

# Eclipse에서 GitHub 사용하기

## 1. GitHub 가입하고 Repository 생성하기

당연한 얘기겠지만 GitHub 서비스를 사용하기 위해서는 먼저 <http://www.github.com>에 회원으로 가입되어 있어야 한다. GitHub 서비스는 공용 및 오픈소스 프로젝트에 무료로 사용할 수 있으며 유료 플랜을 통해 무제한의 개인 저장소에서 작업할 수 있다.

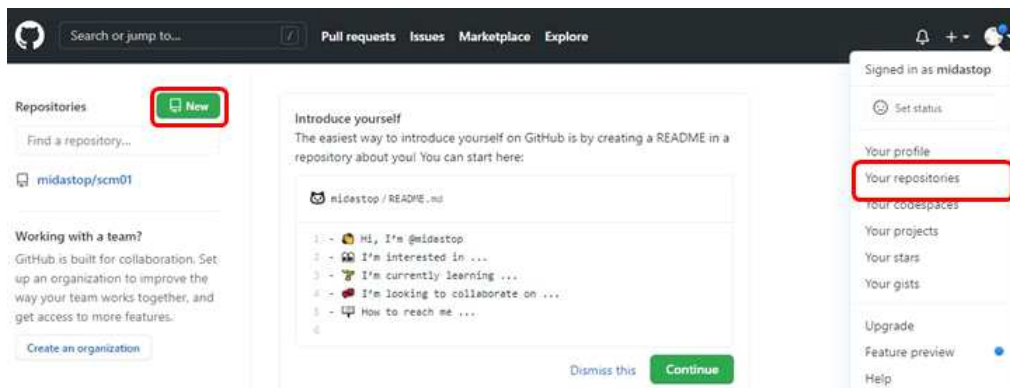
### Choose the plan that's right for you.

How often do you want to pay?

Monthly Yearly **Get 2 months free**

Free	Team	Enterprise
The basics for individuals and organizations	Advanced collaboration for individuals and organizations	Security, compliance, and flexible deployment
\$0	\$48	\$252
per year forever	\$40	\$210
	per user/year for the first 12 months*	per user/year for the first 12 months*
<a href="#">Create a free organization</a>	<a href="#">Continue with Team</a>	<a href="#">Contact Sales</a>
<ul style="list-style-type: none"><li>&gt; Unlimited public/private repositories</li><li>&gt; Automatic security and version updates</li><li>&gt; 2,000 CI/CD minutes/month Free for public repositories</li><li>&gt; 500MB of Packages storage Free for public repositories</li><li>&gt; New Issues &amp; Projects (in limited beta)</li></ul>	<ul style="list-style-type: none"><li>&lt; Everything included in Free, plus...</li><li>&gt; Access to GitHub Codespaces</li><li>&gt; Protected branches</li><li>&gt; Multiple reviewers in pull requests</li><li>&gt; Draft pull requests</li></ul>	<ul style="list-style-type: none"><li>&lt; Everything included in Team, plus...</li><li>&gt; Enterprise Managed Users</li><li>&gt; User provisioning through SCIM</li><li>&gt; Enterprise Account to centrally</li></ul>

GitHub는 회원에 가입하고 나서 가입시 등록한 이메일을 통해 본인 인증을 해야 비로소 서비스를 사용할 수 있다. email 확인이 완료된 후 GitHub에 로그인 하면 아래 그림과 같이 Dashboard가 나타나는데 이 화면에서 다음과 같이 저장소(Repository)를 생성할 수 있는 메뉴를 만날 수 있다.



앞의 화면에서 New repository 메뉴를 클릭하면 아래와 같은 속성을 선택해 저장소를 생성할 수 있는 화면이 나타난다.

## 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 \* 저장소이름 지정

midastop Team01

Great repository names are short and memorable. Need inspiration? How about [super-goggles?](#)

Description (optional)

2차 프로젝트 협업용 저장소

☒ 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.

GitHub에서는 저장소에 README 파일을 만들 것을 권장한다.

☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

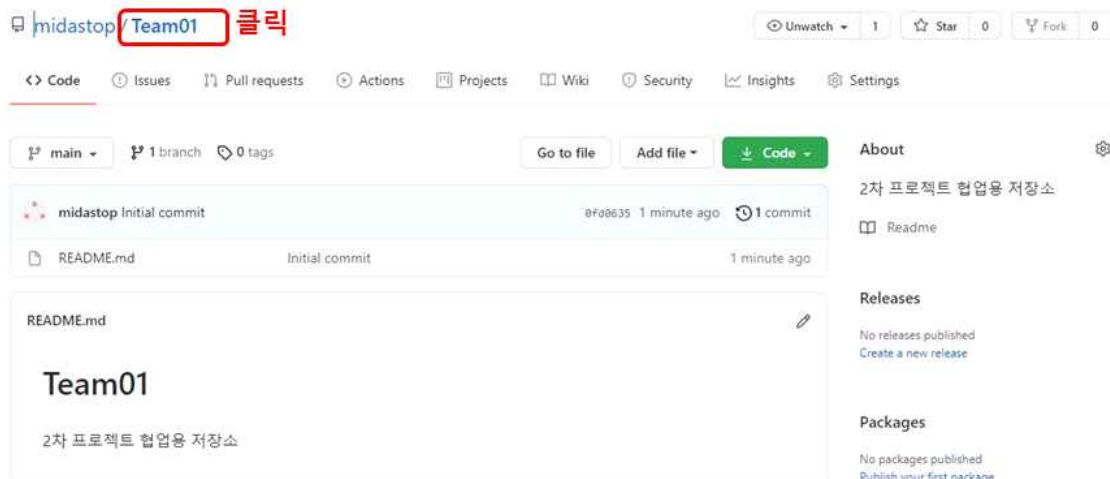
☐ Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ 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](#).

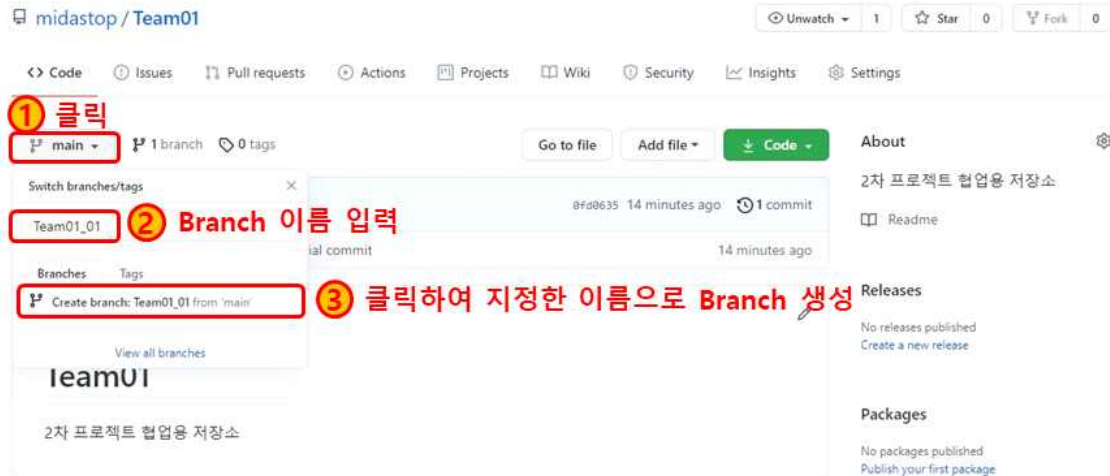
Create repository

위의 화면에서 “Create repository”를 클릭하면 아래와 같이 새로운 저장소가 생성된다.

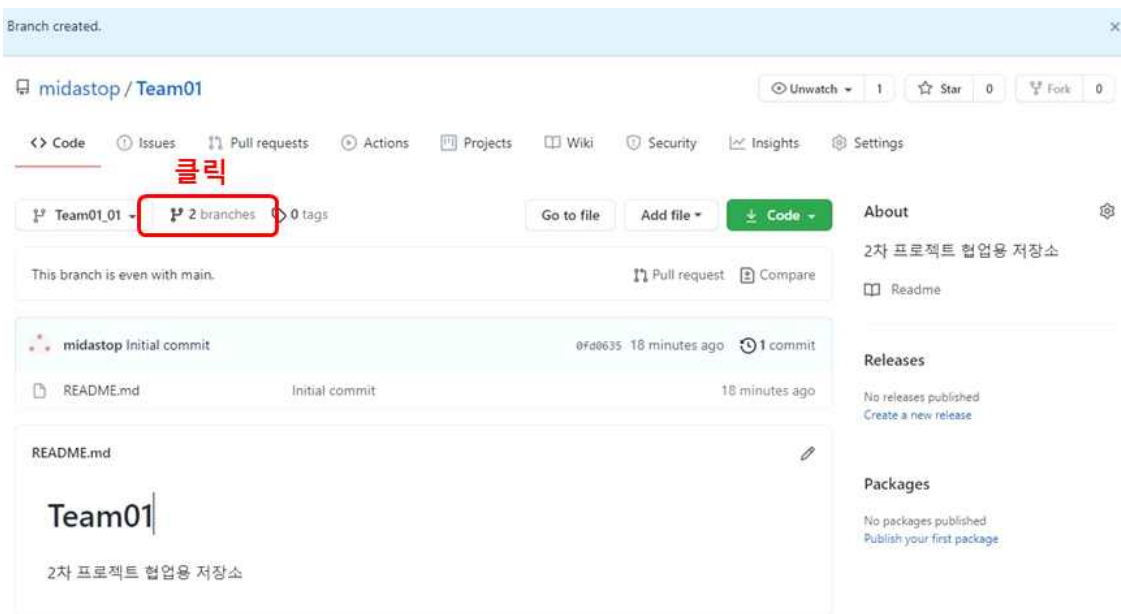


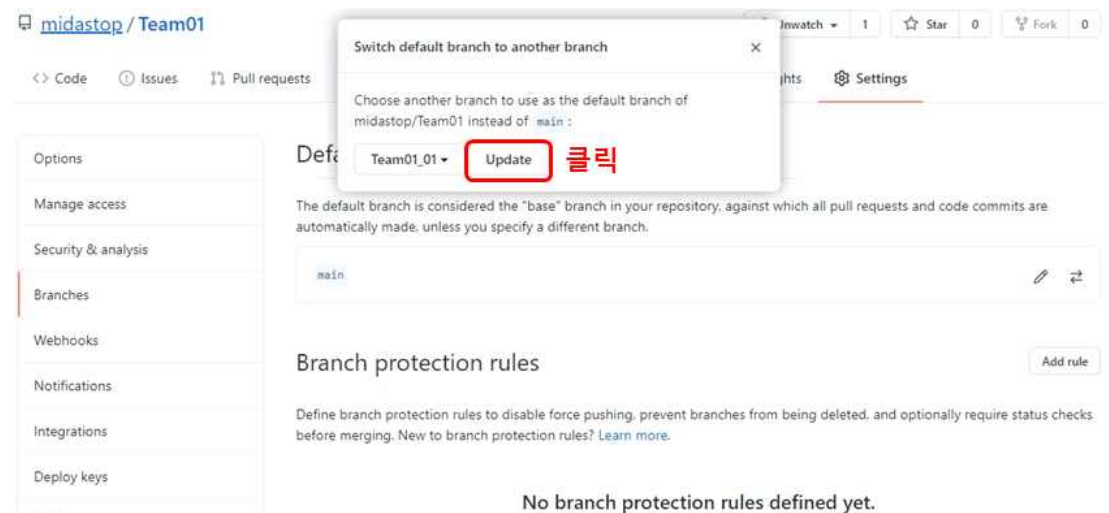
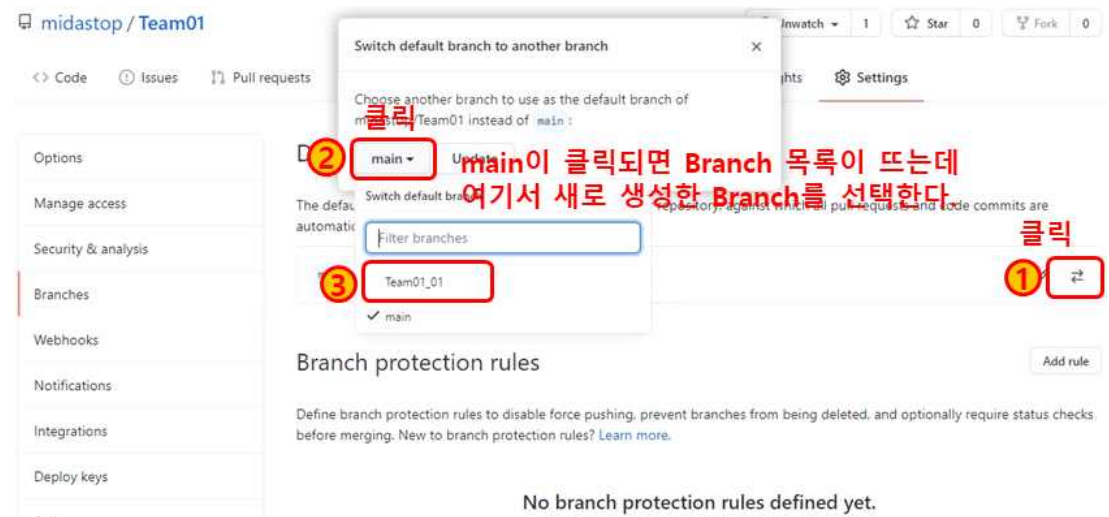
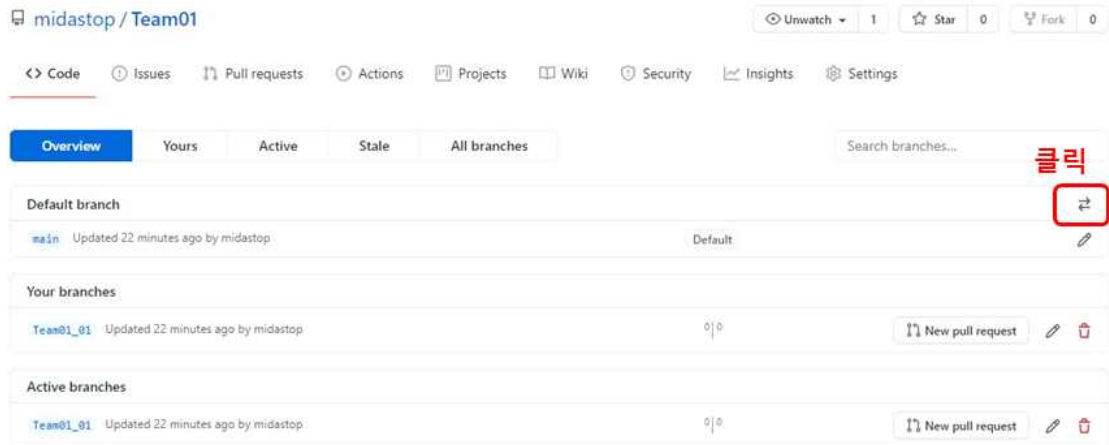
새로운 저장소를 생성했다면 이 저장소에 저장될 프로젝트 소스의 안전을 위해서 최소 하나 이상의 Branch를 만들어 주는 것이 좋다.

다음 그림은 main(이전 master) Branch외에 Team01\_01 이라는 Branch를 생성하는 예이다.

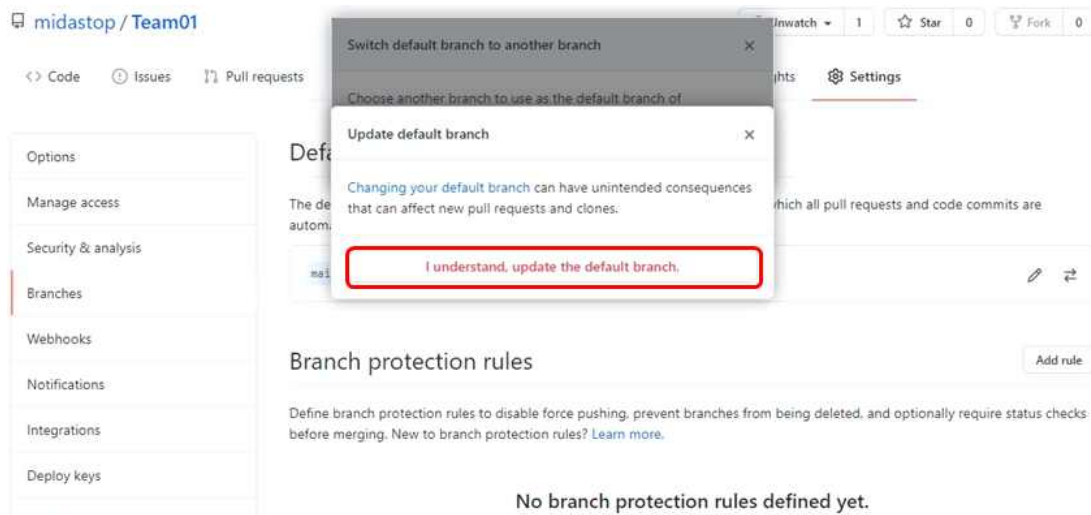


Branch가 만들어지면 다음 그림과 같이 branches를 클릭해 기본으로 사용할 branch를 설정한다.

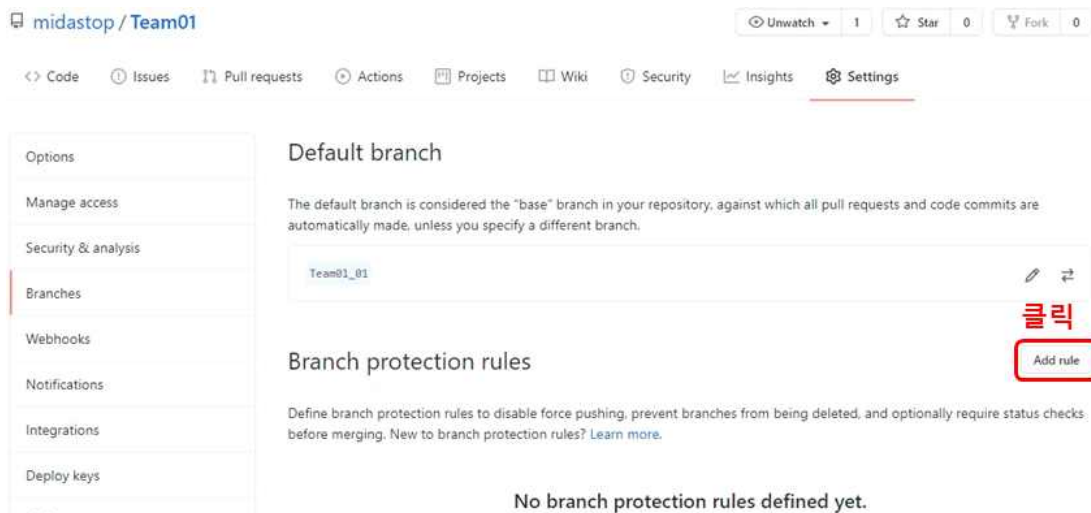




위의 그림과 같이 “Update” 버튼을 클릭하면 아래 그림과 같이 Update default branch 대화상자가 나타나는데 여기서 “I understand update the default branch”를 클릭해 다음으로 넘어간다.



Default branch가 변경되었으면 아래 그림과 같이 main branch 보호를 설정하자.



Options

Manage access

Security &amp; analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Environments

Secrets

Moderation settings

## Branch protection rule

Branch name pattern

main

main(master) Branch 입력

Protect matching branches

☐ Require pull request reviews before merging

When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.

☒ Require status checks to pass before merging

Choose which status checks must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

☒ Require branches to be up to date before merging

The chosen pull requests targeting a matching branch have been tested with the latest code. This setting will not take effect unless at least one status check is enabled (see below).

## No status checks found

Sorry, we couldn't find any status checks in the last week for this repository.

[Learn more about status checks](#)☐ Require signed commits

Commits pushed to matching branches must have verified signatures.

☐ Require linear history

Prevent merge commits from being pushed to matching branches.

☒ Include administrators

Enforce all configured restrictions above for administrators.

Rules applied to everyone including administrators

☐ Allow force pushes

Permit force pushes for all users with push access.

☐ Allow deletions

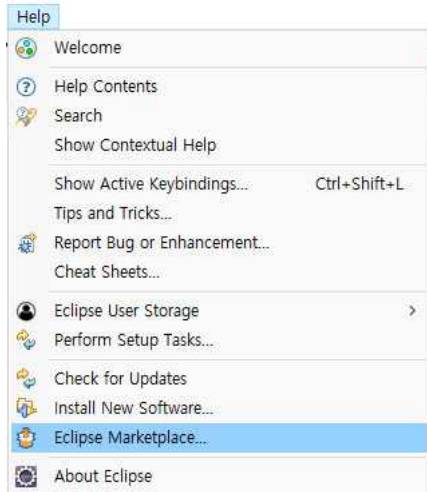
Allow users with push access to delete matching branches.

Create

옵션을 선택하고 Create 버튼을 클릭하면 비밀번호 입력 화면이 나타나고 올바른 비밀번호를 입력하면 Branch protection rules 설정이 완료된다.

## 2. Eclipse에 EGIt 플러그인 설치하기(최신 버전은 설치되어 있음)

최근에 출시되는 Eclipse 또는 SpringToolSuite(이하 이클립스)을 설치하면 GitHub 관리 툴인 EGIt이란 플러그인이 설치되어 있지만 따로 설치해야 할 경우에는 아래 그림과 같이 Eclipse Help -> Eclipse Marketplace... 메뉴를 선택해 Eclipse Marketplace 창에서 EGIt으로 검색해 설치할 수 있다.



EGIt 플러그인을 설치하는 과정에서 라이선스 동의 창이 나오는데 라이선스 동의를 하고 “Next” 버튼을 클릭해 계속 진행하면 된다. 그리고 Security Warning 경고창이 한 번 뜨는데 이때에도 “Install anyway”를 클릭해 설치를 계속 진행하면 된다. 플러그인 설치가 완료되면 이클립스 재시작을 묻는 창이 뜨는데 이때 이클립스를 재시작하면 플러그인을 사용할 수 있게 된다.

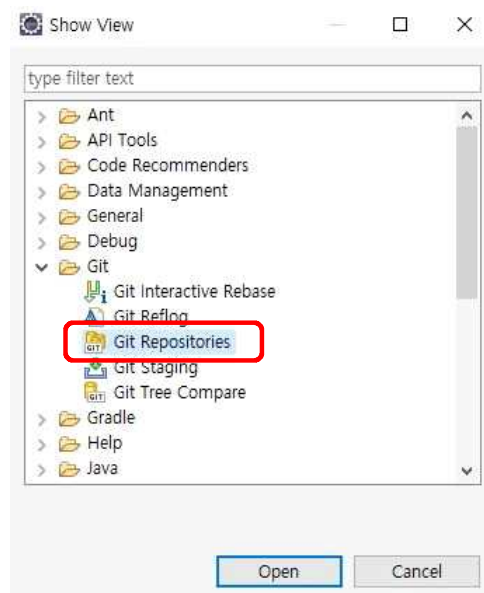
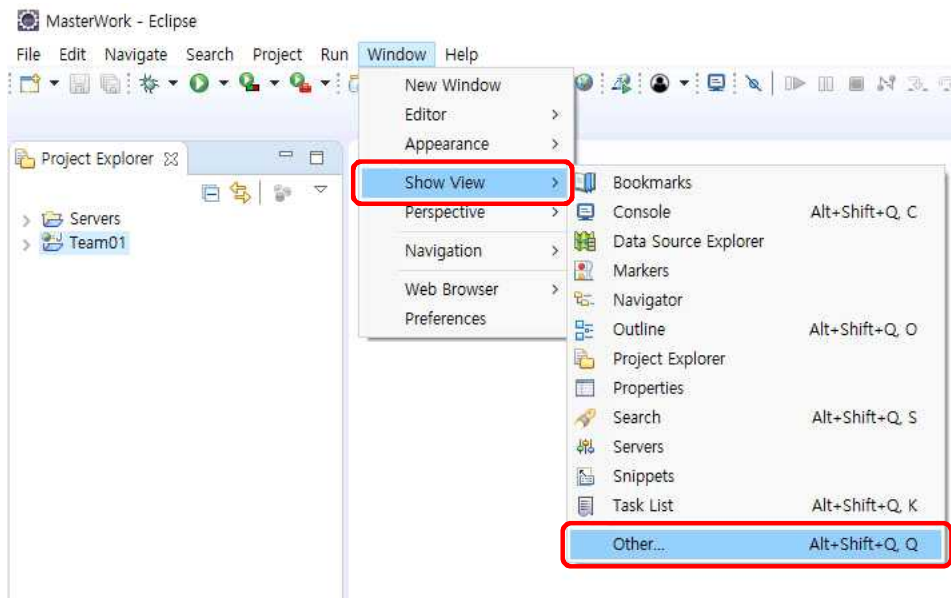


### 3. Eclipse에서 로컬 저장소 생성하고 프로젝트 연결하기

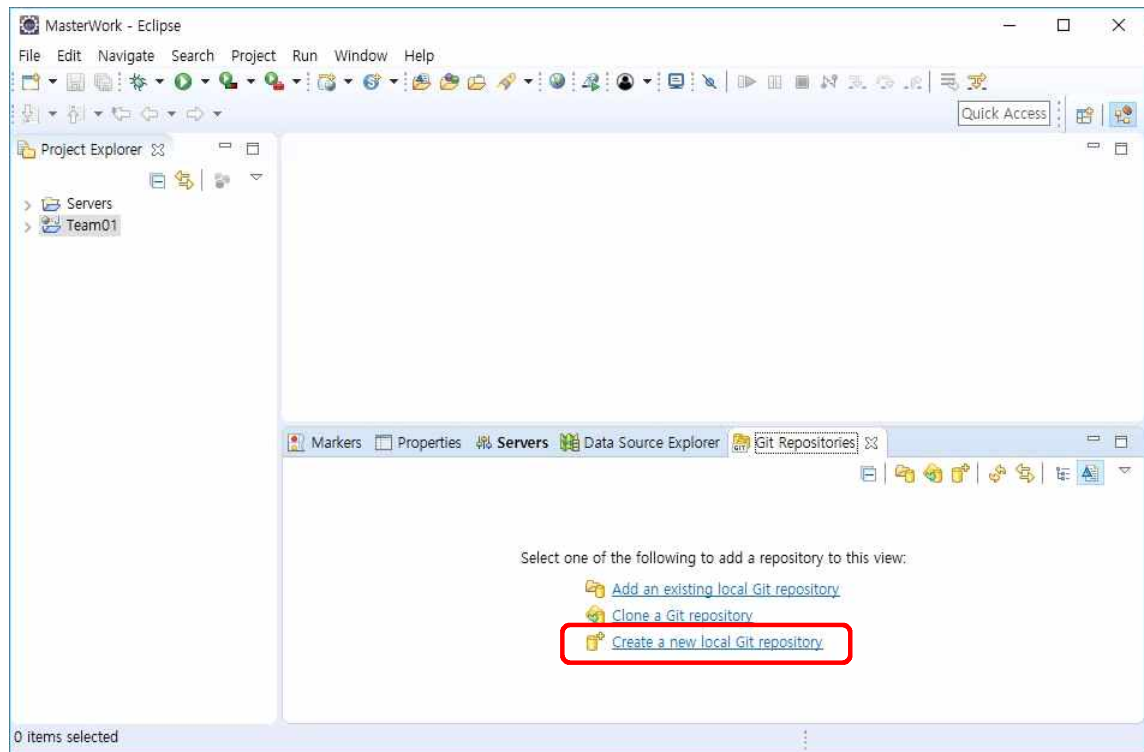
프로젝트 파일을 Eclipse에서 바로 GitHub에 업로드 할 수 있는 것이 아니라 로컬 저장소를 만들어 프로젝트 파일을 추가하고 GitHub에 업로드 해야 하기 때문에 먼저 로컬 저장소를 만들어 프로젝트 파일을 연결하는 방법에 대해 알아 볼 것이다.

먼저 Eclipse를 실행해 GitHub에 저장할 “Team01” Dynamic Web Project를 만들자.

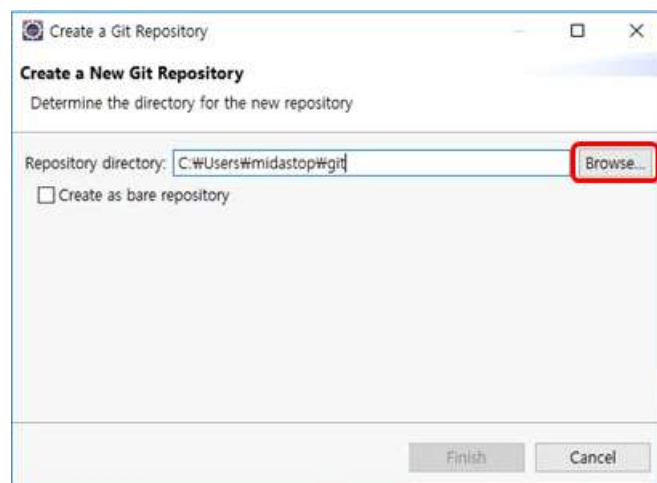
그리고 Eclipse에서 아래 그림과 같이 Window - Show View - Other 메뉴를 선택하면 Show View 대화상자가 나타난다.

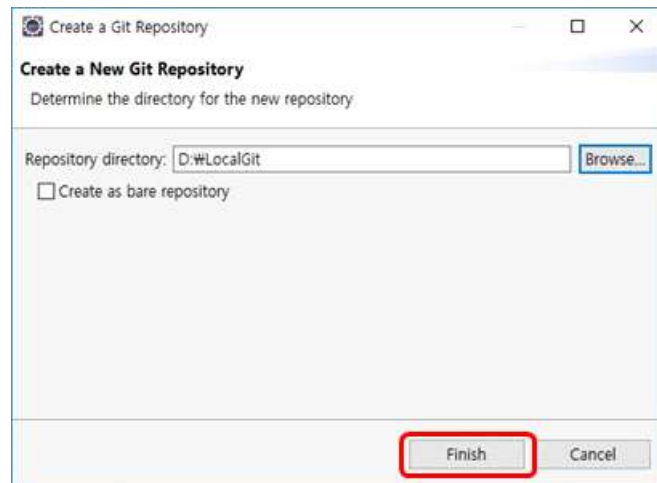
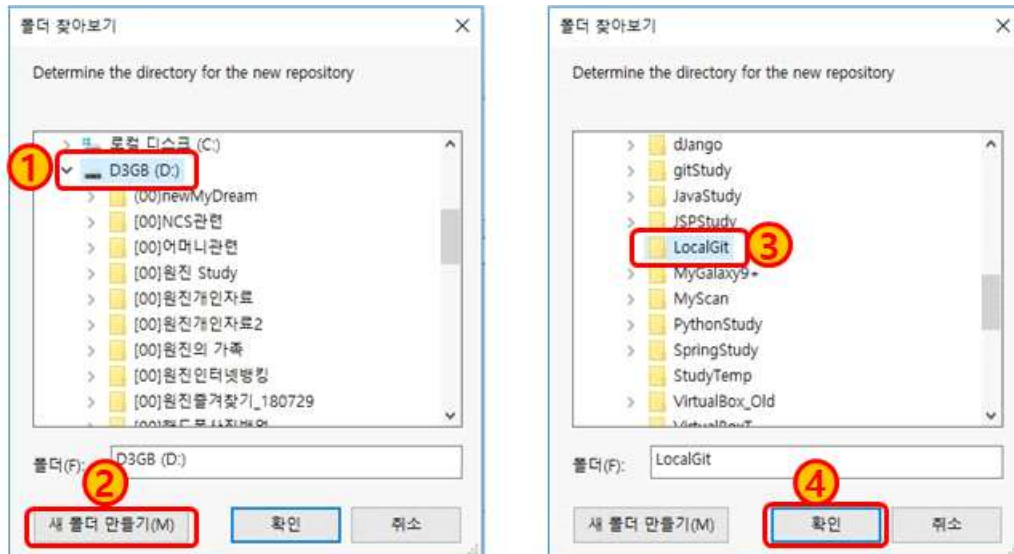


Show View 대화상자에서 Git - Git Repositories를 선택하고 Open 버튼을 클릭하면 다음과 그림과 같이 Git Repositories 뷰가 Eclipse 화면에 나타난다.

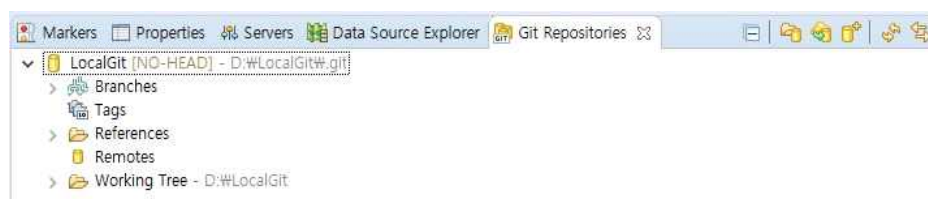


먼저 Git의 Local 저장소를 만들어 보자. 위의 그림에서 “Create a new local Git repository”를 클릭하면 다음과 같이 “Create a Git Repository” 대화상자가 나타나는데 이 화면에서 “Browse...” 버튼을 클릭하여 다음 그림들과 같이 로컬 저장소로 사용할 폴더를 지정한다.





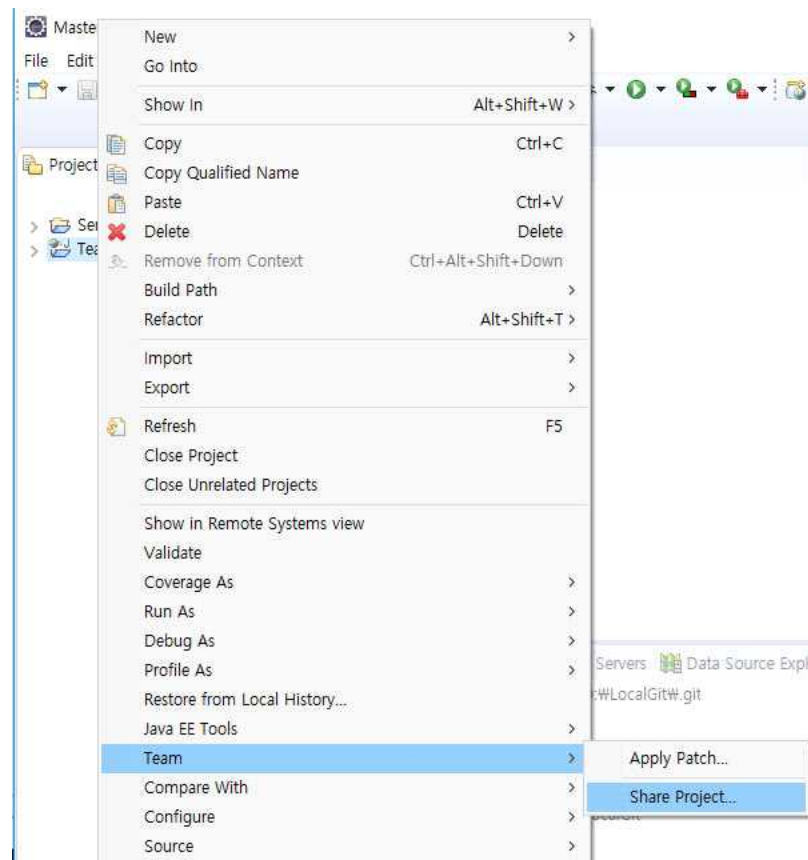
다음 그림과 같이 Git Repositories 뷰에 로컬 저장소가 생성된 것이 보일 것이다.



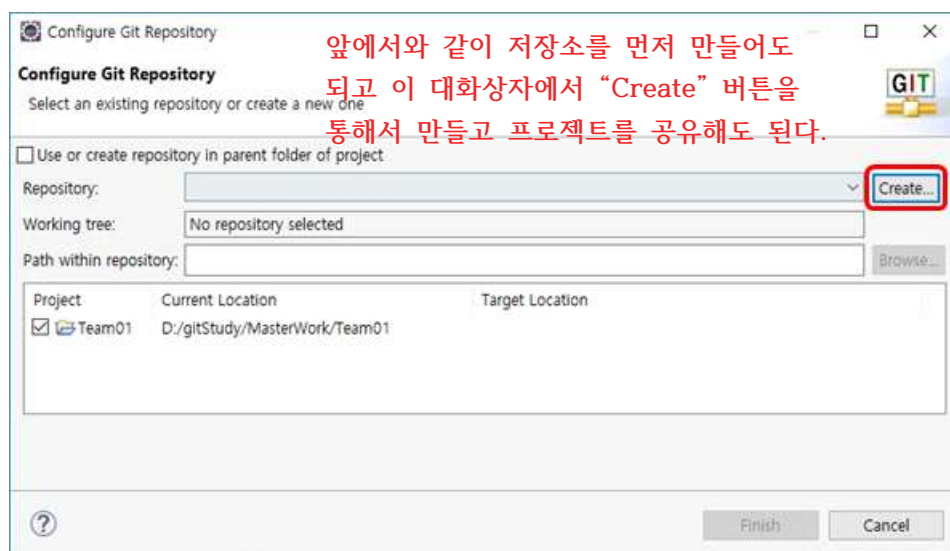
로컬 저장소를 생성했으니 이제 프로젝트를 로컬 저장소와 연결해 저장해 보자.

GitHub에 공유할 프로젝트에 마우스 우 클릭해서 아래 그림과 같이 Team - Share Project... 메뉴

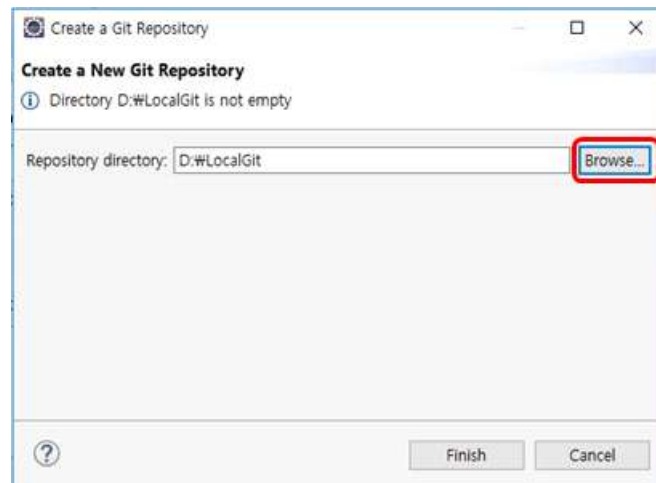
뉴를 선택한다.



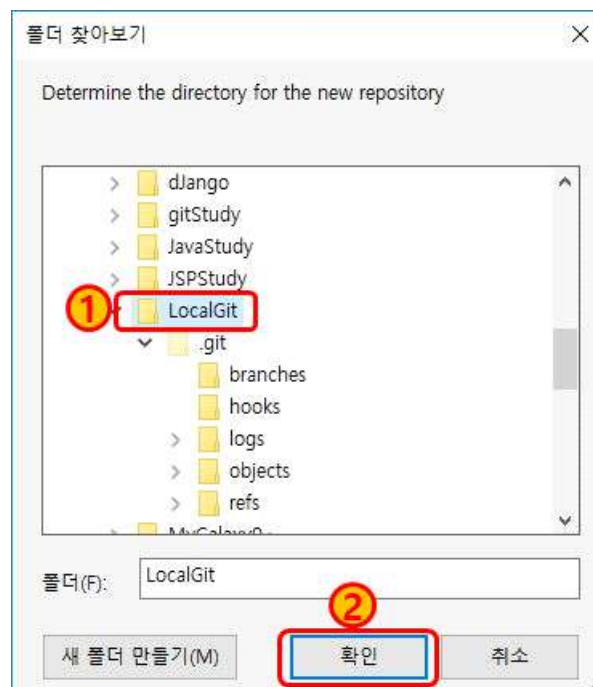
다음과 같이 “Configure Git Repository” 대화상자가 나타나면 “Create” 버튼을 클릭하여 “Create a Git Repository” 대화상자를 띄운다.



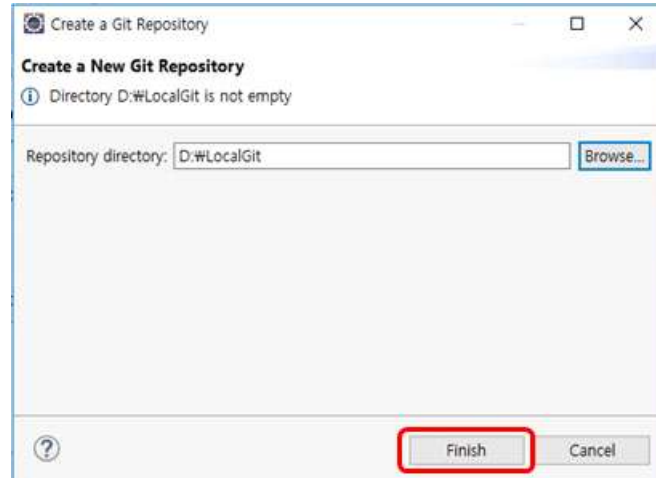
다음의 “Create a Git Repository” 대화상자에서 “Browse...” 버튼을 클릭해 “폴더 찾아보기” 대화 상자를 띄운다.



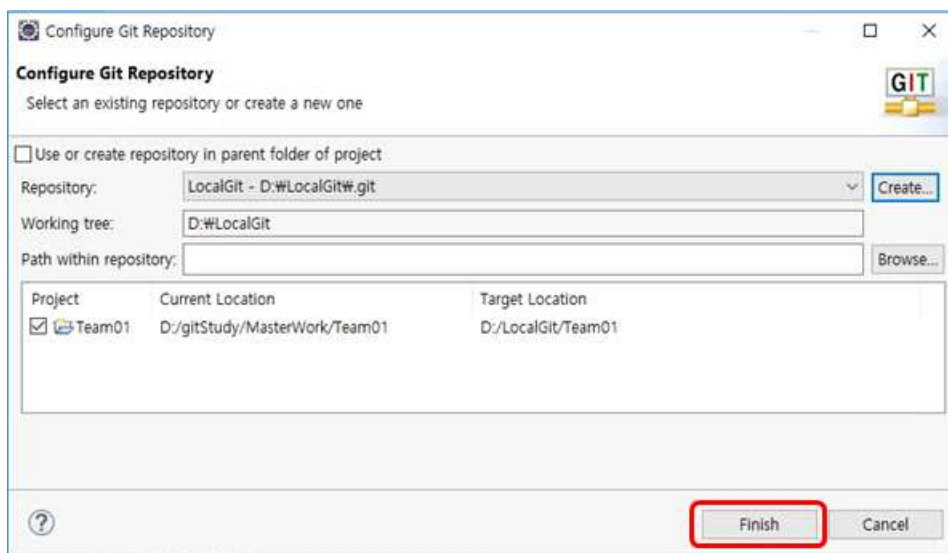
아래의 “폴더 찾아보기” 대화상자에서 새로운 저장소를 만들거나 기존의 저장소를 지정하면 된다. 우리는 앞에서 D:\LocalGit 폴더를 로컬 저장소로 만들었으니 아래 그림과 같이 앞에서 미리 생성한 “LocalGit” 폴더를 선택하고 “확인” 버튼을 클릭하여 “폴더 찾아보기” 대화상자를 닫는다.



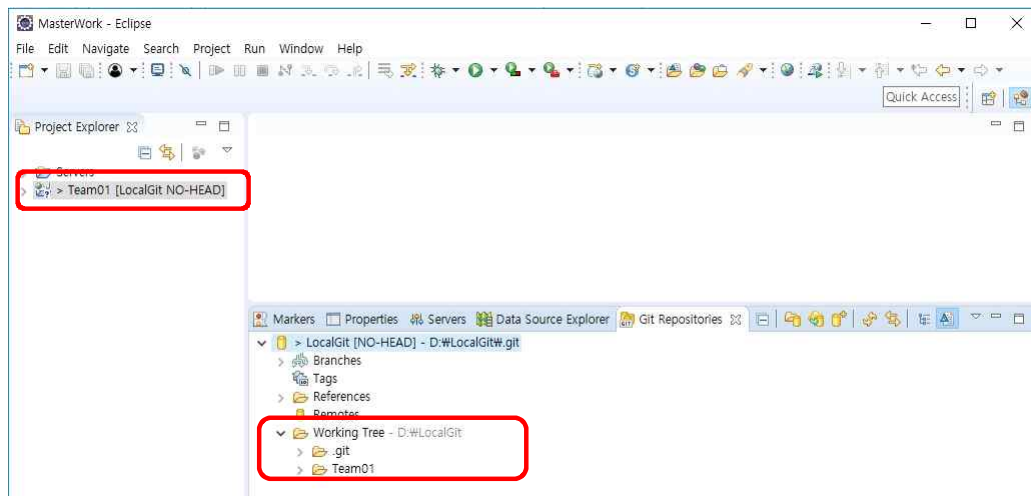
그러면 다음 그림과 같이 “Create a Git Repository” 대화상자에서 로컬 저장소로 생성한 D:\LocalGit 폴더를 “Repository directory”로 설정된 것을 확인할 수 있을 것이다. 이 “Create a Git Repository” 대화상자에서 “Finish” 버튼을 클릭 대화상자를 닫는다.



아래의 “Configure Git Repository” 대화상자를 살펴보면 Current Location은 현재 Eclipse의 Workspace에 저장된 프로젝트의 폴더 위치이고 Target Location은 로컬 저장소인 D:\LocalGit으로 설정된 것을 확인할 수 있다.

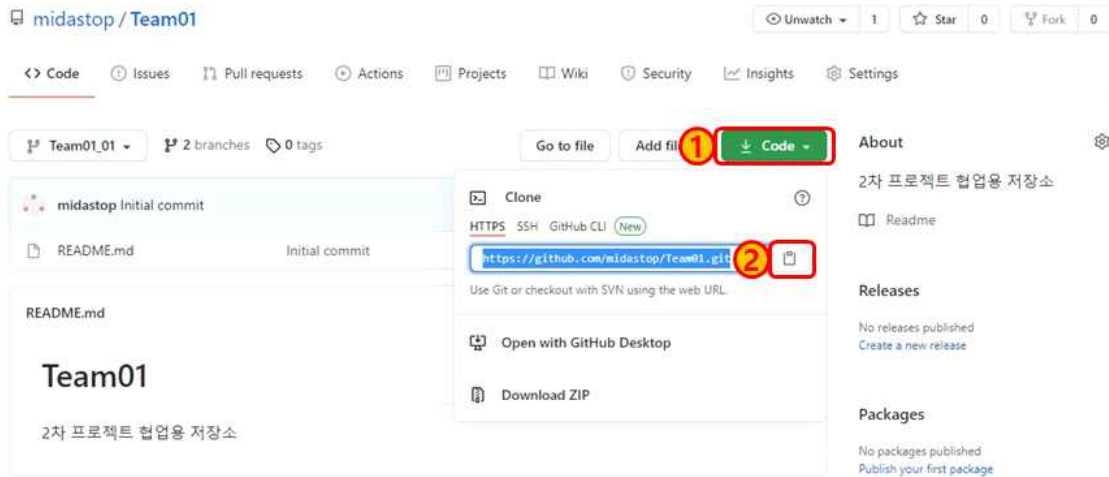


위의 “Configure Git Repository” 대화상자에서 “Finish” 버튼을 클릭하면 우리의 프로젝트가 Eclipse의 Workspace에서 로컬 저장소로 지정한 D:\LocalGit 폴더 아래로 이동해 저장된다. 그리고 다음 그림과 같이 Eclipse의 “Project Explorer”와 “Git Repositories”를 살펴보면 우리의 프로젝트가 로컬 저장소와 연결된 것을 확인할 수 있다.

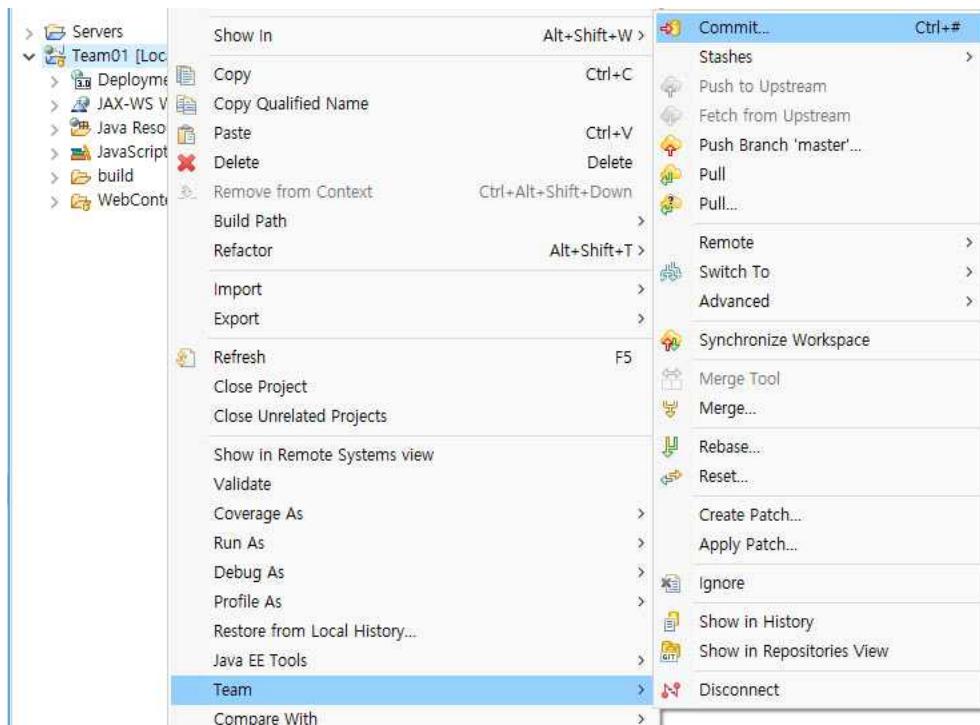


## 4. Eclipse에서 GitHub Repository에 프로젝트 올리기

Eclipse에서 GitHub 저장소를 설정하기 위해서는 저장소 주소가 필요하다. 먼저 아래 그림을 참고해 GitHub에 생성한 저장소 주소를 클립보드에 복사 하자.



그리고 다음 그림과 같이 프로젝트에 마우스 우 클릭해서 Team - Commit... 메뉴를 차례대로 선택한다.



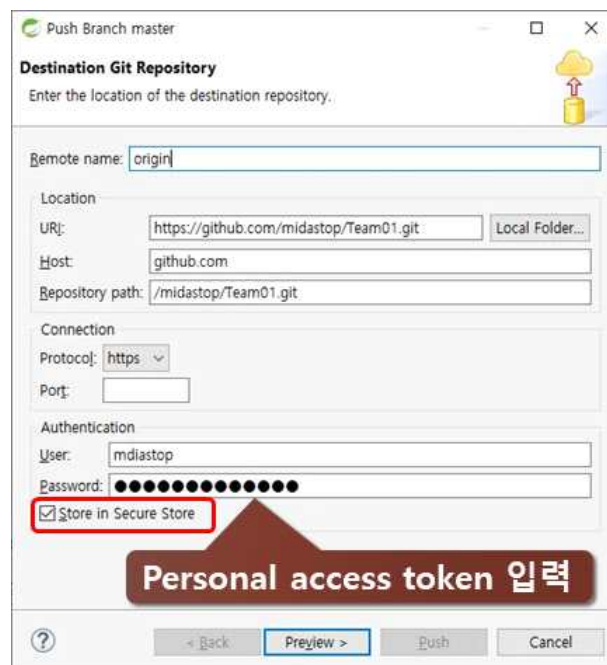
다음 그림과 같이 “Git Staging” 뷰가 Eclipse에 나타나는데 아래 그림과 같이 “Unstaged Changes”에서 파일 추가 버튼을 클릭해 “Staged Changes”로 GitHub에 올릴 파일을 추가하고 현재 Commit에 대한 주석을 추가한다.





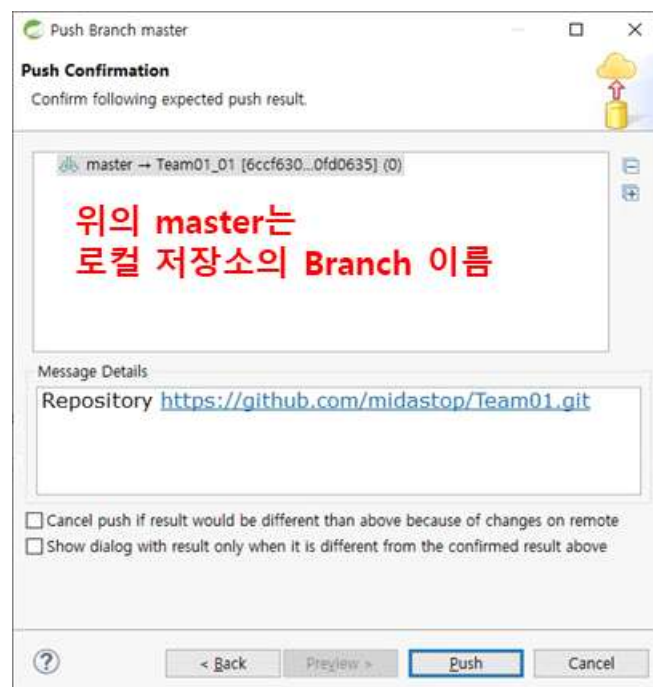
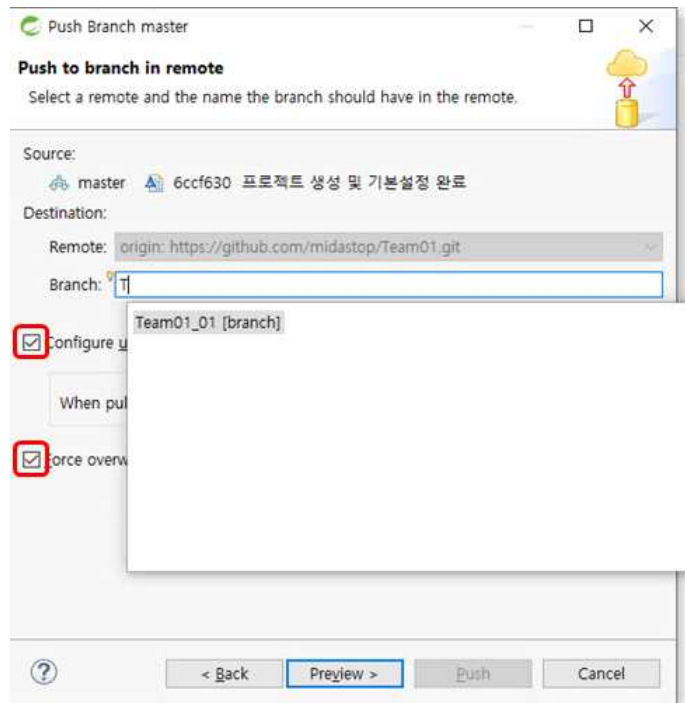
“Commit and Push...” 버튼을 클릭하면 다음과 같은 “Destination Git Repository” 대화상자가 나타나는데 URI 입력란에 앞에서 클립보드에 복사해 둔 GitHub URL을 입력한다. GitHub 주소가 클립보드에 복사된 상태라면 이 창이 뜨면서 자동으로 주소가 입력된다.

User와 Password 입력란에 GitHub 아이디와 비밀번호를 입력하고 “Store in Secure Store”를 체크한 다음 “Preview >” 버튼을 클릭 한다.



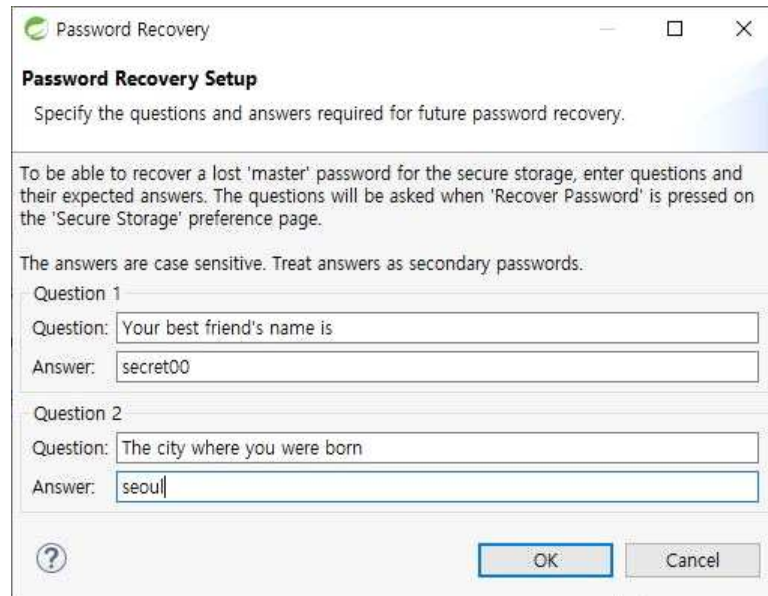
“Push Branch master” 대화상자가 뜨면 아래 그림과 같이 Branch 입력란에 GitHub 저장소 Branch를 입력한다. Branch의 첫 글자만 입력하면 아래 그림과 같이 Branch를 선택할 수 있도록 표시해 준다. 아래 그림에서 “Configure upstream for push and pull”과 “Force overwrite

branch in remote if it exists and has diverged”를 체크하고 “Preview >” 버튼을 클릭하면 다음 그림과 같이 “Push Branch master” 대화상자가 뜨는데 여기서 “Push” 버튼을 클릭한다.



위에서 “Push” 버튼을 클릭하게 되면 Secure Storage - Password Hint Needed 대화상자가 나타나는데 “Yes” 버튼을 클릭하면 다음과 같이 Password Recovery Setup 대화상자가 나타나는데 이 대화상자에서 적절히 비밀번호 복구 설정을 지정한 다음 “OK” 버튼을 클릭하면 프로젝트 소스코

드가 GitHub에 푸쉬(Push) 된다.



The image shows a 'Password Recovery Setup' dialog box. It has a title bar with a green icon and the text 'Password Recovery'. Below the title bar, the text 'Password Recovery Setup' is followed by 'Specify the questions and answers required for future password recovery.' Below this, a paragraph explains that to recover a lost 'master' password, questions and answers must be entered, which will be asked when 'Recover Password' is pressed on the 'Secure Storage' preference page. It also notes that answers are case sensitive and should be treated as secondary passwords. There are two question sets. 'Question 1' has a question 'Your best friend's name is' and an answer 'secret00'. 'Question 2' has a question 'The city where you were born' and an answer 'seoul'. At the bottom, there is a help icon (question mark), an 'OK' button, and a 'Cancel' button.

**Password Recovery Setup**  
Specify the questions and answers required for future password recovery.

To be able to recover a lost 'master' password for the secure storage, enter questions and their expected answers. The questions will be asked when 'Recover Password' is pressed on the 'Secure Storage' preference page.

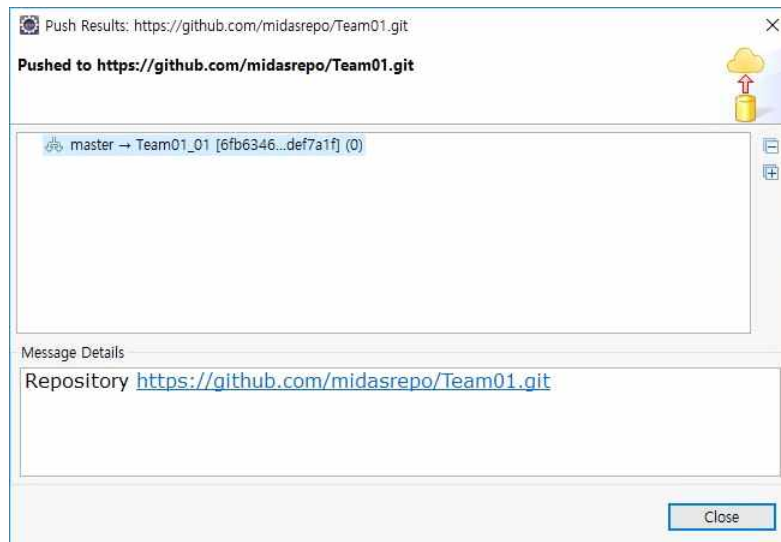
The answers are case sensitive. Treat answers as secondary passwords.

Question 1  
Question: Your best friend's name is  
Answer: secret00

Question 2  
Question: The city where you were born  
Answer: seoul

? OK Cancel

프로젝트 소스 코드가 GitHub에 업로드 되고 아래 그림과 같은 대화상자가 화면에 나타나면 “Close” 버튼을 클릭해 대화상자를 닫는다.



The image shows a 'Push Results' dialog box. It has a title bar with a blue icon and the text 'Push Results: https://github.com/midasrepo/Team01.git'. Below the title bar, the text 'Pushed to https://github.com/midasrepo/Team01.git' is displayed. Below this, a text area shows 'master → Team01\_01 [6fb6346...def7a1f] (0)'. To the right of the text area are icons for a folder and a plus sign. Below the text area, the text 'Message Details' is followed by 'Repository https://github.com/midasrepo/Team01.git'. At the bottom right, there is a 'Close' button.

Push Results: https://github.com/midasrepo/Team01.git

Pushed to https://github.com/midasrepo/Team01.git

master → Team01\_01 [6fb6346...def7a1f] (0)

Message Details  
Repository https://github.com/midasrepo/Team01.git

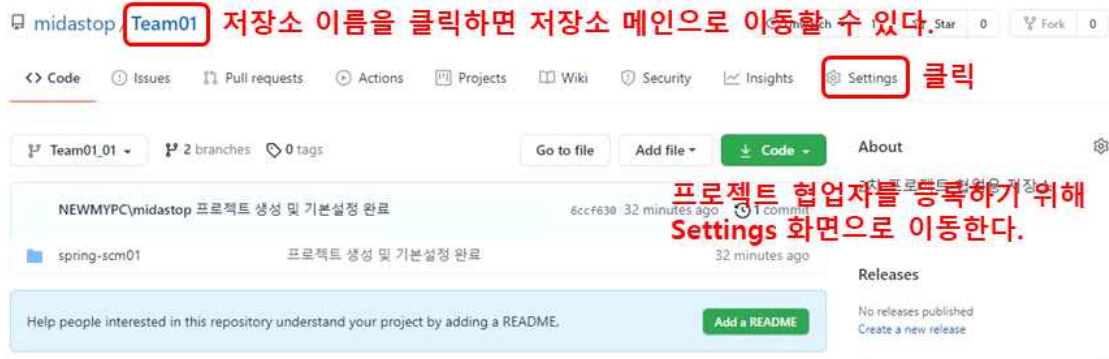
Close

GitHub의 저장소를 확인해 보면 다음과 같이 프로젝트 소스가 업로드 된 것을 확인할 수 있다.

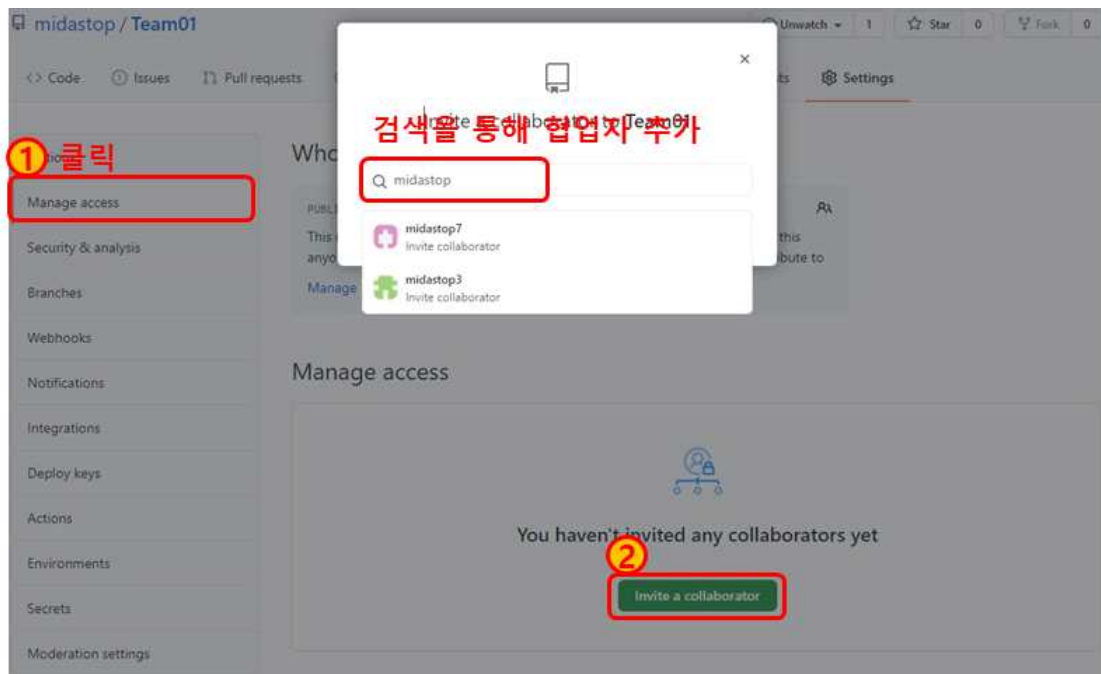
The screenshot shows the GitHub interface for a repository named 'midastop / Team01'. The repository has 1 watch, 0 stars, and 0 forks. The navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The breadcrumb path is 'Team01\_01 > Team01 / spring-scm01 /'. There are buttons for 'Go to file', 'Add file', and a menu icon. The main content area shows a file list for the path 'NEWMYPC\midastop 프로젝트 생성 및 기본설정 완료'. The file list includes folders like '.settings', 'SpringStudy3.9.15/spring-scm01/target/m2e-wtp/...', 'src', and 'target', and files like '.classpath', '.project', '.springBeans', and 'pom.xml'. All items are dated '26 minutes ago'.

File/Folder	Commit Hash	Time
..		
.settings	6ccf630	26 minutes ago
SpringStudy3.9.15/spring-scm01/target/m2e-wtp/...	6ccf630	26 minutes ago
src	6ccf630	26 minutes ago
target	6ccf630	26 minutes ago
.classpath	6ccf630	26 minutes ago
.project	6ccf630	26 minutes ago
.springBeans	6ccf630	26 minutes ago
pom.xml	6ccf630	26 minutes ago

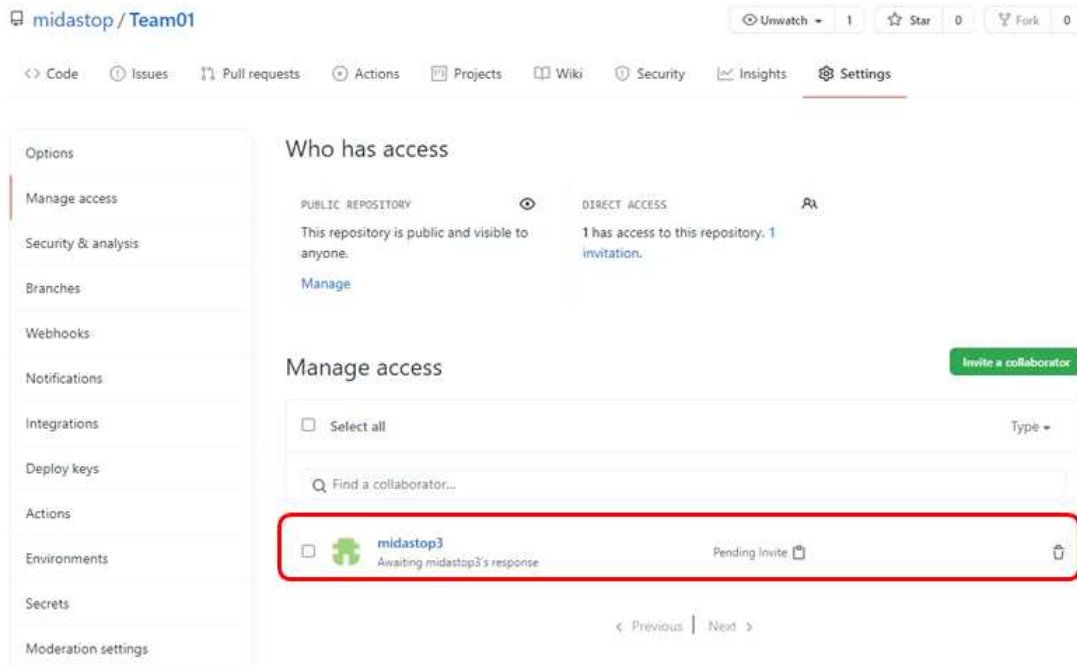
## 5. 프로젝트 협업자 등록하고 프로젝트 받아오기



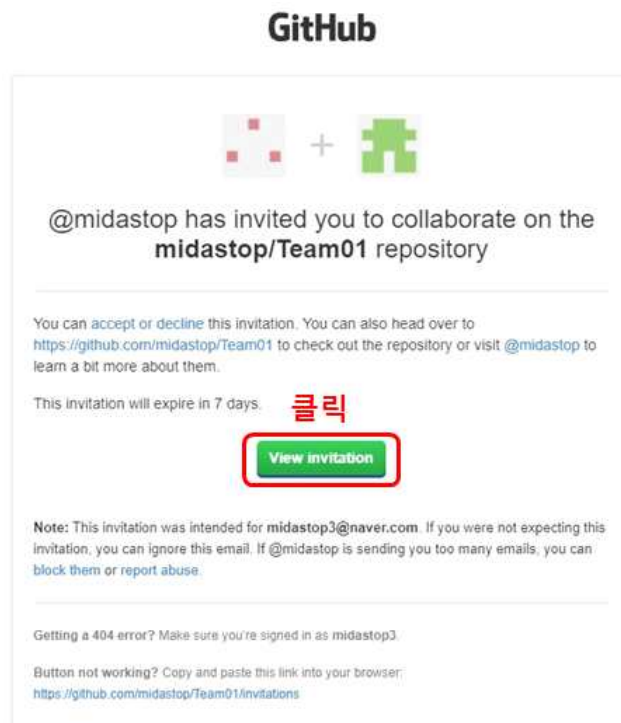
아래 그림에서 “Invite a collaborator” 버튼을 클릭하면 협업자를 검색해 추가할 수 있으며 검색된 리스트에서 추가할 협업자를 선택하여 추가하면 된다.



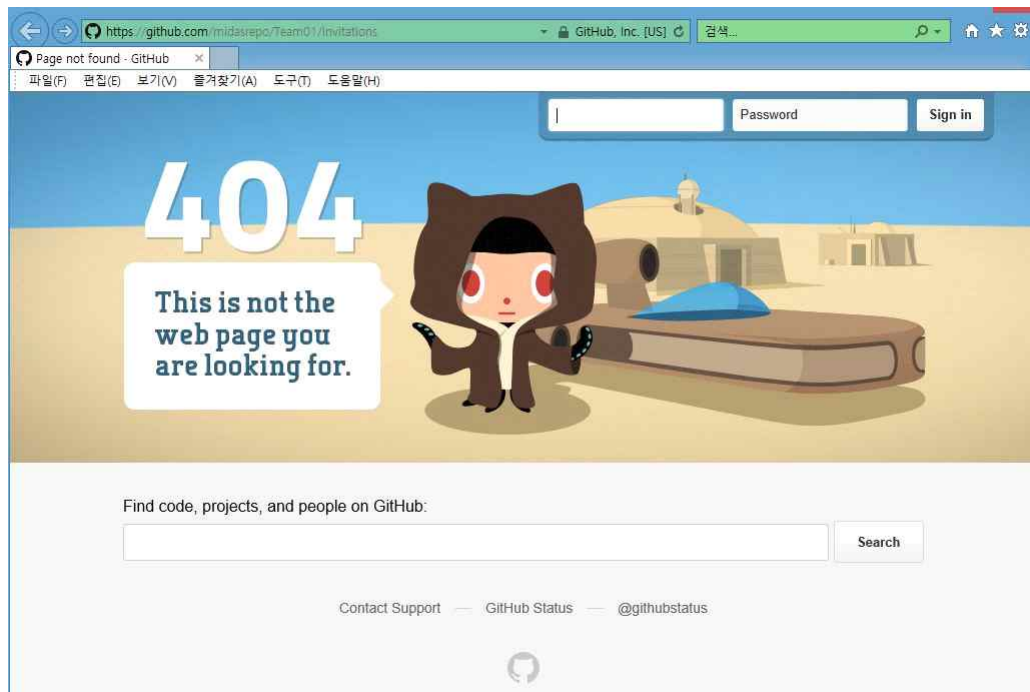
협업자 초대가 완료되면 다음 그림과 같이 초대한 협업자가 리스트에 나타나며 협업자 상태가 “Pending Invite” 상태로 표시된다.



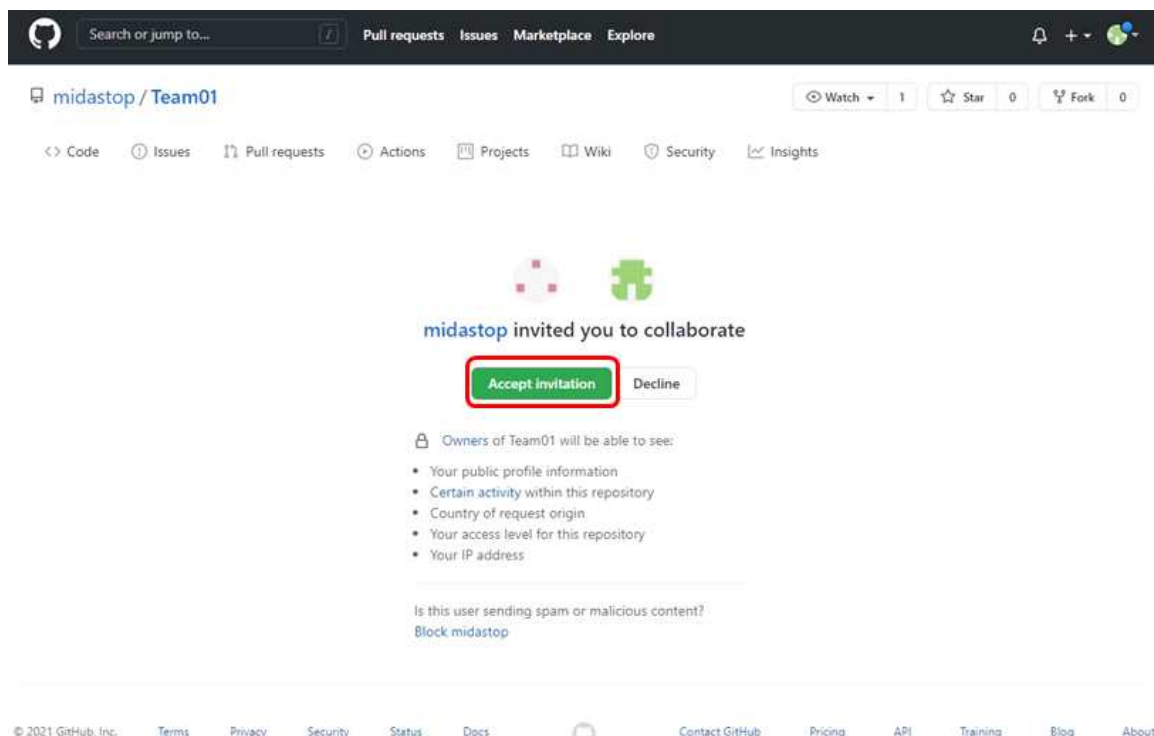
프로젝트 협업자 등록이 완료되면 GitHub에서 다음과 같이 해당 아이디의 이메일 계정으로 메일을 보내준다. 반드시 “View invitation”을 클릭해서 메일을 확인해야 정상적으로 협업이 가능하다.



“View invitation”을 클릭하여 초대에 응답하면 GitHub 사이트로 이동하지만 바로 인증 화면이 표시되는 것이 아니라 GitHub의 404 페이지가 표시된다. 이렇게 404페이지가 뜨는 이유는 로그인 상태가 아니기 때문에 발생하는 문제로 이 페이지에서 아이디, 비밀번호를 입력하고 로그인을 하게 되면 초대에 응답할 수 있는 화면으로 이동할 수 있다.

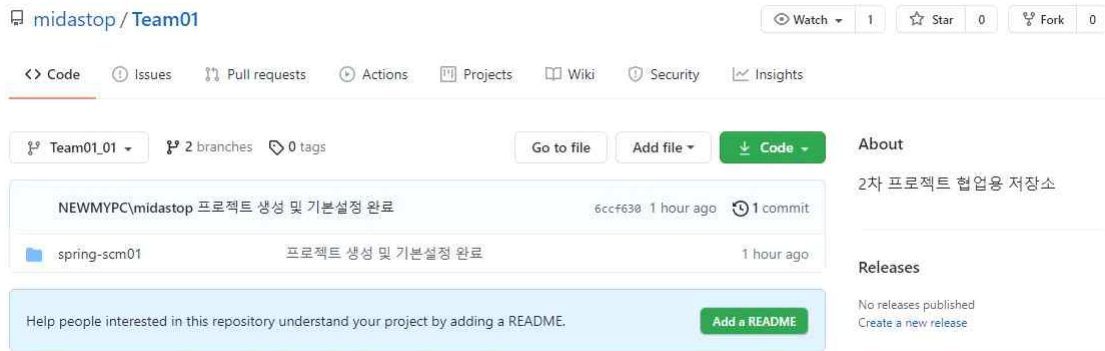


위의 화면에서 로그인을 하게 되면 아래와 같이 초대에 응답할 수 있는 화면이 나타나는데 이 화면에서 “Accept invitation” 버튼을 클릭하면 된다.

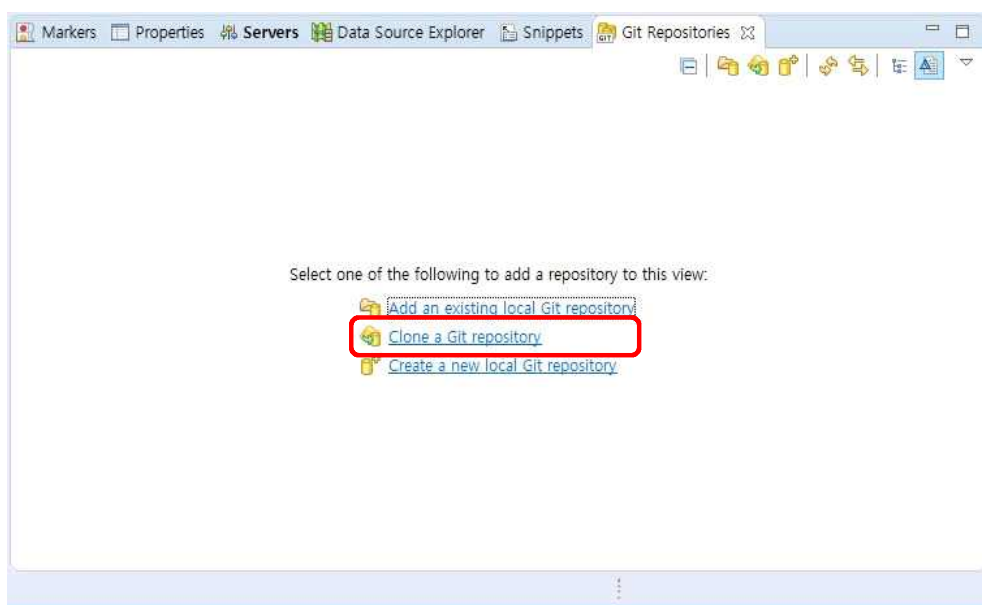




앞에서 “Accept invitation” 버튼을 클릭해 초대에 응답하게 되면 인증이 완료되고 아래와 같이 프로젝트 저장소 메인 화면으로 이동한다.

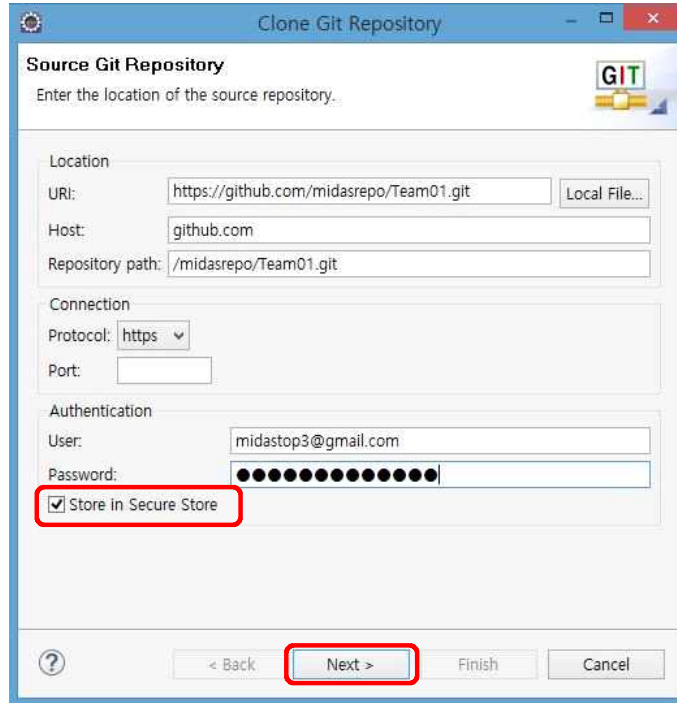


인증이 완료되면 Eclipse를 실행하고 교안 15페이지를 참고해 GitHub에 생성한 프로젝트 저장소 주소를 클립보드에 복사한다. 그리고 Eclipse의 “Git Repositories” 뷰에서 아래 그림과 같이 “Clone a Git repository”를 클릭하거나 GitHub 주소가 클립보드에 복사된 상태에서 “Git Repositories” 뷰를 선택하고 Ctrl + V로 붙여넣기 하면 “Clone Git Repository” 대화상자가 화면에 나타난다.

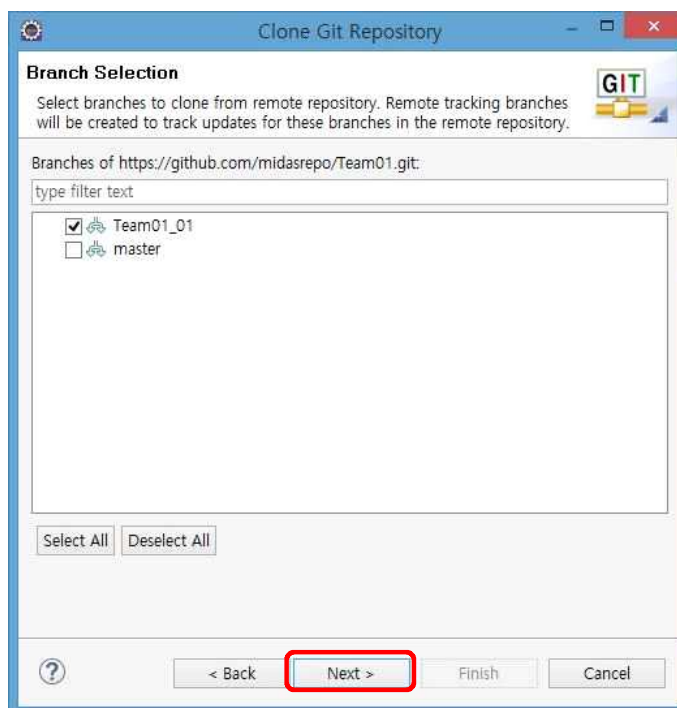


“Clone Git Repository” 대화상자에서 “Clone URI”를 선택하고 “Next” 버튼을 클릭하면 다음과 같은 “Source Git Repository” 입력 화면이 나타나는데 이 화면에서 URI에 GitHub 저장소 주소를 입력하면 나머지는 자동완성 된다. 아래 그림에서와 같이 “Store in Secure Store”를 선택하고 사용자 아이디와 비밀번호를 입력하고 “Next” 버튼을 클릭해 다음으로 이동한다.

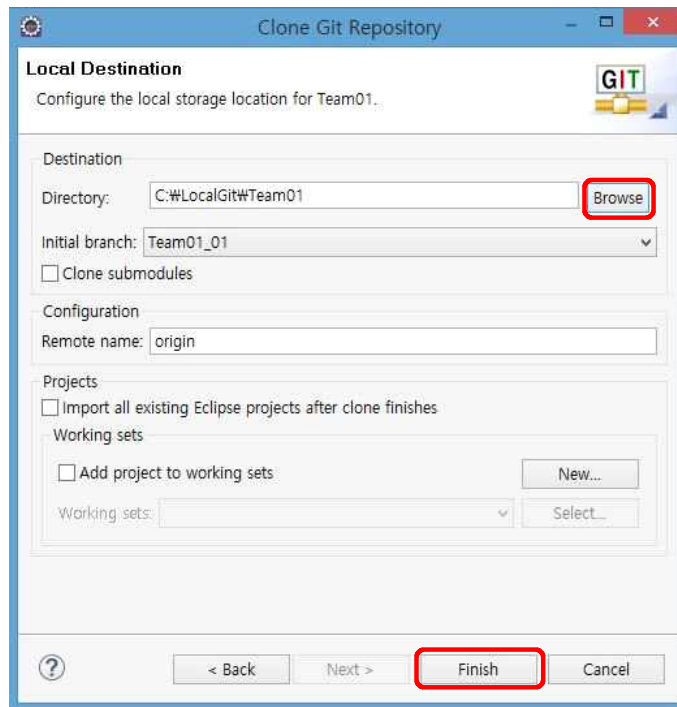




“Branch Selection” 화면에서 “master”는 아래 그림과 같이 해제하고 “Next” 버튼을 클릭해 다음으로 넘어간다.

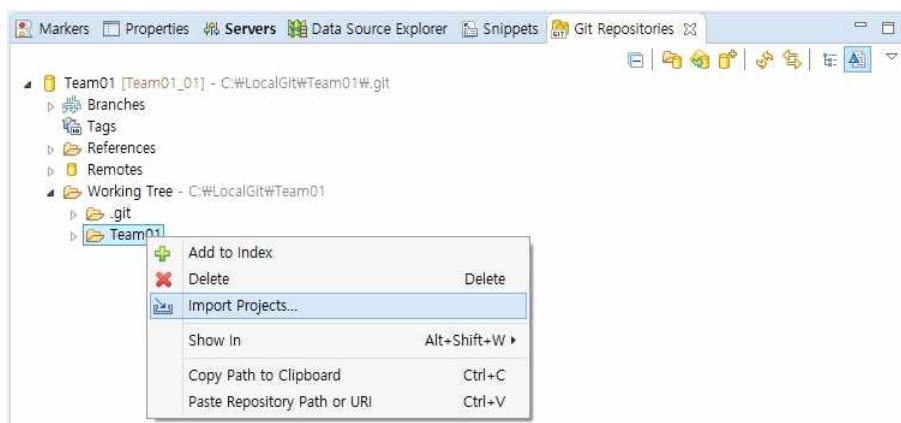


로컬 저장소를 지정하는 “Local Destination” 화면에서 “Browse”를 클릭해 아래 그림의 Directory 입력란에 입력된 텍스트와 같이 로컬 저장소를 지정하고 “Finish” 버튼을 클릭한다. 참고로 Directory 입력란에 로컬 폴더를 선택해 입력할 때는 저장소 이름을 제외하고 한 단계 상위 폴더를 지정하면 된다.

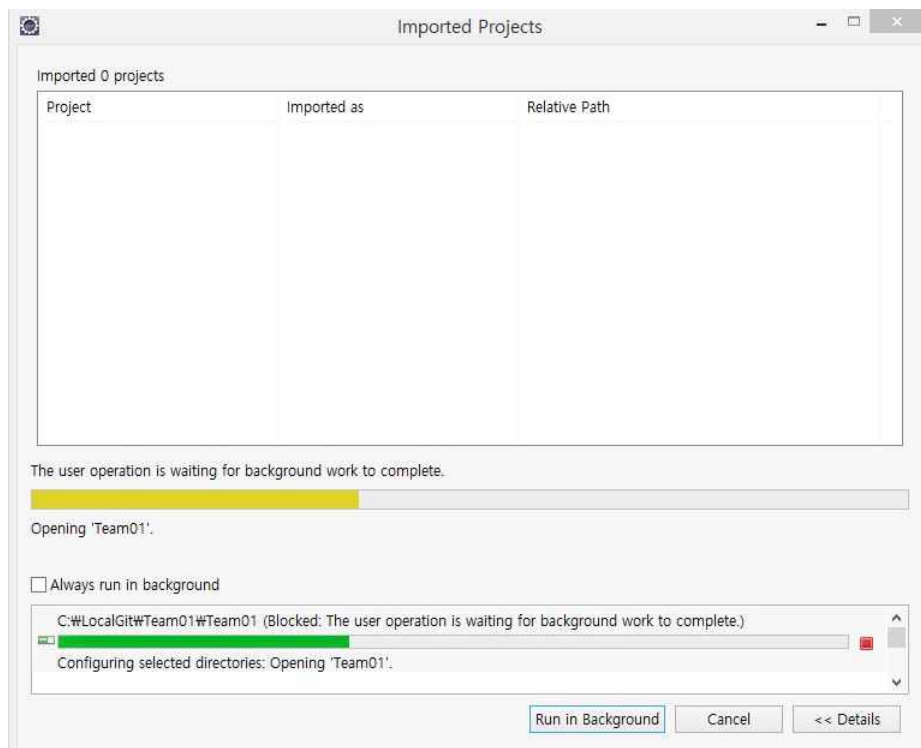
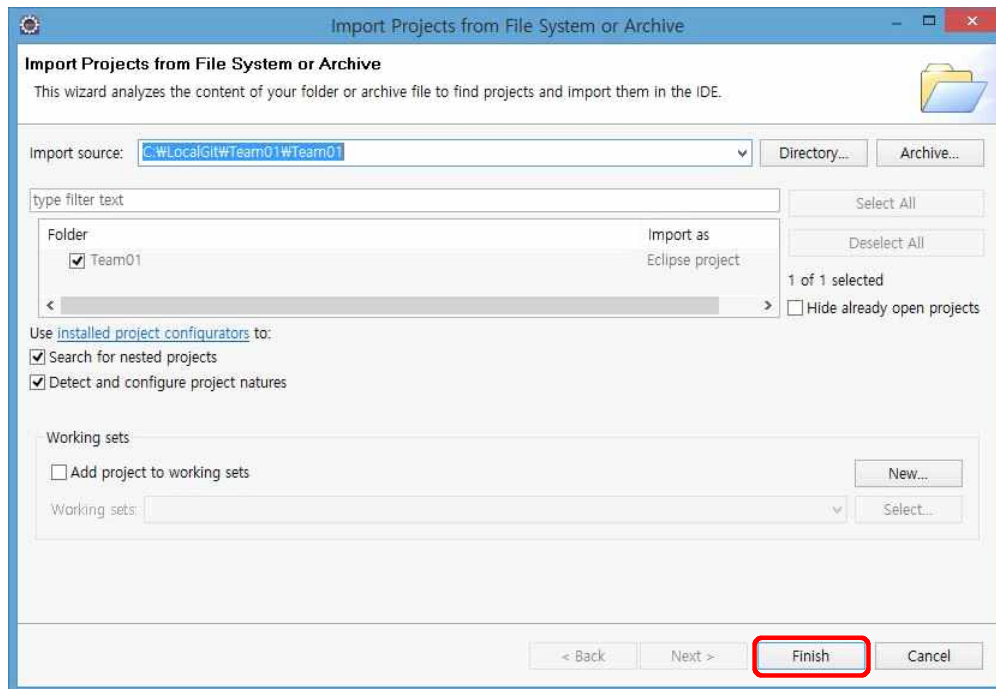


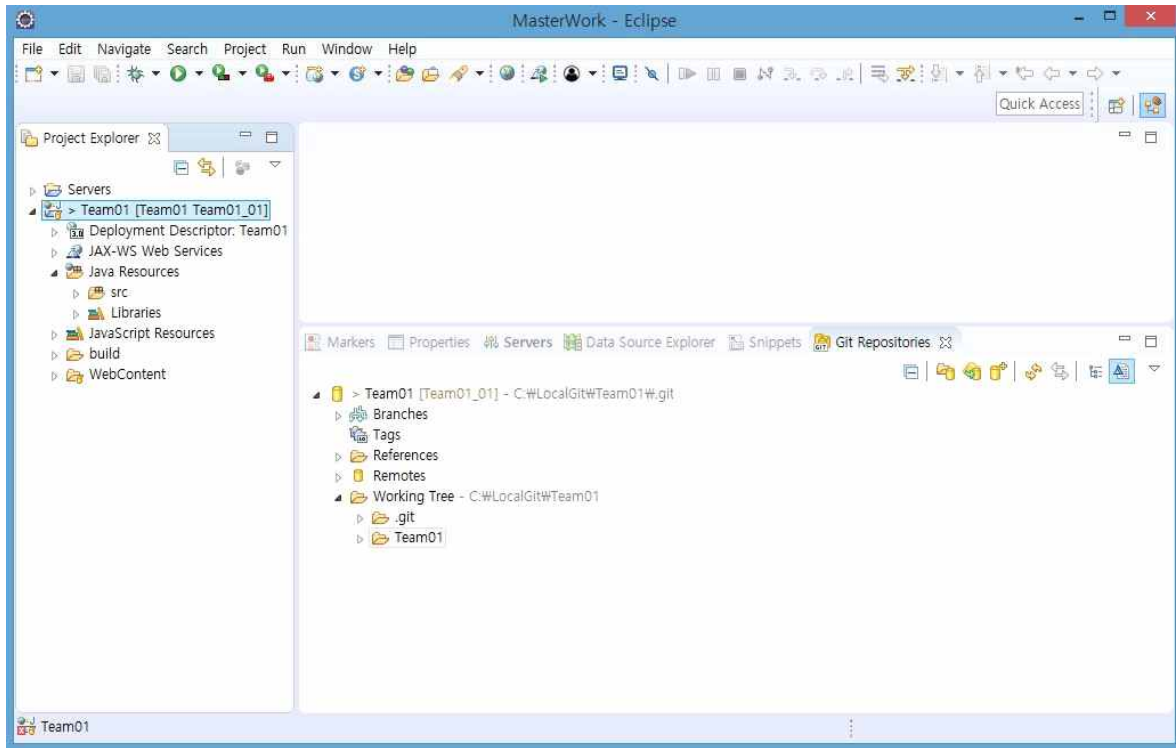
“Git Repositories” 뷰를 보면 아래와 같이 저장소가 생성된 것을 확인할 수 있다.

GitHub 저장소에 있는 프로젝트를 개인의 로컬 저장소로 Clone은 했지만 Eclipse로 가져오지 않았기 때문에 프로젝트를 Import 해 줘야 한다. 저장소 트리를 펼치면 아래 그림과 같이 앞에서 추가한 프로젝트 폴더를 볼 수 있는데 이 프로젝트 폴더에 마우스 우 클릭해 “Import Projects...” 메뉴를 선택하면 프로젝트를 Eclipse로 Import 할 수 있다.



앞의 화면에서 “Import Projects...” 메뉴를 선택하면 아래와 그림과 같이 “Import Projects from File System or Archive” 화면이 나타난다. 이 화면에서 프로젝트 경로가 맞는지 확인하고 “Finish” 버튼을 클릭하면 다음 그림과 같이 프로젝트 Import가 이루어지고 잠시 후에 Eclipse의 “Project Explorer”에 프로젝트가 추가된 것을 확인할 수 있다.





GitHub 저장소에 있는 프로젝트 Clone이 완료되고 “Git Repositories” 뷰의 Repository를 펼쳐보면 다음 그림과 같이 Branches에 Local과 Remote Tracking에 각각의 저장소가 보이게 된다. Local은 사용자 개인 컴퓨터의 로컬 저장소에 만든 Branch이고 Remote Tracking은 GitHub에 만든 Branch 이다.

아래 그림을 살펴보면 Branch 이름 앞에 체크 표시가 있는 Branch가 있는데 이 체크 표시가 있는 Branch가 현재 작업하고 있는 Branch 이므로 Local Branch가 체크 표시가 되어 있는지 확인하자.



프로젝트의 WebContent/NewFile.jsp를 추가하고 이 JSP 파일을 간단히 편집하고 저장 한다.  
수정된 소스 코드를 GitHub에 반영하기 위해 아래 그림과 같이 프로젝트에 마우스 우 클릭해 Team - Add to Index 메뉴를 선택한다.  
Add to Index 메뉴는 GitHub 저장소에 업로드 할 파일을 선택하는 과정으로 이 외의 Git Staging 뷰를 통해서 GitHub에 업로드 할 파일들을 선택해서 Commit 할 수도 있다.  
실제 수정된 파일을 GitHub에 업로드 하는 과정은 본 교안 16페이지를 참고하길 바란다.

