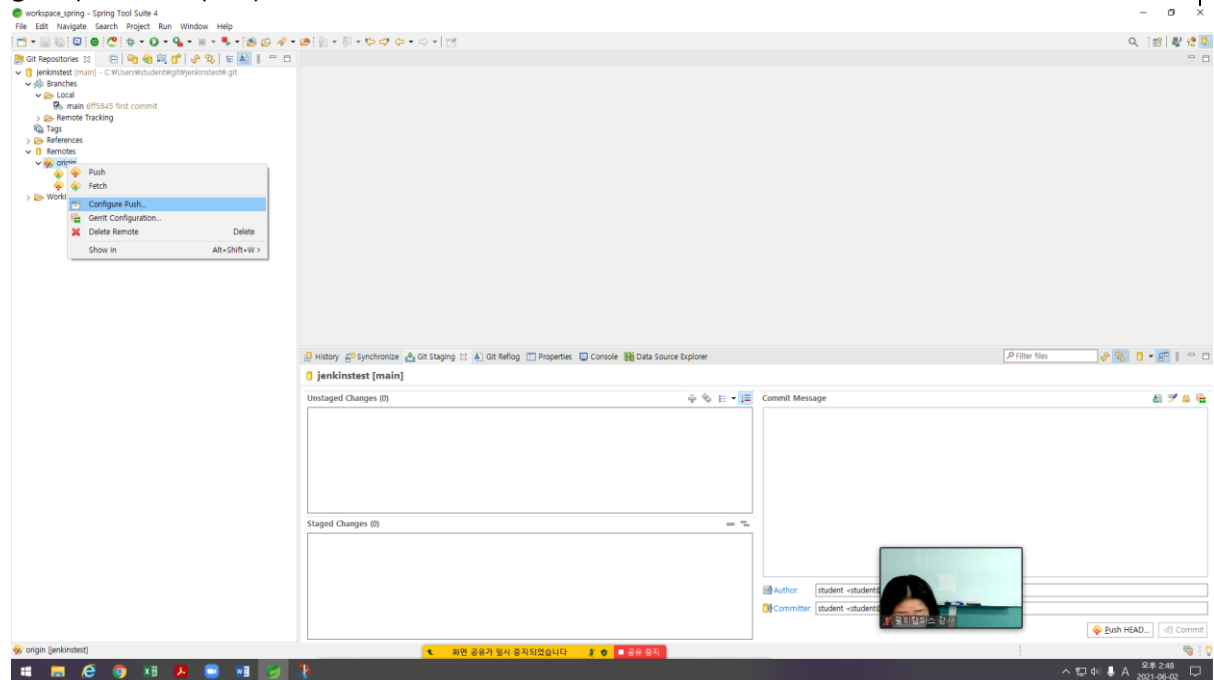
11. 원격 저장소를 확인한다.



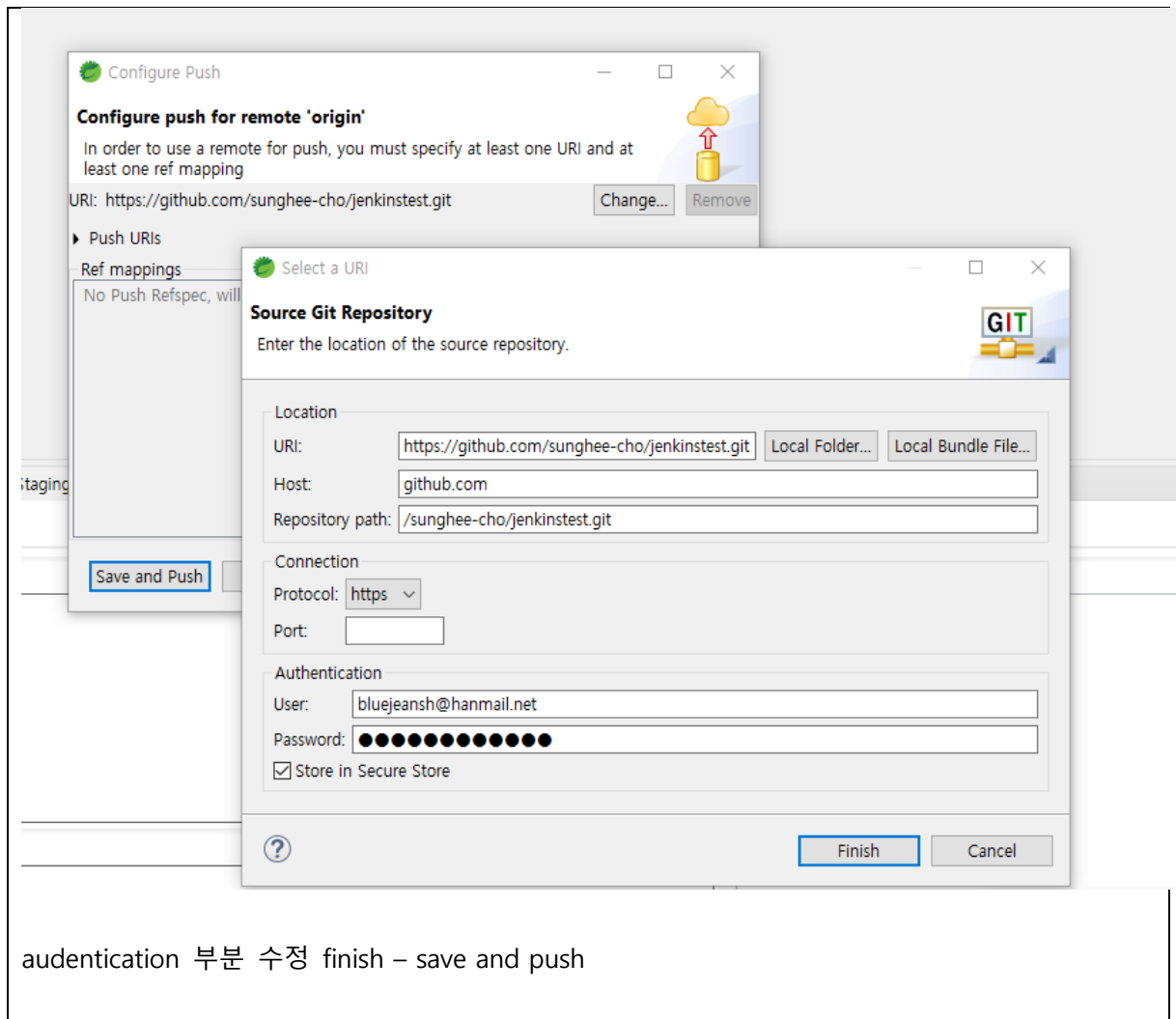
tip

id,password 정보가 잘못되면 ... git-receive-pack .. 오류 발생

git repositories perspective

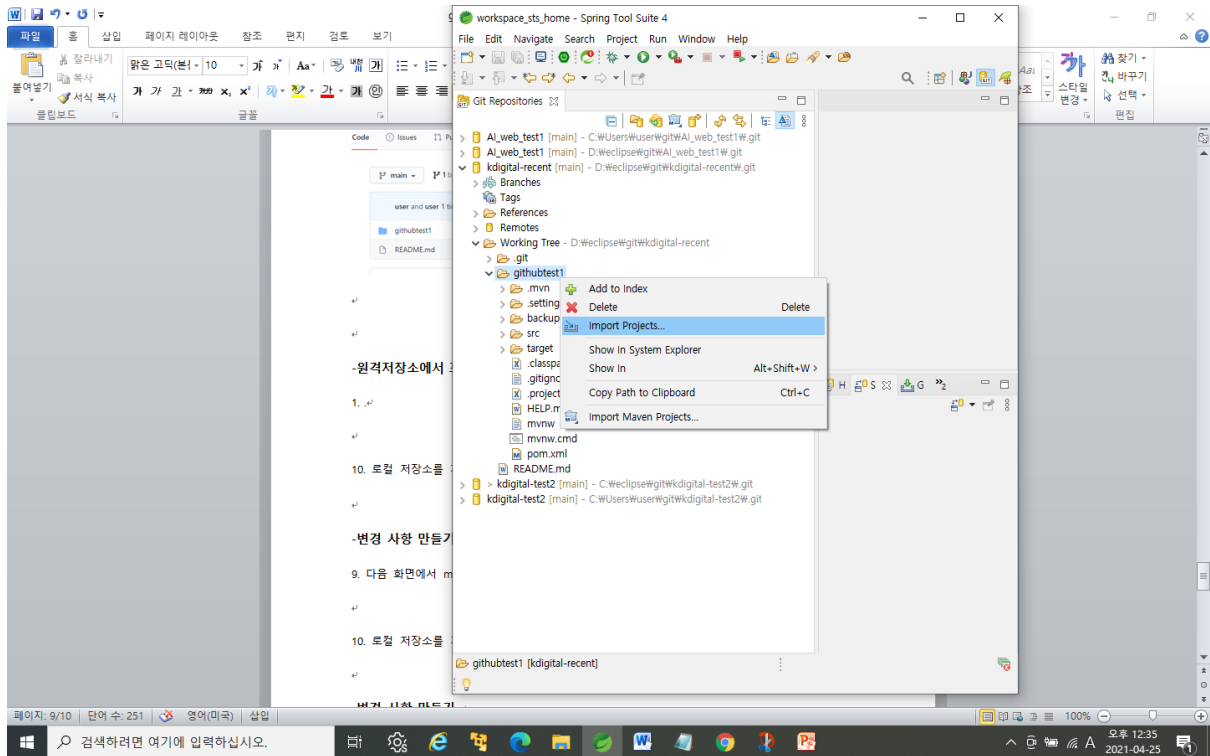


change 클릭

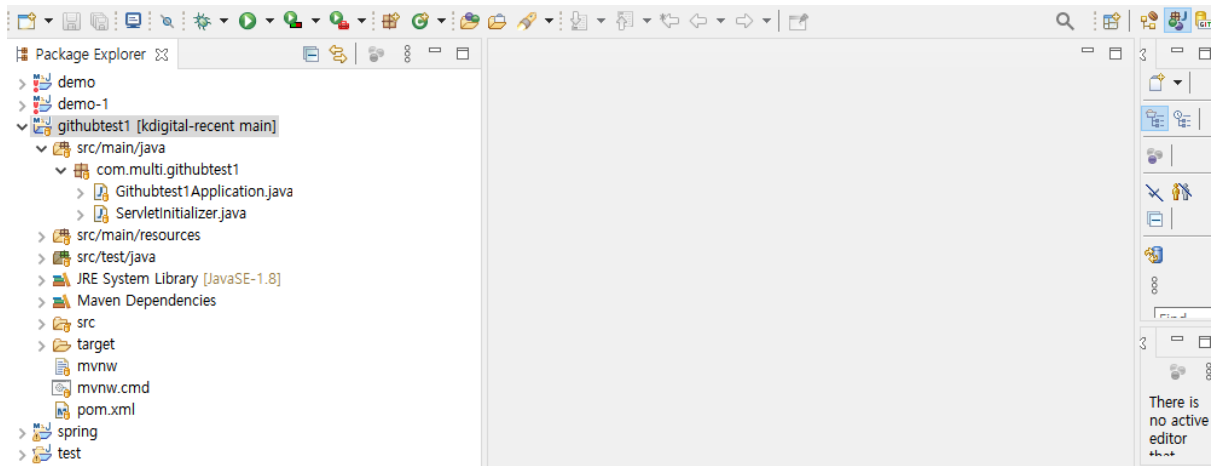


### -원격저장소에서 프로젝트 가져오기-

1. sts에서 같은 이름의 프로젝트를 삭제하거나 이클립스에서 import 한다.
2. git perspective에서 import project 한다



3. 가져온 프로젝트 아이콘이 공유 모양으로 변경되어 있다.



**저장소 – repository**

**local repository – 내컴퓨터 저장소(main이라는 브랜치로)**

**remote repository – 깃헙 저장소(origin/main이라는 브랜치로)**

**branch-** 코드를 전체 복사해서 독립적으로 개발하는 것 (자신만의 버전)

독립적인 작업 공간(기본적으로 main이라는 공간이 생긴다)

**checkout** – 다른 브랜치로 이동

**add** – 저장소로 전송할 파일 선택하여 준비하기

**commit** – 로컬 저장소로 변경된 내용 확정하여 파일 올리기

**push** – 원격 저장소로 파일 올리기

**fetch** – 원격 저장소에서 로컬 저장소로 파일 가져오기

**merge** – fetch된 파일을 로컬 저장소에 있는 파일들과 합치기

**pull** – fetch+merge

**-변경 사항 만들어 저장소로 가져가기-**

: jsp 파일 추가

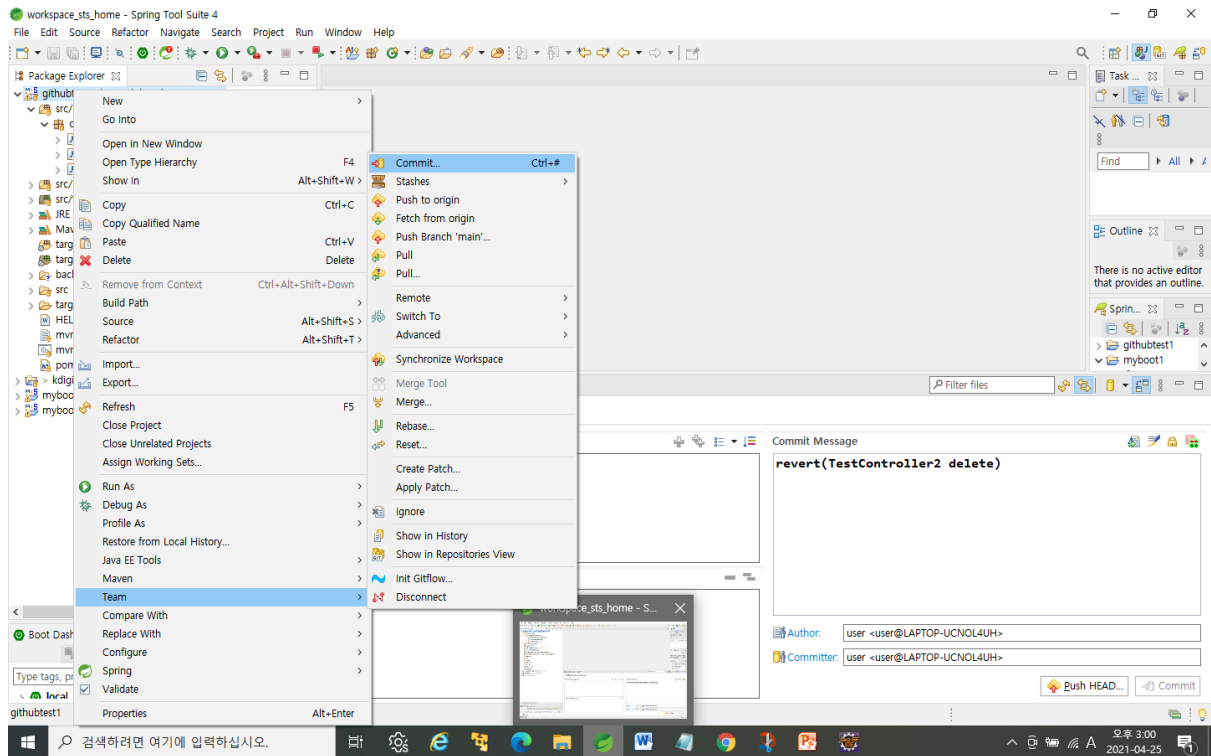
동기화 (Team / Synchronize Workspace)

Add to stage(변화없으면 git staging의 아이콘 메뉴에서 switch repository 선택 - refresh)

Commit push

(추가팁 - 로컬 저장소에서 파일 삭제 후 복구하려면 Team / Synchronize Workspace-overwrite 한다)

**-저장소에서 가져오기**



로컬과 원격 저장소 변경사항 확인한다.

team – pull 하여 변경사항 내려받는다.

=====

<실습환경>

-올리기

workspace_springboot gitupload 프로젝트	local c:/users/xxxx/git/myboottest	github https://github.com/sunghee-cho/myboottest.git
변경	반영	반영

---

## -내리기

workspace_spring	<-- local c:/users/xxxx/git/mvc/myboottest	github https://github.com/sunghee- cho/myboottest.git
------------------	--	---