[프로젝트 버전 관리 – git 사용법] 버전 관리 시스템이란?

문서를 작성할 때 '보고서1.docx', '보고서1_수정.docx', '보고서1_최종.docx' 와 같이 문서를 저장하거나 프로그램 소스를 수정할 경우 오류가 날 경우를 대비하여 소스나 문서 사본을 백업하게 되는데, 이 때 하나하나의 저장 파일이나 백업본을 개별 버전으로 간주하고 그것을 관리하는 방법을 말한다.
즉, 버전 관리 시스템은 사본 생성, 보존, 복원을 한 번에 해 줄 수 있는 도구이다.

1. 버전 관리 시스템의 종류
   A. 클라이언트-서버 모델
      하나의 중앙 저장소를 공유한 후 각각의 클라이언트(개발자)는 저장소의 일부분만 갖는 형태이다.
      즉, 자신이 작업하는 부분만 로컬에 임시로 저장한 후 작업하는 형태이다.

   B. 분산 모델
      프로젝트에 참여하는 모든 클라이언트가 전체 저장소에 대한 개별적인 로컬 저장소를 갖고 작업하는 형태이다.
      '클라이언트-서버' 모델과 달리 '분산' 모델의 클라이언트는 각자의 온전한 전체 저장소의 사본을 로컬에 가지게 된다.

   C. CVS
      Concurrent Versions System 을 의미하는 말로 클라이언트-서버 방식의 버전 관리 시스템이다.
      공식 웹 사이트 : https://savannah.nongnu.org/projects/cvs

   D. 서버버전
      CVS의 여러 단점을 개선한 '클라이언트-서버' 모델의 자유 소프트웨어 버전 관리 시스템이다.
      공식 웹 사이트 : https://subversion.apache.org/

   E. 머큐리얼
      현재 구글에서 이 시스템의 관리를 지원하고 있는 분산 모델의 버전 관리 시스템이다. Git은 필요한 기능을 골라서 사용하지만 머큐리얼은 버전 관리 시스템에 필요한 모든 기능을 한 번에 통합해서 제공한다. 파이썬으로 개발되었다는 특징이 있다

2. Git

A. Git 의 장점

① 전 세계의 수많은 사용자가 사용 중

② Git을 사용한 저장소로 공유 사이트인 GitHub 웹 사이트의 존재

③ 사용자 수에서 나오는 어마어마한 숫자의 튜토리얼과 프로젝트가 존재

B. Git 의 특징

① 로컬 및 원격 저장소 생성

② 로컬 저장소에 파일 생성 및 추가

③ 수정 내역을 로컬 저장소에 제출

④ 파일 수정 내역 추적

⑤ 원격 저장소에 제출된 수정 내역을 로컬 저장소에 적용

⑥ Master 에 영향을 끼치지 않는 브렌치 생성

⑦ 브렌치 사이의 병합(Merge)

⑧ 브렌치를 병합하는 도중의 충돌 감지

[Git 설치]

공식 사이트인 https://git-scm.com/ 에서 오른쪽 화면에 보이는 'Downloads for Windows'을 클릭하여 다운로드 페이지로 이동한다.

※ 현재 접속중인 컴퓨터의 운영체제에 따라 자동으로 <Downloads for Linux>, <Downloads for Mac>등으로 달라진다.
※ Git에 대해 더 알고 싶다면 https://git-scm.com/book/ko/v2 사이트를 참조하면 된다.

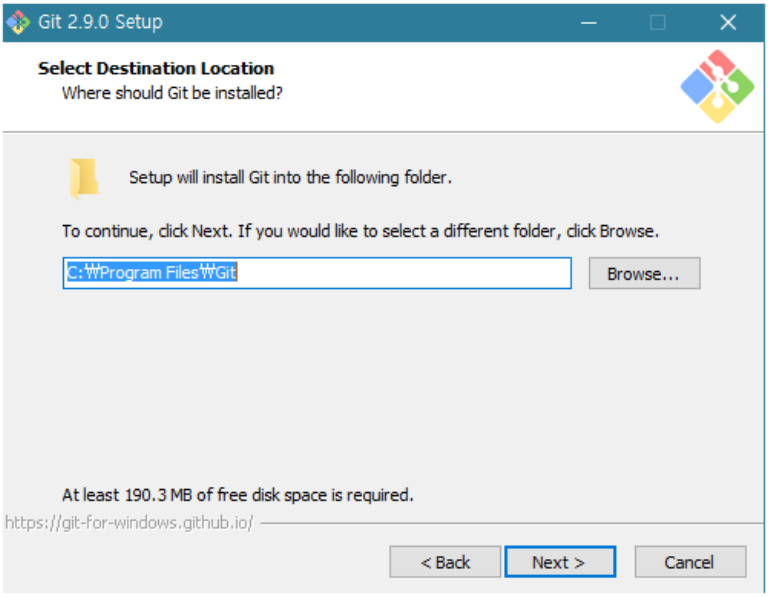
- 밑에 자동으로 자신의 버전과 맞는 최신 버전이 뜬다. 다른 버전을 다운로드 받고 싶다면 취소를 누르고 다운로드 받으면 된다.

※ Git은 안정성이 검증되지 않았기 때문에 실제 다운로드 페이지에서 표기된 최신 버전보다 더 옛날 버전을 사용한다.

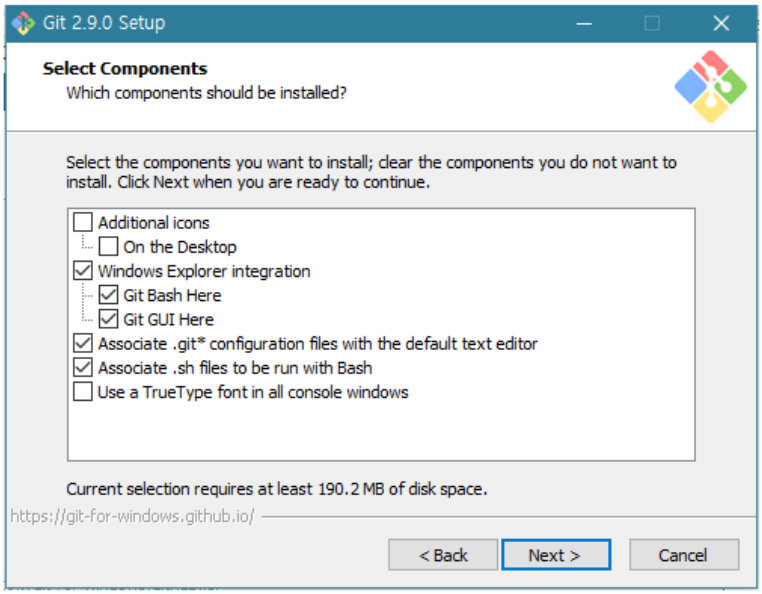
1. 윈도우에서 설치

- <Next> 클릭

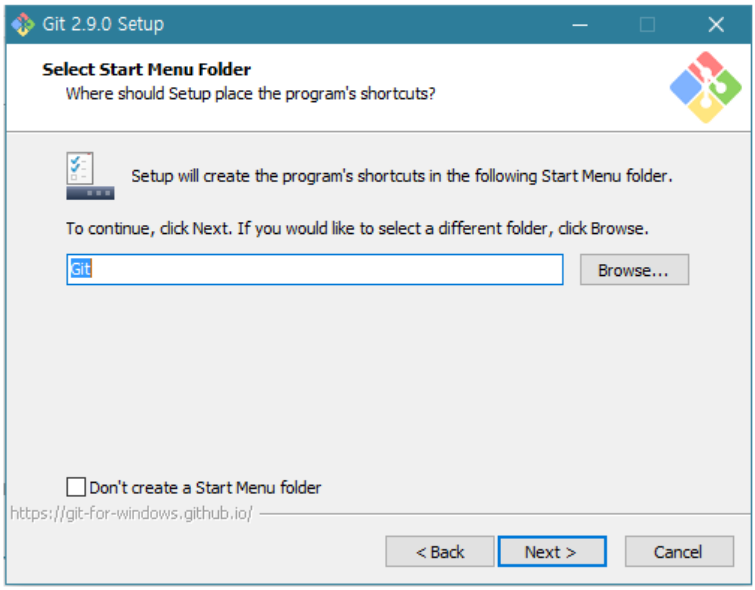
- <Next> 클릭



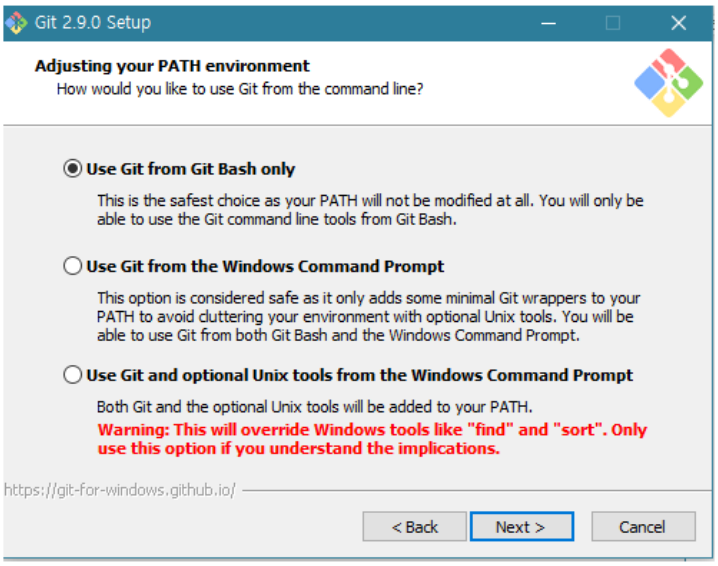
- 추가로 추가할 옵션이 있다면 선택하고 Next 클릭



- <Next> 클릭

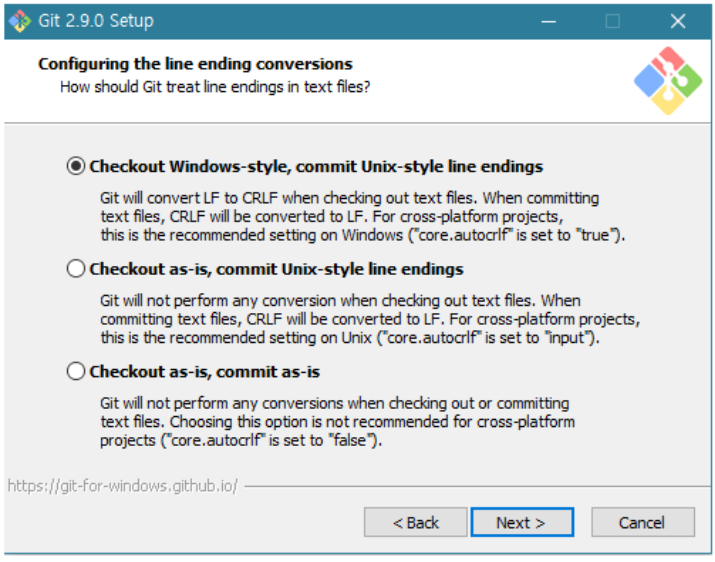


- 환경 변수를 맨 위로 체크하고 <Next> 클릭

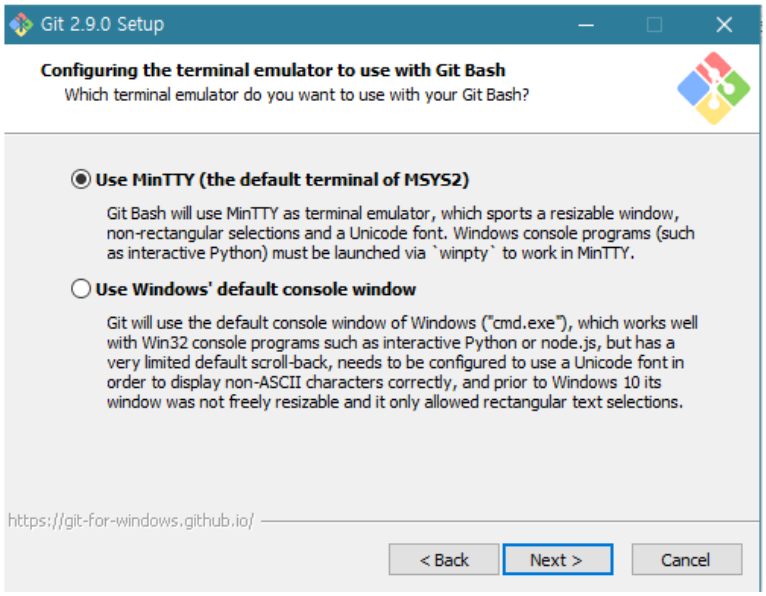


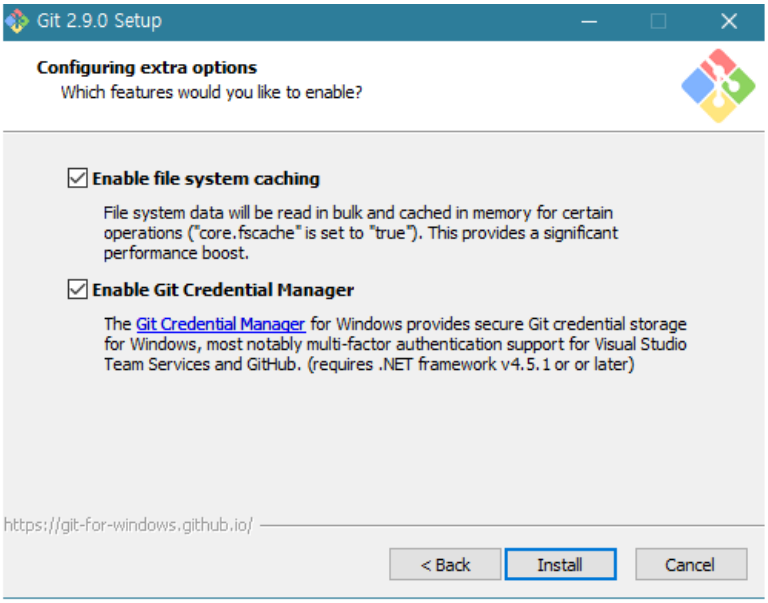
※ Use Git from Git Bash only : 환경 변수를 변경하지 않고 Git bash 커맨드 라인 도구에서만 실행한다.
※ Use Git from Windows Command Prompt : 윈도우 커맨드 프롬프트에서 Git을 실행할 수 있는 최소한의 내용을 환경 변수에 추가해 설치한다.
※ Use Git from optional Unix tools from the Windows Command Prompt : Git과 부수적인 UNIX 도구들을 모두 윈도우 환경 변수에 추가한다.

- Git에서 커밋할 때 작성하는 라인 끝을 어떻게 처리할 지 결정한다. 기본으로 선택된 첫 번째 옵션으로 두고 <Next> 클릭

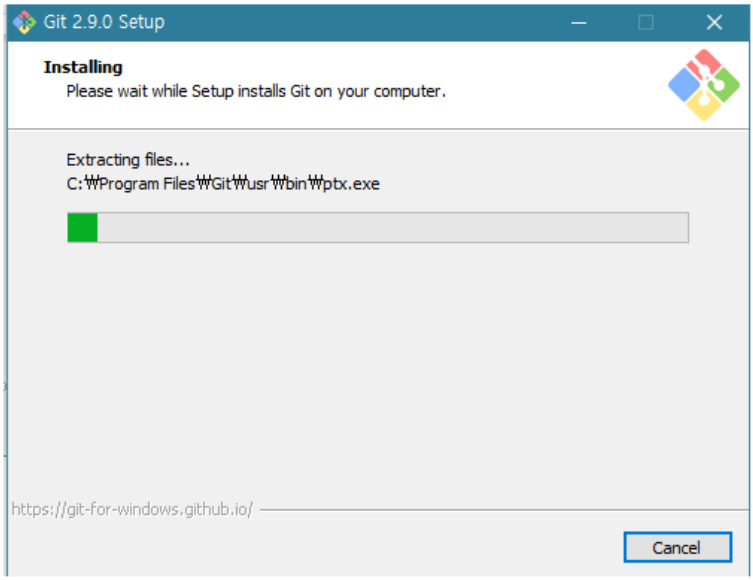


- <Next> 클릭


Git 2.9.0 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://git-for-windows.github.io/
< Back
Next >
Cancel

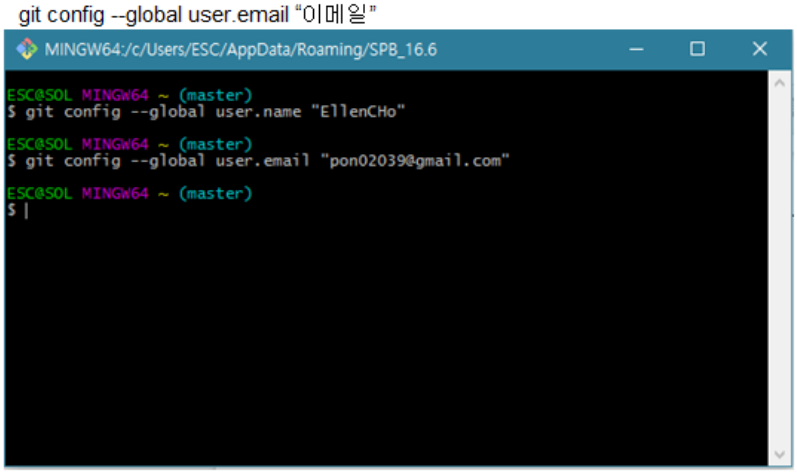

Git 2.9.0 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable Git Credential Manager
The Git Credential Manager for Windows provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or or later)
https://git-for-windows.github.io/
< Back
Install
Cancel

- 설치



- 시작메뉴 -> Git -> Git Bash 를 실행하여 사용자 이름과 이메일을 설정한다.

```
git config --global user.name "사용자 이름"
git config --global user.email "이메일"
```



- 설정 완료

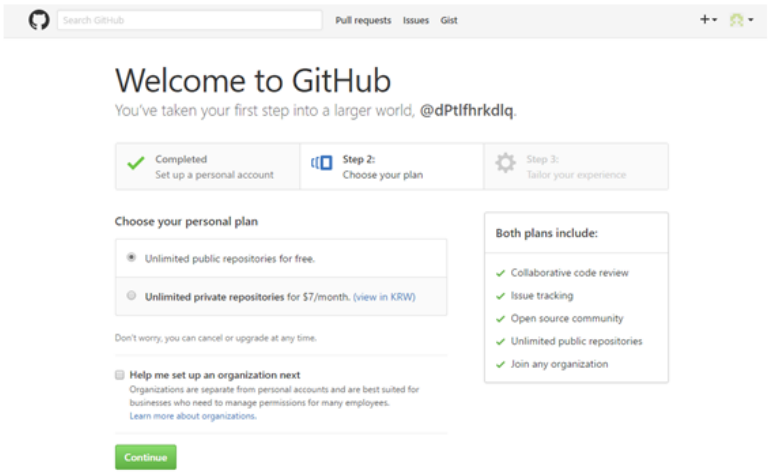
[원격저장소 GitHub]

GitHub는 원격 저장소를 제공하며 여러가지 프로젝트 진행을 원활하게 하는 도구를 함께 제공한다.

1. 회원 가입

- 공식 사이트인 https://github.com/ 에 접속해서 오른쪽에 사용자 아이디, 이메일, 비밀번호를 치고 <Sign up for GitHub>를 클릭한다.

- 지불하는 비용에 따라 비공개 저장소를 사용할 수 있다. 무료로 GibHub를 이용한다면 모든 저장소는 공개 저장소가 된다. 맨 아래 기본으로 선택된 'Free'항목으로 그대로 두고 <Finish sign up> 버튼을 누른다.



※ 'Help me set up an organization next' 체크 박스는 많은 사람이 협업할 때 팀을 만들어서 활동하도록 설정하겠다는 의미이다.

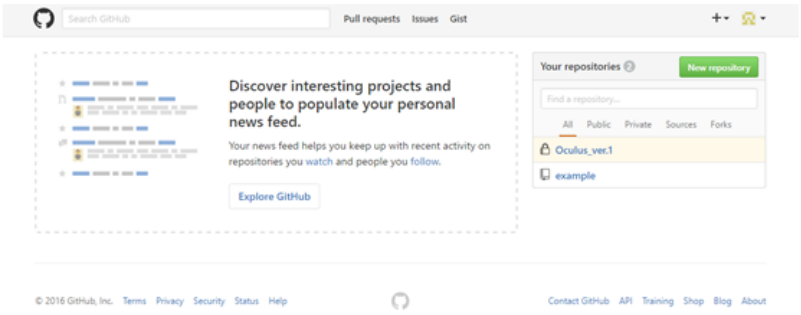
- 회원 가입할 때 입력했던 이메일을 확인해서 GibHub 가입을 인증한다.

- 회원 가입 완료

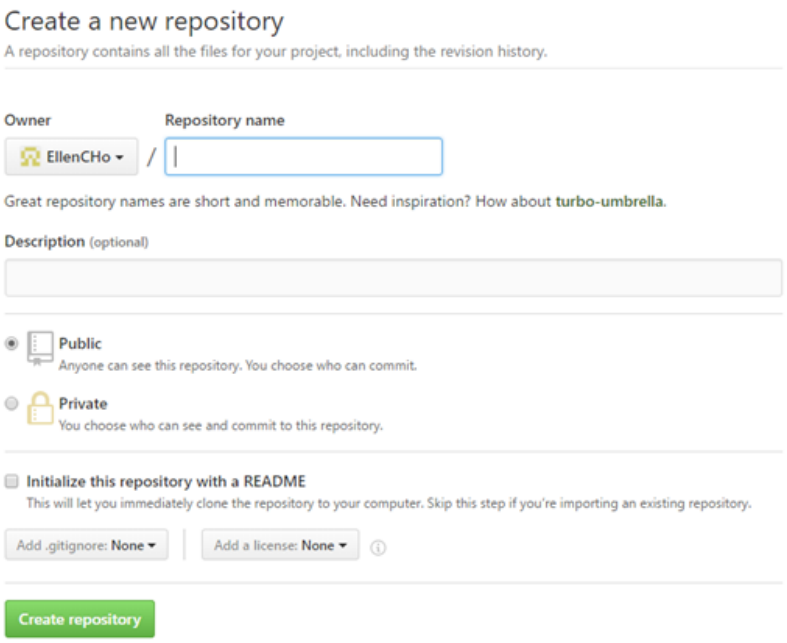
2. 원격 저장소 생성

- 포크 : 다른 사람의 저장소를 내 저장소로 복사하는 기능
- 풀 리퀘스트 : 포크한 저장소를 수정해 다시 원본 저장소에 병합해달라는 요청을 보내 사용자 사이의 상호작용을 일으키는 기능
- 이슈 : 저장소 안에서 사용자들 사이의 문제를 논의하는 기능
- 위키 : 저장소와 관련된 체계적인 기록을 남기는 기능

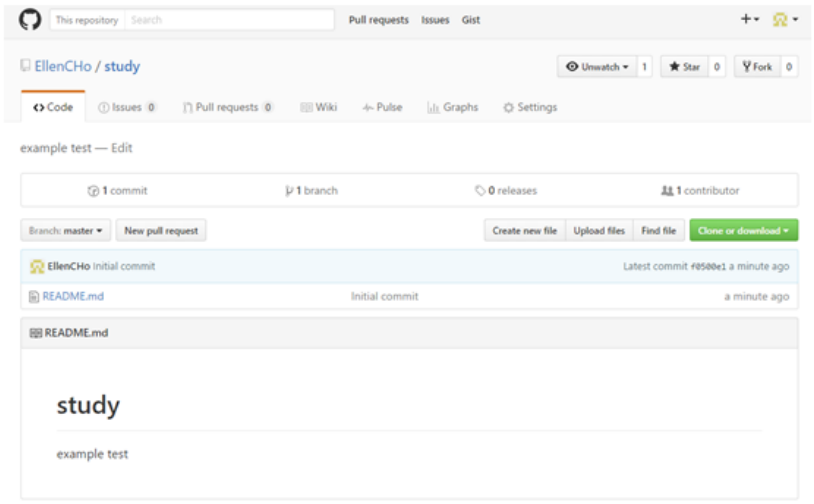
- 오른쪽 아래에 있는 <New repository>를 클릭해서 새 원격 저장소를 생성한다.



- 아래 항목을 참고해서 빈칸을 채운다.
  - Owner : 사용자의 아이가 표시된다. 협업 환경에서는 다른 사용자의 아이디를 지정 할 수 있다.
  - Repository name : 새로 생성할 원격 저장소의 이름을 입력한다. 가능하면 로컬 환경에서 작업할 Git 프로젝트 디렉터리 이름과 같게 하는 것이 좋다.
  - Description : 꼭 작성할 필요는 없는 항목이지만 생성한 원격 저장소가 어떤 역할을 하는 지를 간단하게 적어두면 원격 저장소가 많아졌을 때 구분하기 쉽다.
  - Public/Private : 원격 저장소의 공개 여부를 선택하는 옵션이다. 무료 사용자는 Public만 사용 가능하다.
  - Initialize this repository with a README : 기본적으로는 체크 표시를 해준다. 체크해주면 GitHub에서 생성한 원격 저장소를 바로 로컬 저장소에 복사해서 가져올 수 있다. 또한 '저장소 이름'과 'Description' 항목의 내용을 담은 README.md 파일을 생성한다.
  - Add .gitgnore : 원격 저장소에 포함하지 않을 파일들의 목록을 만들 때 사용한다. 지금 당장은 사용할 필요가 없으니 'none' 상태로 둔다.
  - Add a license : 원격 저장소에 저장할 프로젝트가 어떤 라이선스에 속할지를 선택한다. 지금 당장은 사용할 필요가 없으니 'none' 상태로 둔다.



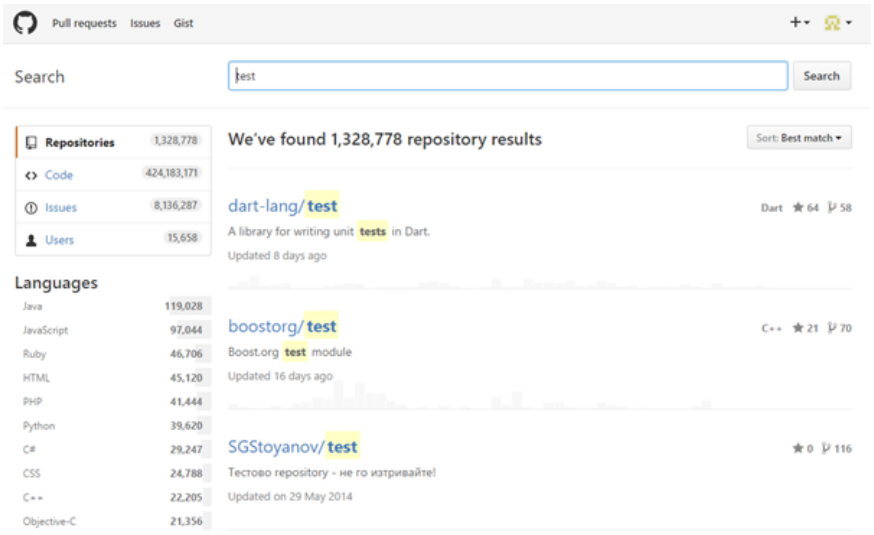
- <Create repository>를 클릭하면 아래와 같은 원격저장소가 생성된다.



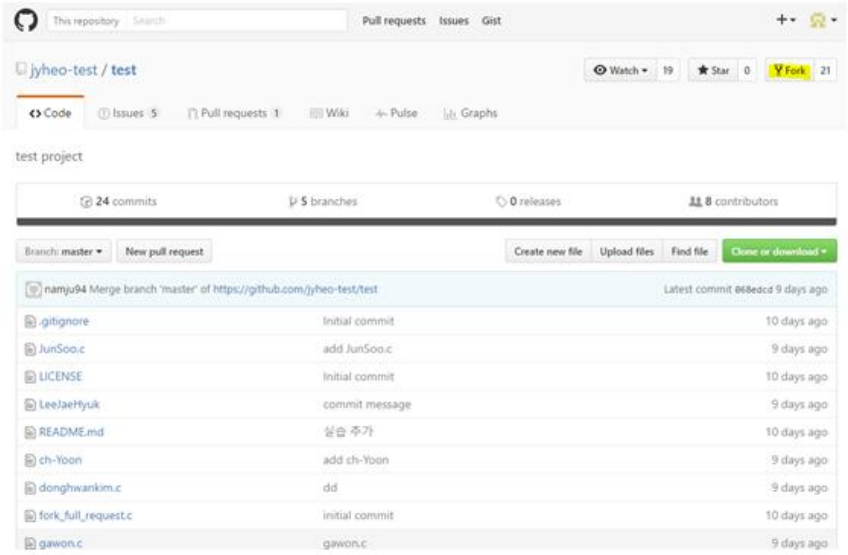
3. 포크

다른 사람의 원격 저장소를 내 계정으로 복사하는 방법을 포크라고 한다. 만약 포크하지 않는다면 내 것이 아닌 저장소 다시말하면 쓰기 권한이 없는 원격 저장소를 사용하는 것이므로 자유롭게 파일을 생성하거나 수정하여 원격 저장소에 반영하는 것이 불가능하다.

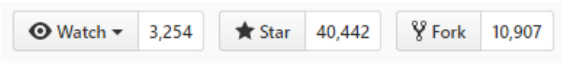
- 원격 저장소 검색(ex. test)



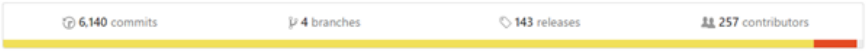
- 오른쪽에 있는 <Fork>를 클릭하면 내 원격 저장소에 저장됨



4. GitHub 원격 저장소의 구조

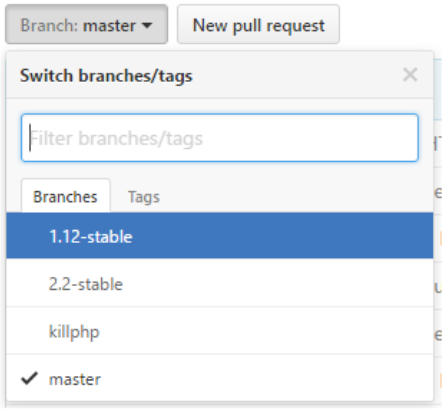


- Watch : 해당 버튼을 클릭하면 원격 저장소의 활동 내역을 사용자에게 알려준다. 댓글이나 이슈 등에서 언급될 때만 알려주는 Not Watching, 모든 활동 내역을 알려주는 Watching, 모든 알림을 무시하는 Ignoring을 선택할 수 있다. 오른쪽 숫자는 현재 활동 내역을 보고 있는 사람의 수이다.
- Star : 해당 원격 저장소에 관심이 있을 때 클릭하면 된다. 오른쪽 숫자는 관심이 있는 사람의 수를 나타낸다.
- Fork : 해당 버튼을 클릭하면 원격 저장소를 포크한다. 오른쪽은 포크한 사람의 수를 나타낸다.



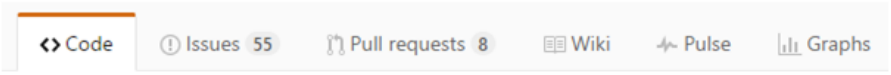
- Description : 원격 저장소를 설명하는 메시지가 나타난다.
- Commits : 원격 저장소의 총 커밋 수를 나타낸다.
- Branches : 원격 저장소의 브랜치 수를 나타낸다.
- releases : 원격 저장소의 태그 수를 나타낸다. 주로 특정 버전에 표식을 주고 싶을 때 사용한다. 이 표식을 통해서 특정 버전을 다운로드할 수 있다.
- Contributor : 원격 저장소에 커밋 혹은 풀 리퀘스트가 받아들여진 사용자 수이다. 이 저장소가 오픈 소스라면 이 오픈 소스에 공헌한 사람 수라고 생각해도 된다.

※ 커밋은 뒤에 나오지만 저장할 때마다 끼워넣는 책갈피라고 생각하면 쉽다. 지금 저장한 버전이 무엇을 수정했는 지, 무슨 내용을 담고 있는 지 설명해주는 지표이다.



- Current branch : 원하는 브랜치를 선택하는 기능

- Current branch : 원하는 브랜치를 선택하는 기능



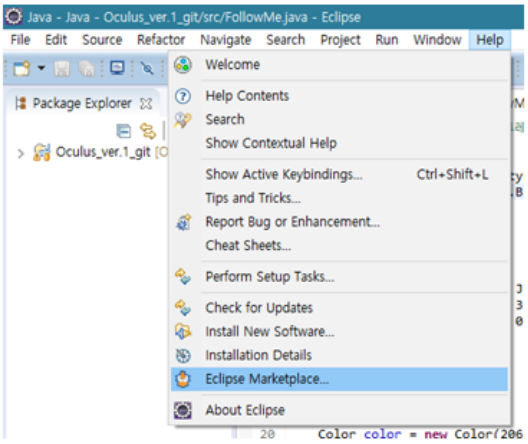
- Code : 해당 원격 저장소의 루트 디렉토리로 이동한다. 어떤 경로에 있더라도 루트 디렉터리로 이동한다.
- Issues : 해당 원격 저장소의 주요 이슈 사항을 기재한 후 관리합니다. 게시판 형태이며 댓글 형태로 토론이 이뤄지기도 한다. 보통은 문제점이나 개선점을 얘기한다.
- Pull Requests : 풀 리퀘스트 전체 목록을 모아서 보여준다. Issues와 마찬가지로 목록마다 댓글 형태로 토론할 수 있다. 보통 오른쪽에 있는 숫자는 현재 요청이 온 풀 리퀘스트를 받아들일 것인지에 대한 논의가 몇개인지 알려주는 기능이다.
- Wiki : 공유할 정보나 개발 문서, 참고 자료 등을 작성하기 위한 기능입니다. 마크 다운과 위키위키 문법을 사용한다.
- Pulse : 해당 원격 저장소의 최근 변경 내역을 확인할 수 있다. 최대 한 달까지의 변경 내역을 확인할 수 있으며 풀 리퀘스트 몇 개중에 몇 개가 받아들여졌나, 이슈는 몇 개가 있고 몇 개가 해결되었나, 해당 풀 리퀘스트와 이슈에 관련된 활동을 볼 수 있다.
- Graphs : 공헌자의 공헌 내역, 커밋 수 등 해당 저장소의 활동 내역을 그래프화해서 보여준다.

[이클립스에서의 Git 사용법(egit)]

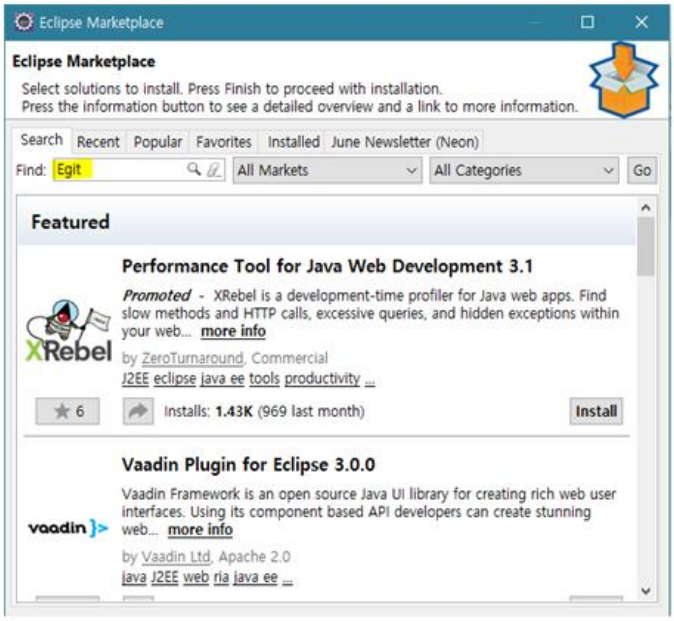
설치한 Git bash를 이용하여 git을 사용할 수 있지만 우리는 이클립스에서 제공하는 egit을 이용하여 git을 사용한다. 이클립스 말고도 Visual Studio, IntelliJ IDEA, Xcode에서 제공하는 툴로 git을 사용할 수 있다.

1. EGit 설치

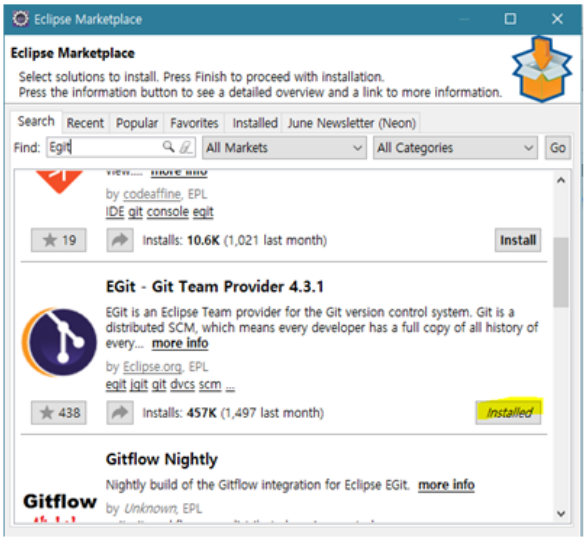
- 이클립스를 실행한 뒤 [Help] -> [Eclipse Marketplace]를 클릭한다.



- 마켓 플레이스의 'Find'항목에 Egit을 검색한다.



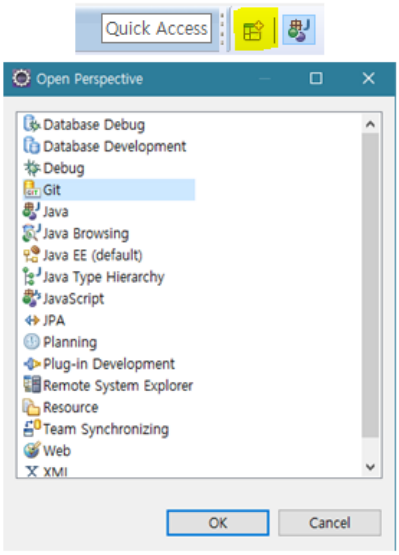
- Egit을 설치해준다. (2014버전인 이클립스 Luna부터는 EGit 플러그인이 기본으로 포함되어 있어 Installed나 update라고 뜨지만 2014버전보다 전 버전은 EGit이 깔려있지 않다.)



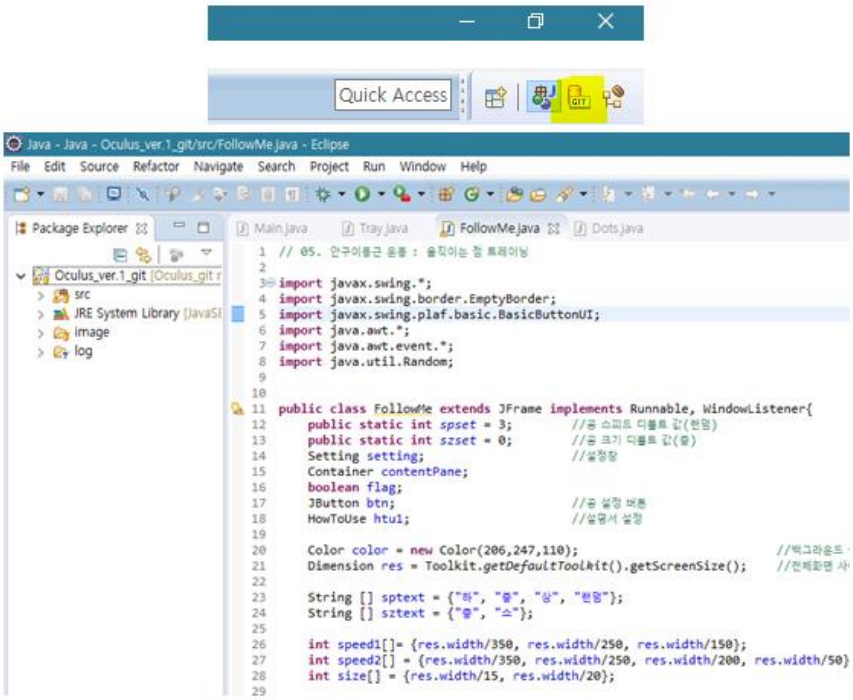
- 따라서 설치과정을 보여줄 수 없지만 설치할 때는 설치에 필요한 모든 항목을 선택한 후 <Confirm>을 클릭한다.

- 라이선스 검토 항목에서는 'I accept the terms of the license agreement'를 선택한 후 <Finish>를 클릭한다. 설치 후에는 이클립스를 다시 실행하겠냐는 메시지가 나타나는데 <Yes>를 클릭한다.

- 재실행 후에 이클립스의 오른쪽 위에 조그마한 영역으로 퍼스펙티브 항목이 보인다. 만약 최신 버전을 설치했다면 처음부터 Git 퍼스펙티브가 활성화되있지 않으므로 아래 사진에 보이는 오른쪽 위의 버튼을 클릭해 Git 퍼스펙티브를 선택한 후 <OK>를 눌러 활성화시킨다.



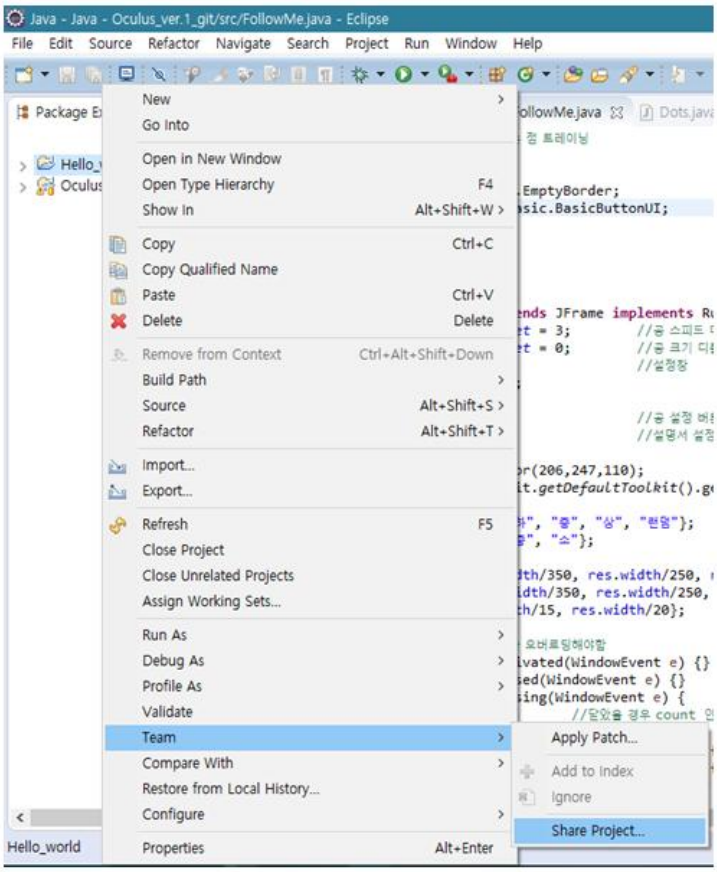
• Git 퍼스펙티브가 추가된 것을 볼 수 있으며 아이콘을 클릭하면 기본 화면이 Git 퍼스펙티브 형태로 변경된다



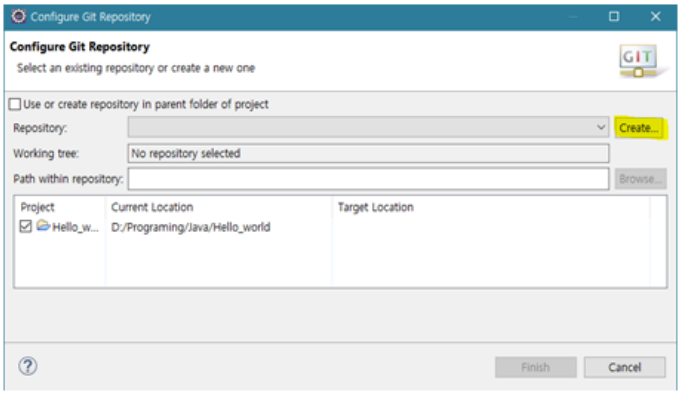
※ 원래 형태로 돌아가고 싶다면  노란색 형광펜이 칠해진 버튼을 누르면 된다.

2. 저상소 생성

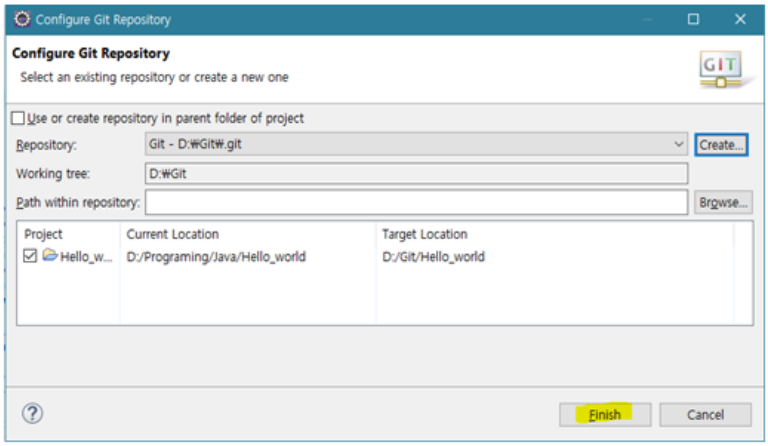
- 퍼스펙티브 항목에서 'Java'를 선택하고 새 프로젝트를 만든다. 새로 생성한 프로젝트에 마우스 오른쪽 버튼을 클릭하고 [Team] -> [Share Project] 클릭한다.



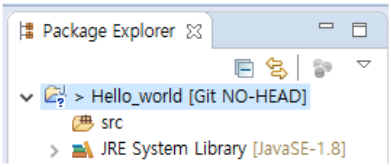
- 'Configure Git Repository'창이 열리면 <Create>를 클릭한다.



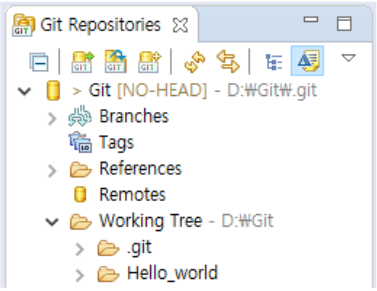
- Git 저장소를 지정하는 창이 나타난다. Git 저장소의 경로가 이클립스의 프로젝트 폴더의 경로와 달라야 하므로 <Browse>를 눌러 원하는 경로를 지정한 후 <Finish>를 누르면 'Configure Git Repository' 창의 저장소 위치, 작업 위치 등 저장소에 관련된 기본 설정이 다 되어있다. <Finish>를 누르면 저장소 생성이 완료된다.



- 프로젝트 루트를 보면 디렉터리 아이콘에 물음표 아이콘이 생기고 프로젝트 이름 옆에 간단한 저장소 정보가 표시된다.



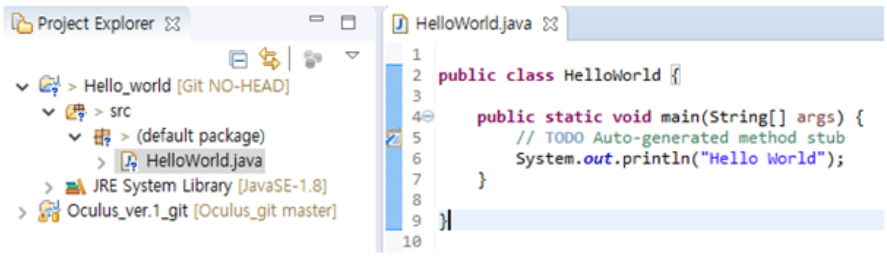
- 오른쪽 위 Git 퍼스펙티브 아이콘을 누르면 Git 저장소 정보를 확인할 수 있다. 'Working Directory'를 선택하고 하위 디렉터리를 살펴본다.



- 'Working Directory'의 하위 디렉터리가 '.git'과 '프로젝트 이름'으로 구성되었는 지 확인한다. 위와 같이 Git 저장소인 '.git' 디렉터리와 이클립스 프로젝트 디렉터리(.project 파일이 있는 디렉터리)를 분리함으로써 의도하지 않은 사고를 방지하고 여러 장점을 취할 수 있다.

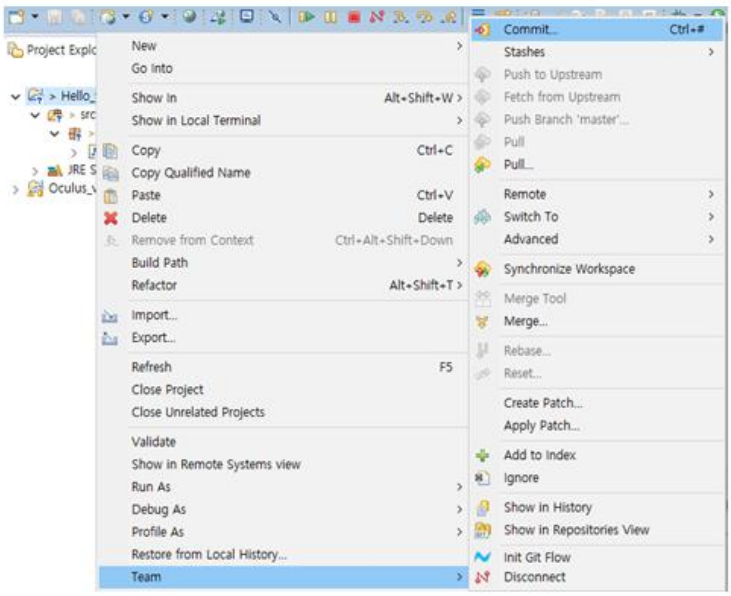
3. 첫 번째 커밋

- “Hellow World”를 출력하는 프로그램을 작성한다.

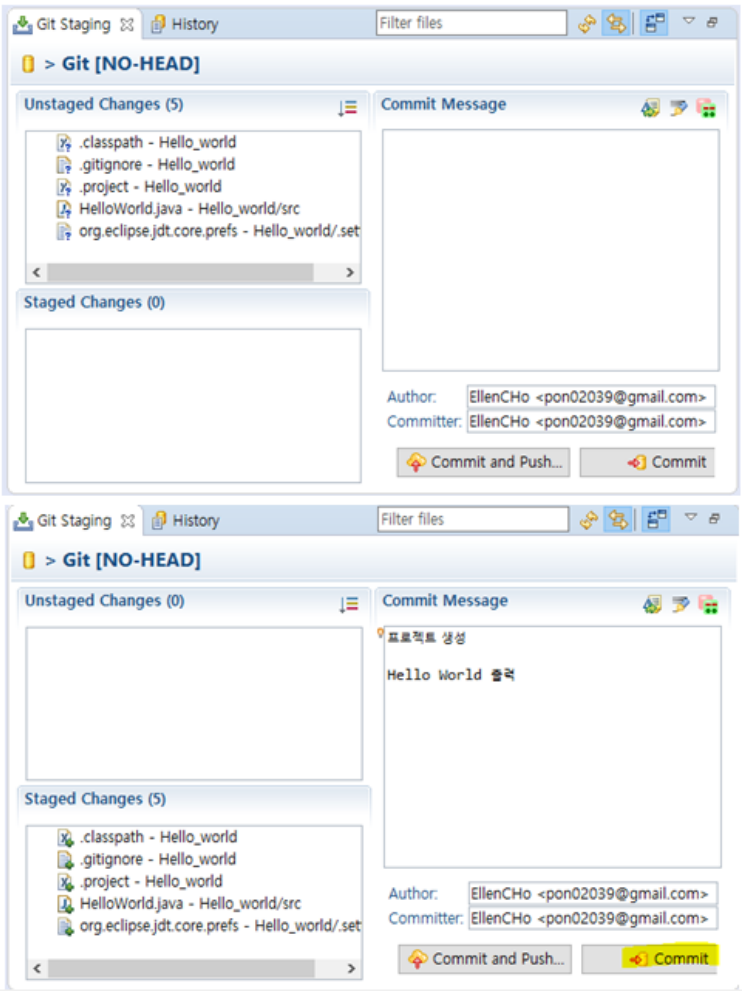


- 패키지 익스플로러에서 ‘HelloWorld.java’ 파일이 속한 트리 구조 전체에 물음표가 생긴 것을 볼 수 있다. 물음표는 저 파일들이 추적 중이 아니라는 뜻이며 파일을 Git 저장소에 추가해야 한다.

- 패키지 익스플로러의 프로젝트 루트(Hello_world[Git NO-HEAD])를 마우스 오른쪽 버튼으로 클릭한 후 [Team] -> [Commit]을 클릭한다.

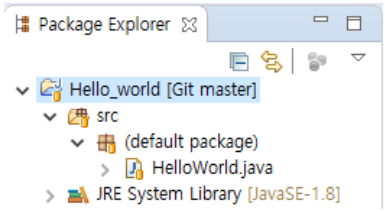


- 아래와 같은 창이 뜨면 ‘Commit message’항목에 알맞은 메시지를 작성한 후 방금 만든 프로젝트 파일 전체를 ‘Staged Changes’로 옮긴다. 그런 후에 마지막으로 <Commit>을 누른다.



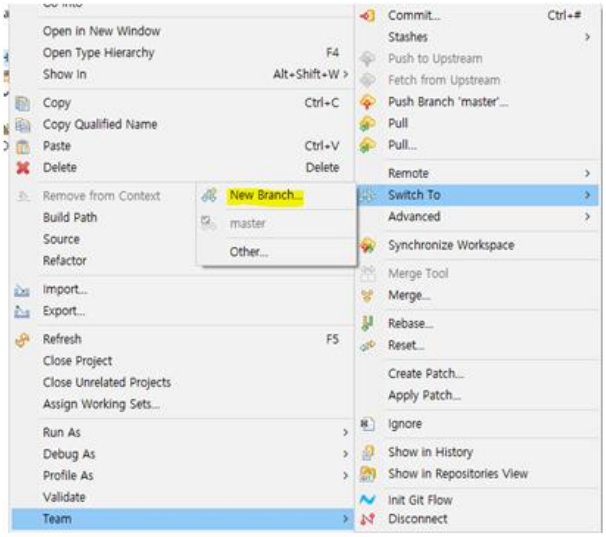
※ 만약 앞에서 Git bash로 기본설정을 하지않았다면 사용자 이름과 이메일 주소를 넣으라는 'Please identify yourself' 창이 나타날 수 있다. 이런 경우 사용자 이름과 이메일 주소를 입력한다.

- 커밋하고 나면 물음표가 사라지고 저장소가 추적 중이라는 의미로 노란 원통 아이콘이 붙는다. 파일 추가와 수정 후 커밋까지 완료된 상태이다.

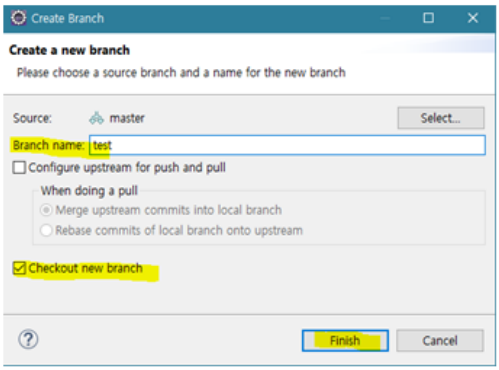


4. 새로운 브랜치 생성과 이동

- 패키지 익스플로러의 프로젝트 루트(Hellow_world [Git master])를 마우스 오른쪽 버튼으로 클릭한 후 [Team] -> [Switch To] -> [New Branch]를 선택한다.

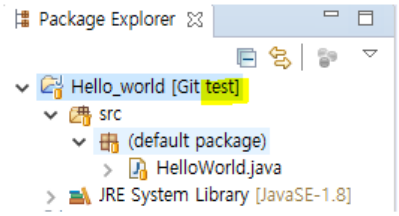


- 'Create a new branch' 창이 열리면 'Branch name'에 알맞은 이름을 넣고 바로 체크아웃을 하기 위해 'Checkout new branch' 항목에 체크 표시를 해준다. 그런 다음 <Finish>을 클릭한다.

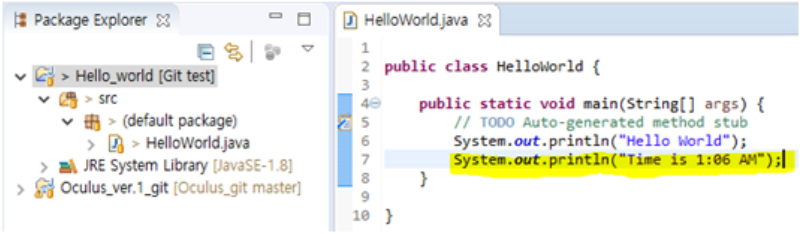


※ 체크아웃 : 선택한 브랜치로 작업 이동

- 패키지 익스플로러의 프로젝트 루트를 살펴보면 프로젝트 이름 옆에 test 브랜치로 체크아웃된 것을 확인할 수 있다.

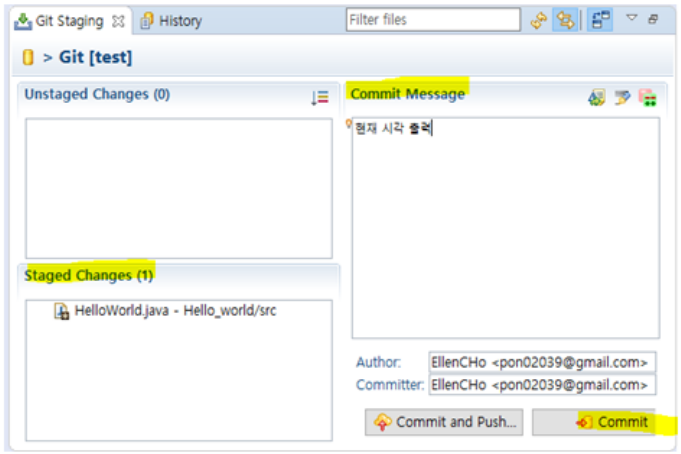


- 프로그램을 수정한다.



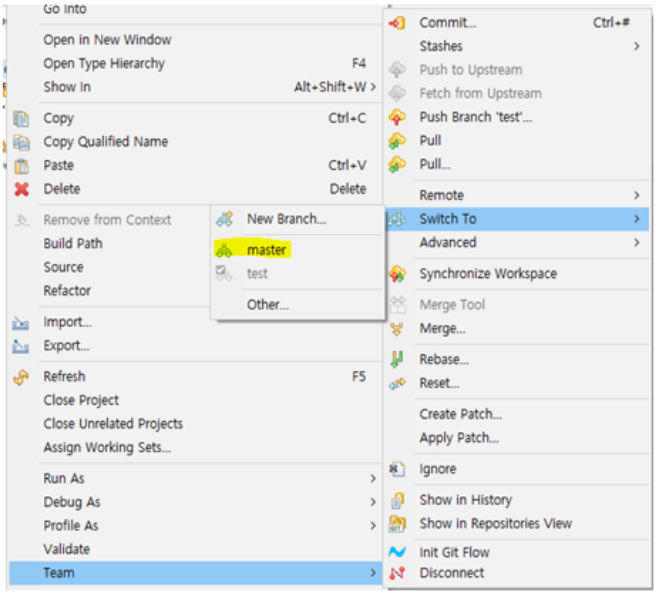
5. 두 번째 커밋

- 파일을 수정했으므로 한번 더 커밋을 해야한다. 따라서 위에서 했던 것처럼 프로젝트 루트에 오른쪽 마우스 버튼을 클릭한 후 [Team] -> [Commit]을 선택한다. 커밋 메시지를 작성한 후 <Commit>을 클릭한다.

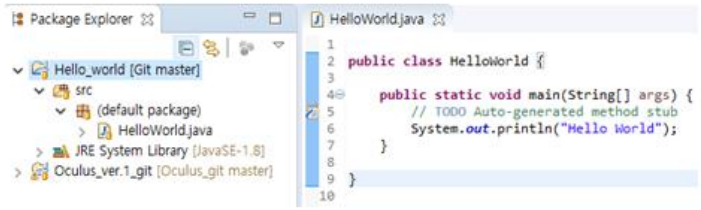


6. master 브랜치와 병합

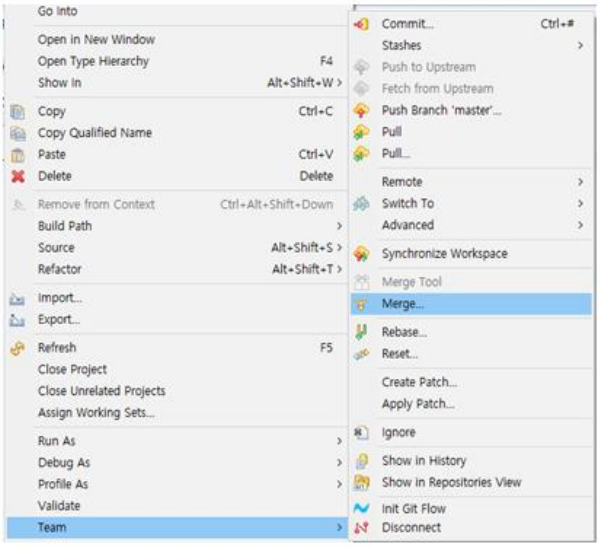
- test 브랜치에서 해야 할 수정 작업이 끝났으니 master 브랜치로 체크아웃해서 test의 수정 내역을 master로 병합해야 한다. 패키지 익스플로러의 프로젝트 루트를 오른쪽 마우스 버튼으로 클릭한 후 [Team] -> [Switch To] -> [master]를 클릭한다.



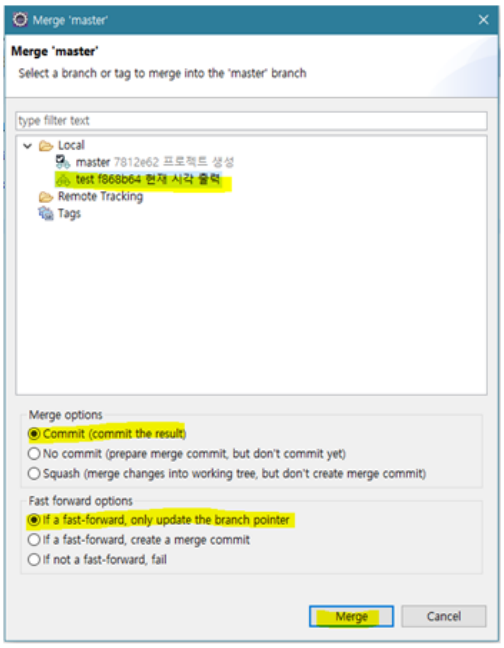
- 파일 내용이 master 브랜치의 마지막 커밋 내용으로 바뀌게 된다.



- master 브랜치에 조금 전 작업한 test의 내용을 병합하기 위해 패키지 익스플로러의 프로젝트 루트를 오른쪽 마우스 버튼으로 클릭한 후 [Team] -> [Merge]를 누른다.



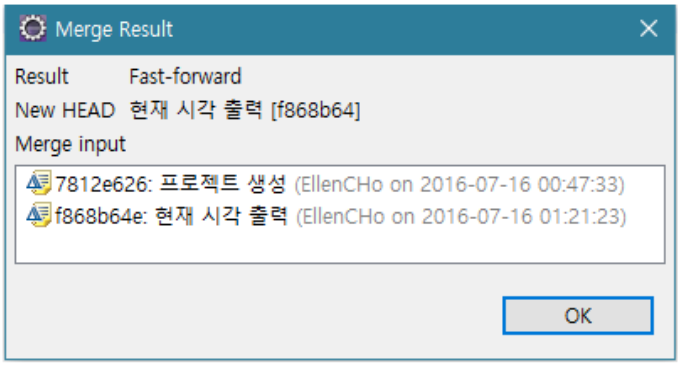
- Merge 'master' 창이 열린다. master 브랜치에 test 브랜치를 병합하려는 것이므로 test 브랜치를 선택하면 <Merge> 버튼이 활성화된다. 아래에 있는 'Merge options'와 'Fast forward options'는 기본상태로 두고 <Merge>를 클릭한다.



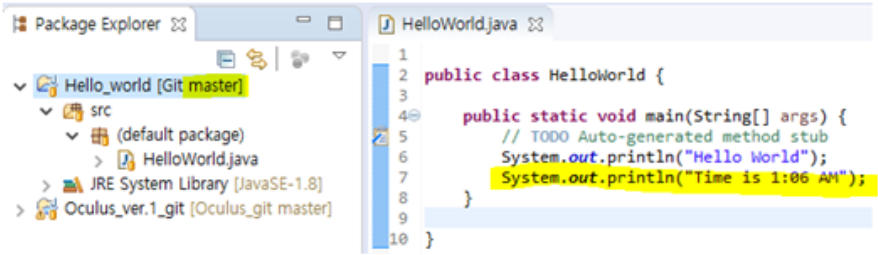
※ Merge options와 Fast forward options의 옵션

- Commit : 일반적인 병합 커밋(병합 커밋이란 브랜치 사이에 변경 사항이 다를 경우 해당 변경 내용을 하나로 통합하는 것)
- No commit : 병합 커밋을 준비하지만 커밋하지는 않는다.
- Squash : 해당 브랜치의 커밋 전체를 통합한 커밋을 추가한다. 단, 병합 커밋을 생성하지는 않는다.
- If a fast-forward, only update the branch pointer : fast-forward일 경우, 브랜치 포인터를 업데이트한다.
  - fast-forward : 변경 내용이 있는 브랜치(ex.test)가 더 최신 내용을 담고 있고 병합하려는 브랜치(ex.master)의 내용을 모두 포함한 경우
  - 브랜치 포인터 : 현재 브랜치 상태를 의미, 예를 들어 master 브랜치가 더 최신 내용을 담은 hotfix 브랜치와 병합했을 경우에는 master 브랜치 상태가 변했으므로 브랜치 포인터가 업데이트됐다고 한다.
- If a fast-forward, create a merge commit : fast-forward일 경우 병합 커밋을 생성한다.
- If not a fast-forward, fail : fast-forward가 아닐 경우 병합하지 않는다.

- 병합 실행 결과를 알려주는 대화 상자가 나타난다. <OK>를 클릭하면 병합이 완료된다.

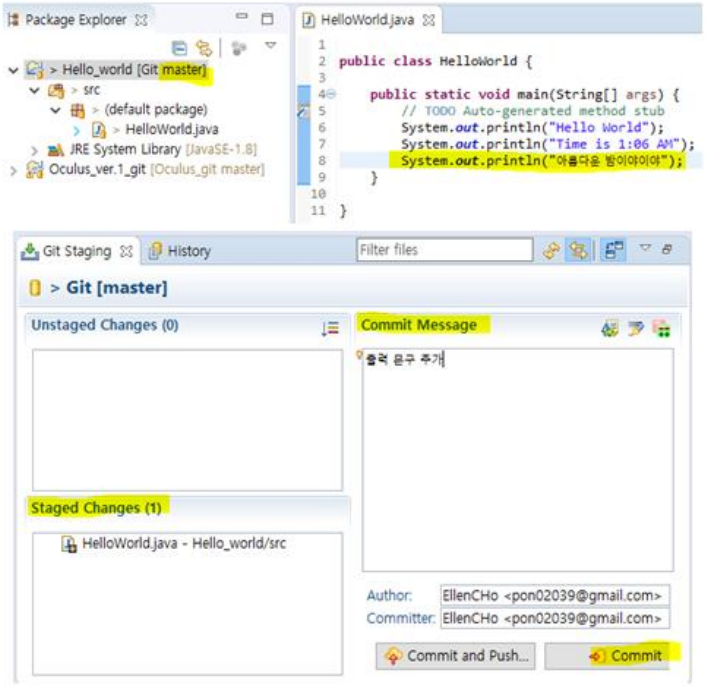


- master 브랜치에 test 브랜치의 내용이 병합된 것을 확인할 수 있다.

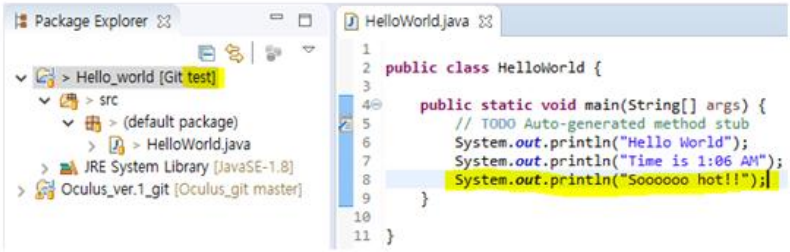


7. 각 브랜치의 독립성 확인

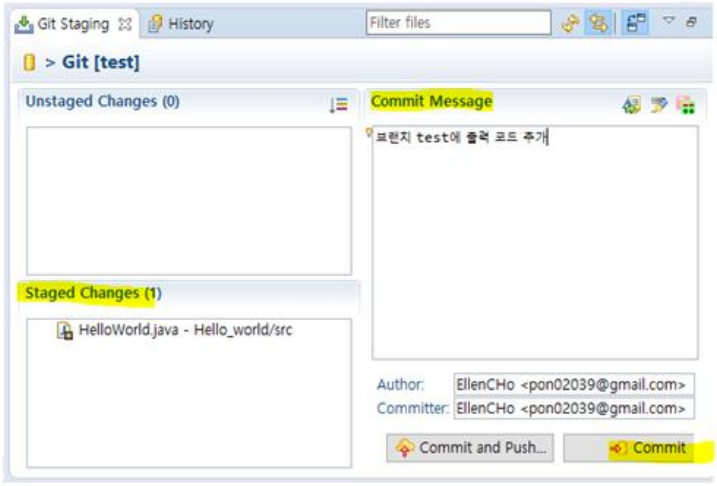
- 먼저 master 브랜치에서 코드를 수정한 후에 변경 사항을 커밋한다.



- 패키지 익스플로러의 프로젝트 루트를 오른쪽 마우스 버튼으로 클릭한 후 [Team] -> [Switch To] -> [test]를 선택한다. 그 후에 다른 코드를 하나 추가한다.



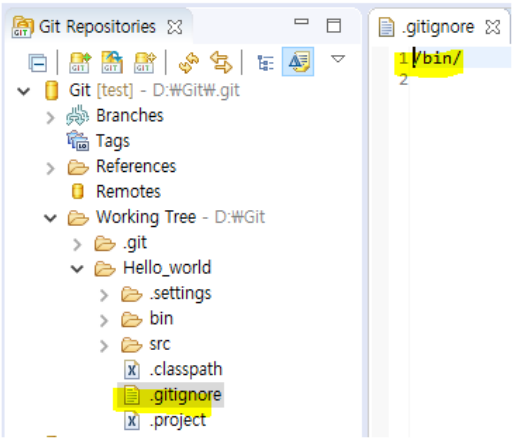
- [Team] -> [Commit}을 선택하여 변경 사항을 커밋한다.



이런 식으로 각 브랜치는 계속 독립성을 갖고 코드를 추가하거나 수정할 수 있다.

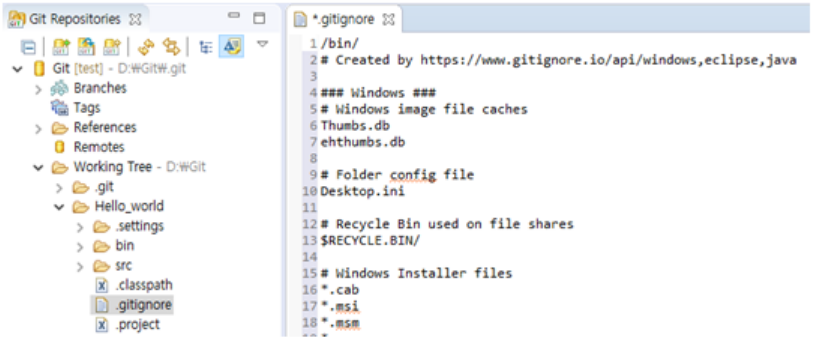
8. 불필요한 파일 및 폴더 무시

- 이클립스의 EGit 플러그인은 기본적으로 저장소를 생성할 때 .gitignore 파일을 함께 생성한다. Git 퍼스펙티브 왼쪽의 'Git Reposotories' 항목에서 'Working Directory'를 열면 화면과 같이 '.git' 디렉터리와 '프로젝트 이름' 디렉터리가 있을 것이다. 그리고 '프로젝트 이름' 디렉터리 아래 .gitignore 파일이 있다. 이 파일을 더블 클릭해서 연 후 파일 내용을 살펴보면 이미 '/bin/' 디렉터리가 포함되어 있다. 이는 이클립스에서 컴파일하면 생성되는 바이너리 파일들을 저장소에 포함하지 않겠다는 의미이다.

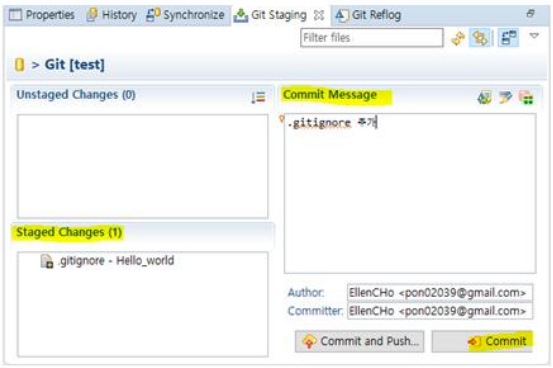


- .gitignore 파일을 좀더 보완하기 위해 https://www.gitignore.io/ 에 접속해서 사용 중인 환경인 'Windows', 'Eclipse', 'Java'를 넣고 <Generate>fmf 클릭하여 .gitignore 파일 내용을 생성한다.

- 생성된 내용 전부를 선택한 후 복사해서 이클립스에 열어 둔 .gitignore 편집기에 붙여넣는다. ('/bin/' 아래 2행 이후로 붙여넣기를 한다.)

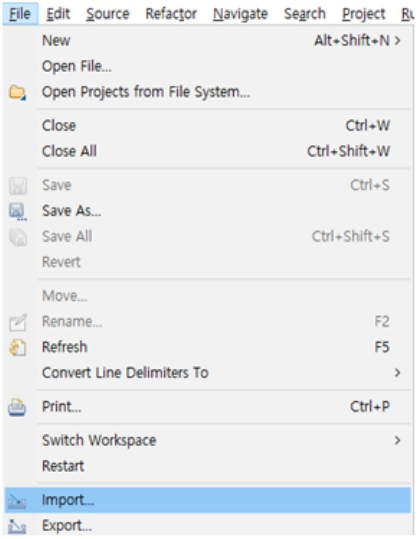


- 저장을 한 후 프로젝트 루트를 오른쪽 마우스로 클릭하고 [Commit]을 선택하여 변경 사항을 커밋한다.

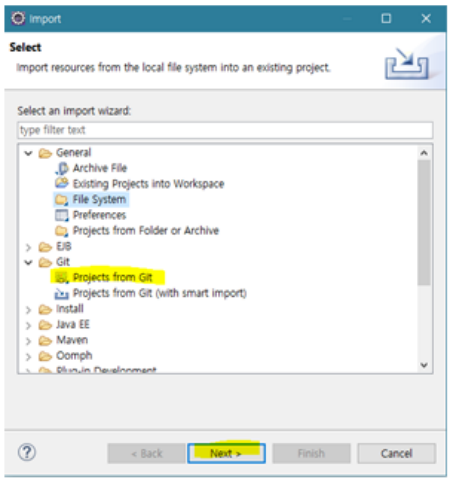


9. 원격 저장소의 내용을 로컬 저장소로 가져오기

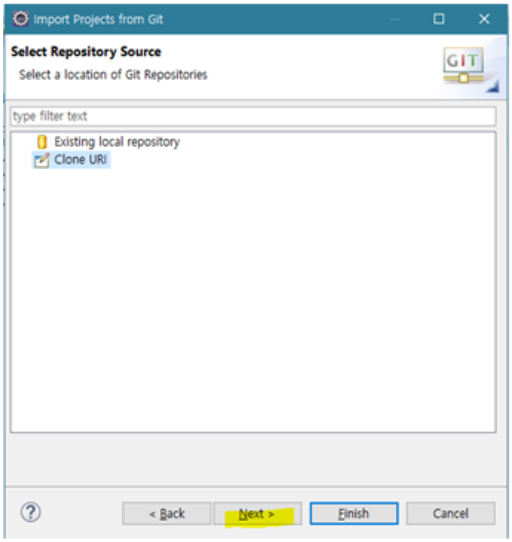
- 이클립스 EGit 플러그인을 이용해서 원격 저장소를 클론하는 방법은 [File] -> [Import]를 선택해서 클론하거나 Git 퍼스펙티브를 이용할 수 있다. 그 중 [Import] 메뉴를 이용해 클론하는 방법을 알아보겠다.



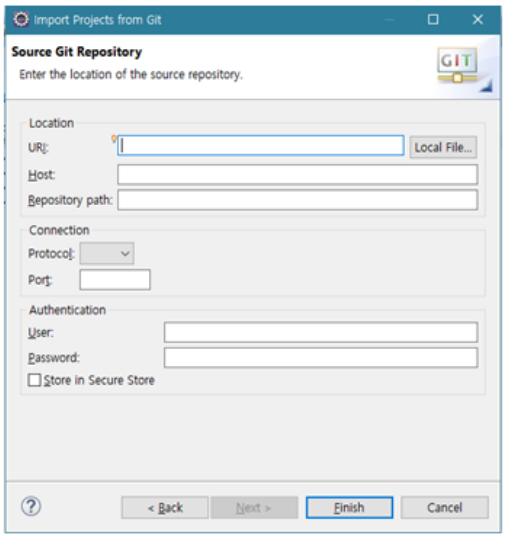
- [Import]를 선택하면 'Select' 창에서 'Git'항목이 보인다. 항목을 펼쳐서 'Projects from Git'을 선택하고 <Next>를 클릭한다.



- 로컬에 있던 Git 프로젝트를 추가할 때는 'Existing local repository'항목을 선택하면 되지만 GitHub에 있는 프로젝트를 클론하는 것이므로 'Clone URI'를 선택하고 <Next>를 클릭한다.

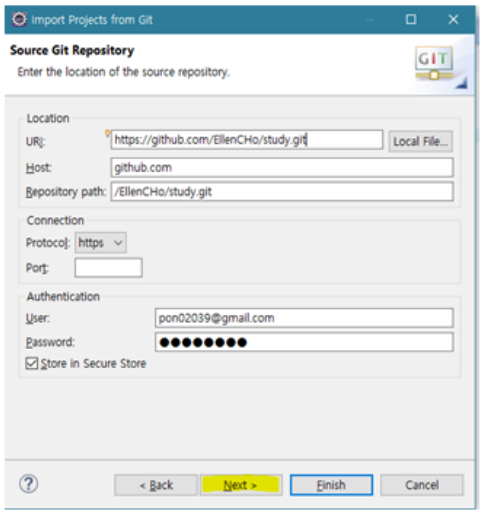


- 가져올 원격 저장소의 정보를 입력해야 하는 'Source Git Repository'창이 열린다.

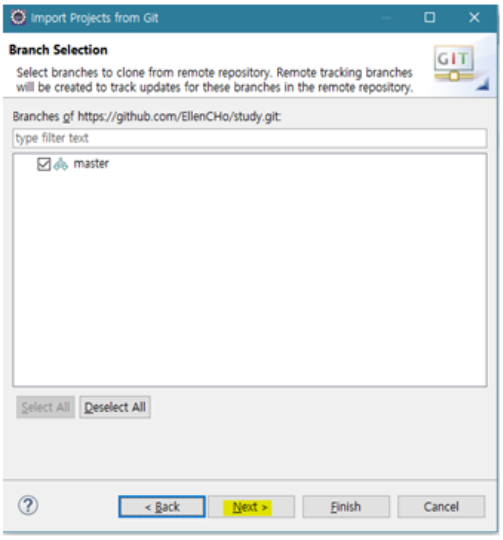


- URI : GitHub 또는 Git 서버에서 알려주는 'HTTPS clone URL' 주소를 입력한다. 상황에 따라 SSH 프로토콜이나 서브버전의 주소를 입력해도 된다.
- HOST : Git 서버의 호스트 주소를 표시한다. GitHub의 경우 URI 항목에 주소를 입력하면 자동으로 값이 입력된다.
- Repository path : 원격 저장소의 경로를 입력한다. GitHub의 경우 URI 항목에 주소를 입력하면 자동으로 값이 입력된다. '/사용자이름/저장소이름.git'형식이다.
- Protocol : 원격 저장소를 가져오는 데 필요한 네트워크 프로토콜을 선택한다. file, ftp, git, http, https, sftp, ssh 중 하나를 선택할 수 있다.
- Port : 특정 포트에 접속해서만 원격 저장소를 가져올 수 있는 경우 해당 포트 번호를 입력한다. 아닌 경우에는 그냥 비워둔다.
- User/Pasword : 원격 저장소가 있는 Git 서버에 로그인할 필요가 있는 경우 사용자 이름과 비밀번호를 입력한다.
- Store in Secure Store : 로컬 보안 저장소에 원격 저장소를 저장해야 할 경우 체크한다.

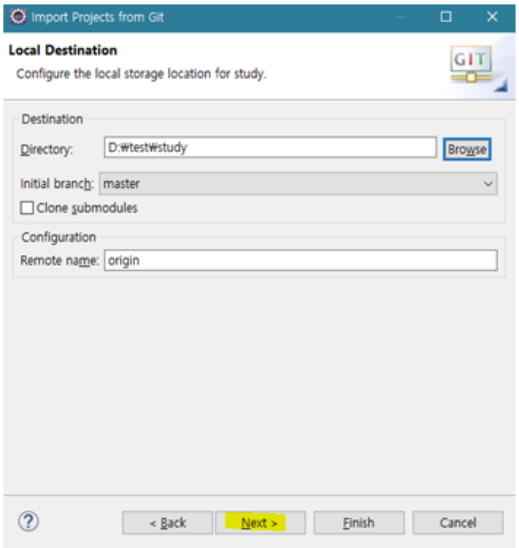
- 'URI'항목에 'HTTPS clone URL' 주소를 붙여넣기하면 항목이 자동으로 채워진다. 채워진 항목 이외의 다른 항목은 입력하지 않고 <Next>를 클릭한다.



- 브랜치를 선택할 수 있는 'Branch Selection' 창이 열린다. master 이외의 다른 브랜치 정보도 함께 로컬 저장소에 가져오고 싶다면 다른 브랜치를 선택하면 된다. <Next>를 클릭한다.



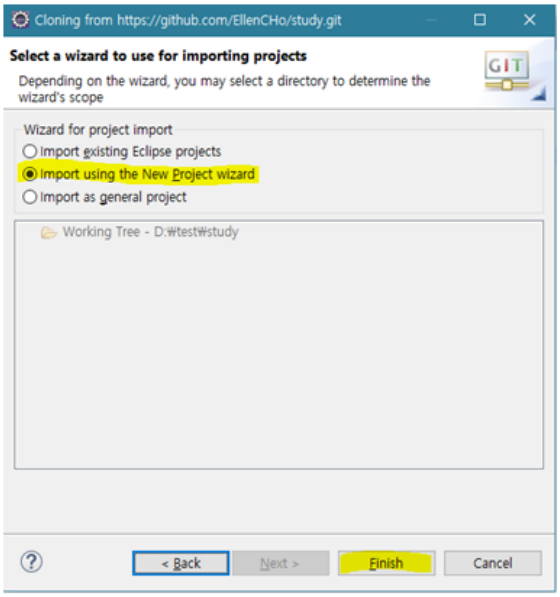
- 'Local Destination' 창은 프로젝트를 어디에 저장할 지 선택하는 화면이다. 자신이 평소에 프로젝트를 넣어두는 디렉터리를 지정하면 된다. <Next>를 클릭한다.



  - Directory : 원격 저장소의 디렉터리 경로를 설정한다.
  - Initial branch : 초기 브랜치를 선택한다. 원격 저장소에 여러 개 브랜치가 있는 경우에는 원하는 브랜치를 선택할 수 있다.
  - Clone submodules : 현재 원격 저장소를 메인 프로젝트의 하위 프로젝트인 서브모듈로 클론하려면 체크 표시를 해준다. 해당 원격 저장소가 프로젝트에 사용되는 라이브러리리 경우 유용하다.
  - Remote name : 원격 저장소의 이름이다. 기본값은 'origin'이다. 로컬 저장소의 브랜치 개념으로 생각하면 이해하기 쉽다.

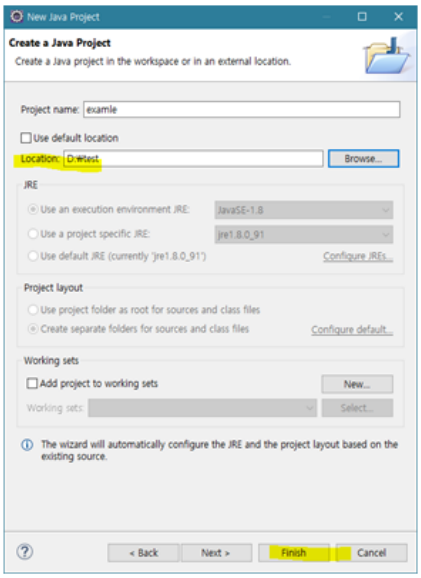
- 여기까지는 단순히 GitHub의 프로젝트를 로컬에 저장한 것에 불과하다. 실제로 이클립스에서 사용하려면 프로젝트를 만들고 이클립스 프로젝트에 로컬 Git 저장소 파일들을 추가해야 한다.

- 'Select a wizard to use for importing projects'창에서 어떤 방식으로 이클립스에 프로젝트를 추가할 지 선택한다. 여기에서는 새로운 프로젝트를 선택하므로 두번째를 선택하고 <Finish>를 클릭한다.

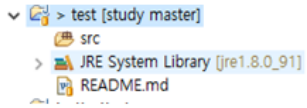


  - Import existing Eclipse projects : 현재 존재하는 이클립스 프로젝트에 원격 저장소를 불러온다.
  - Import using the new Project wizard : 새로운 프로젝트 마법사를 이용해 프로젝트를 만들고 원격 저장소를 불러온다.
  - Import as general project : 이클립스에 생성된 프로제그가 아닌 프로젝트를 불러올 때 선택한다.

- 새 프로젝트 생성을 위한 'Select a wizard' 창이 열린다. 이때 설정 방법의 핵심은 로컬에 저장한 Git 프로젝트 디렉터리를 지정하는 것이다. 즉 Git 저장소 디렉터리와 프로젝트 디렉터리를 동일하게 하는 것이다.

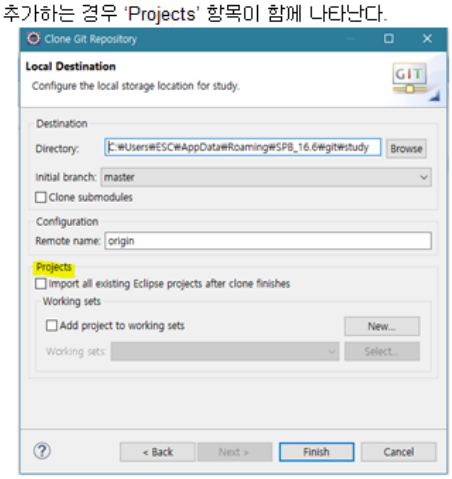


- 프로젝트가 생성되면 Git을 이용한 프로젝트라고 표시하기 위해 조그마한 원통 모양의 아이콘이 붙는다.



※ 경로를 잘못 설정했기 때문에 위의 사진과 프로젝트 이름이 다르다.

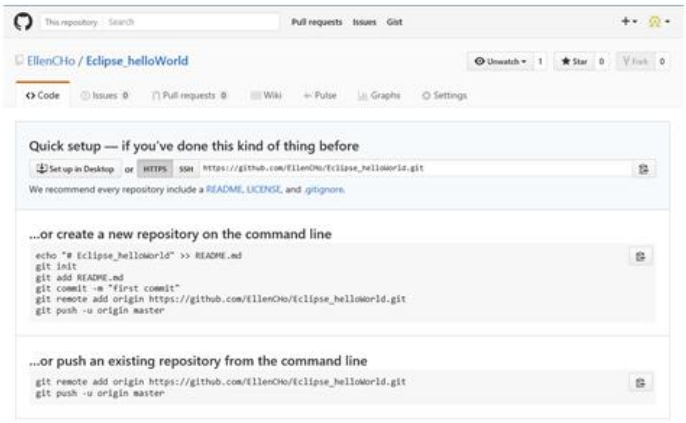
※ Git 퍼스펙티브 창에서 원격 저장소를 추가하는 경우 'Projects' 항목이 함께 나타난다.



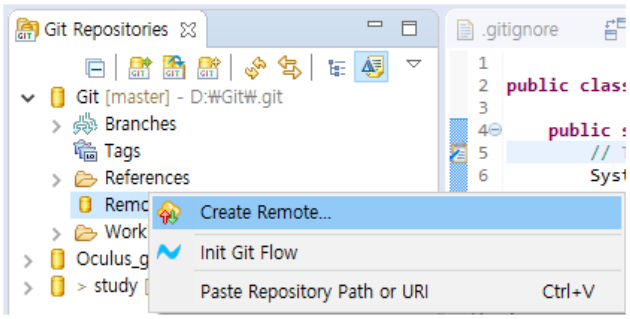
- Import all existing Eclipse projects after clone finishes : 체크 표시를 켜면 원격 저장소에 있는 프로젝트가 이클립스 프로젝트인 경우 클론 후 자동을 프로젝트를 불러온다.
- Add project to working sets : 클론하는 원격 저장소 프로젝트에 Working set를 추가할 경우에 체크를 킨다. 원하는 working set을 지정할 수 있다.
- Git 퍼스펙티브에서 원격 저장소를 추가했으 때 이클립스에서 생성한 프로젝트가 아닌 경우에는 패키지 익스플로러에 프로젝트가 추가되지 않을 수 있다. 그런 경우에는 원격 저장소 하위 항목인 'Working Directory'항목에 오른쪽 마우스 버튼을 클릭한 후 [Import Projects]를 선택한다. 그 후 'Import as general project'를 선택해 패키지 익스플로러에 프로젝트를 불러올 수 있다.

12. 로컬 저장소와 원격 저장소를 연결하기

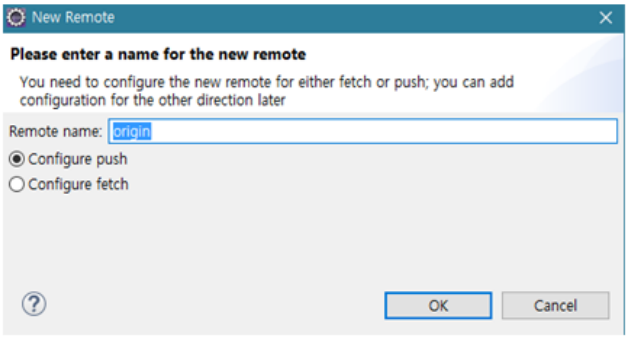
- GitHub에서 빈 원격 저장소를 생성한다.



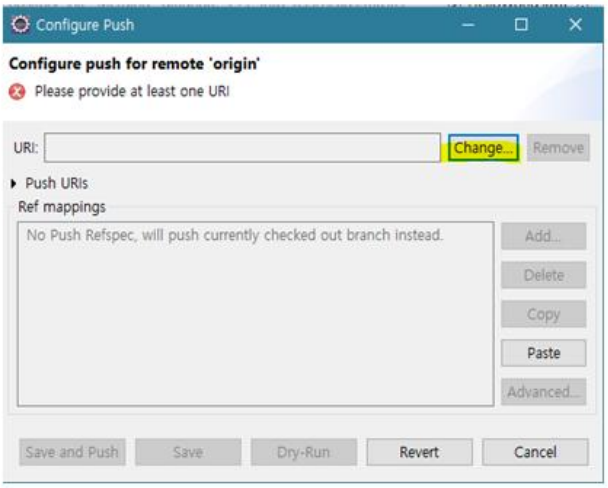
- Git 퍼스펙티브를 선택해서 Hello_world 프로젝트의 하위 항목을 확장한 후 'Remotes' 항목을 오른쪽 마우스 버튼으로 클릭하고 [Create Remote]를 선택한다.



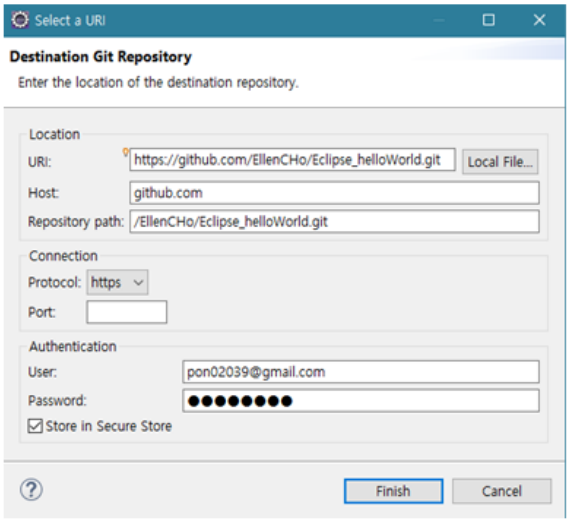
- 창이 열리면 'Remote name ' 항목에 'origin'이 입력되어 있는지와 'Configure push' 항목이 선택되어 있는지를 확인한 후 <OK>를 누른다.



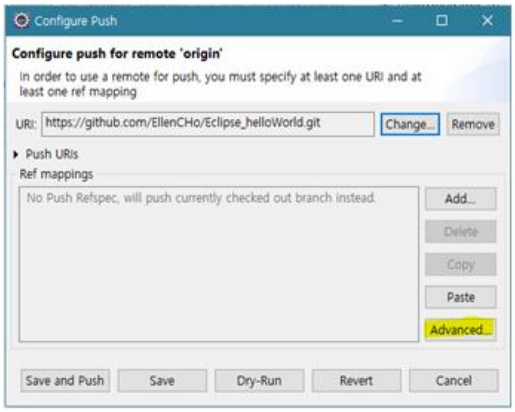
- 다음 창이 열리면 ' URI' 항목 옆에 있는 <Change>를 클릭한다.



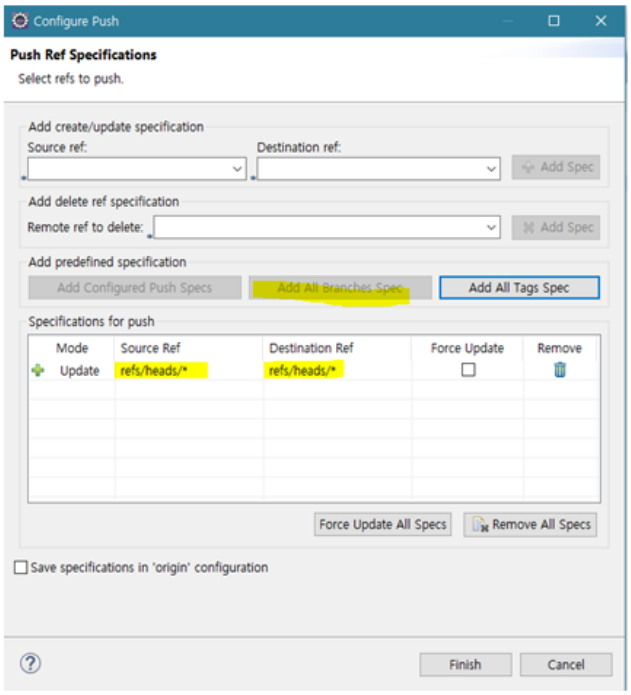
- ‘Destination Git Repository’ 창이 열리고 ‘URL’ 항목에 새로 생성한 원격 저장소의 ‘HTTPS clone URL’ 항목의 주소를 입력하면 자동으로 ‘Host’ 항목과 ‘Repository path’항목이 채워진다. ‘Connection’ 항목의 설정은 그대로 둔다. SSH 등 다른 프로토콜을 선택하는 경우는 ‘Protocol’ 항목의 설정값을 사용하는 프로토콜로 변경하면 된다. ‘User’ 항목과 ‘PassWord’항목은 안채우고 넘어가도 진행 과정 중에 다시 물어본다. 설정이 끝나면 <Finish>를 클릭한다.



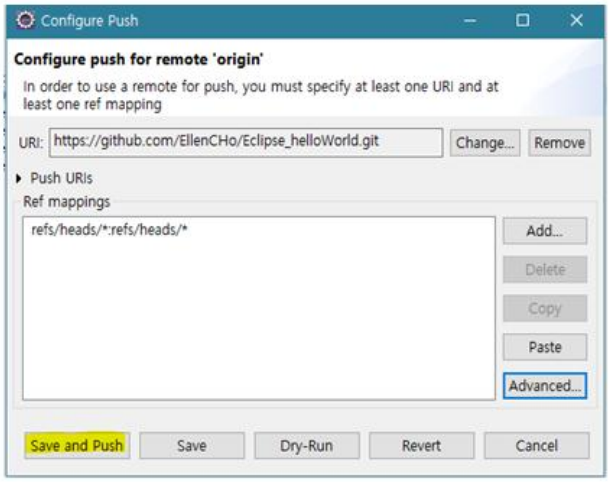
- 다시 창으로 돌아와서 오른쪽 아래에 있는 <Advanced>를 클릭한다.



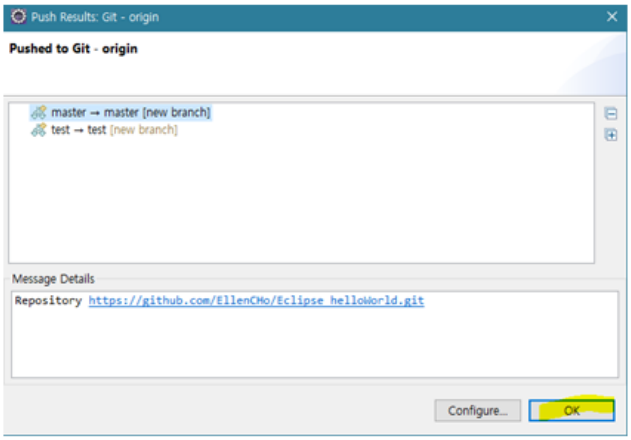
- 'Push Ref Specifications' 창이 나타나는 데 이 부분이 설정 과정의 핵심이다. 지금까지 진행했던 모든 작업을 지정하기 위해 <Add All Branches Spec>을 클릭한다. 'Specifications for push' 항목의 'Source Ref' 항목과 'Destination Ref' 항목에 'refs/heads/*'라고 설정되어 있따면 정상적으로 설정된 것이다. 확인했다면 <Finish>를 클릭한다.



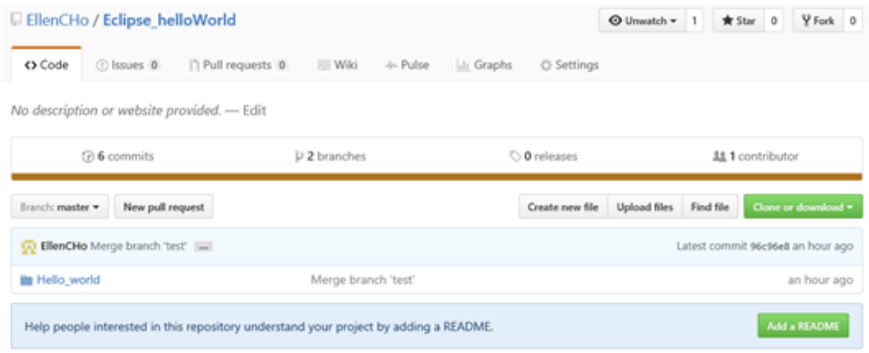
- 아래의 사진과 같다면 설정을 마친 것이다. 원격 저장소 정보 저장 후 바로 푸시하기 위해 <Save and Push>를 클릭한다.



- <OK>를 클릭하면 작업이 완료된 것이다. 혹시 푸시하지 못한 작업 내역이 있다면 <configure>를 클릭해 작업 내역을 추가하면 된다.

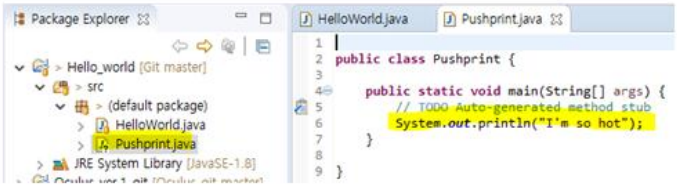


- 원격 저장소를 확인하면 로컬 저장소의 데이터가 푸시되었음을 확인할 수 있다.

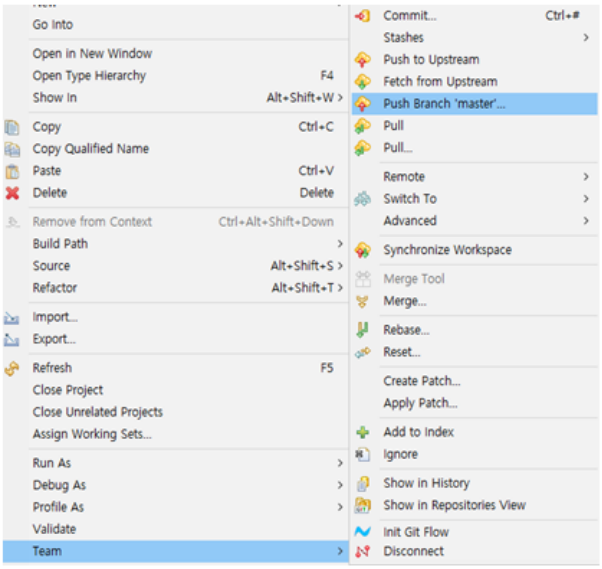


13. 로컬 작업 내역을 원격 저장소에 올리기

- 새 클래스 파일을 생성하고 변경 사항을 커밋한다.

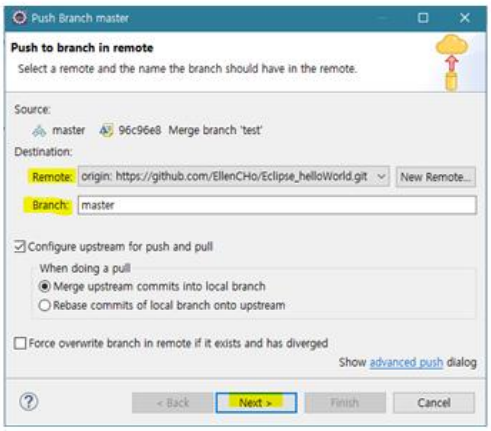


- master 브랜치의 변경 내역을 원격 저장소에 푸시하는 두가지 방법이 있는데 Push to Upstream을 이용하는 방법과 Push branch 'Master'을 이용하는 방법이다. Push to Upstream은 모든 브랜치의 작업 내역을 푸시하는 간편한 방식이지만 작업 내역을 세분화해서 푸시하는 것은 어렵다. 따라서 Push branch 'Master'를 선택해 푸시해보겠다. [Team] -> [Push branch 'Master']을 클릭한다.



※ 커밋 과정에서 <Commit and Push>를 클릭하는 것은 <Push to Upstream>을 실행하는 것이다.

- 창이 뜨면 'Source' 항목의 변경 사항, 'Remote' 항목의 원격 저장소 주소, 'Branch' 항목의 브랜치 등을 확인 한다. 'Branch' 항목의 경우 기본값은 master 브랜치로 설정되어 있는데 만약 다른 브랜치로 바꾸고 싶다면 'refs/heads/브랜치이름' 형태로 입력해 선택할 수 있다. 설정을 확인했다면 <Next>를 누른다.

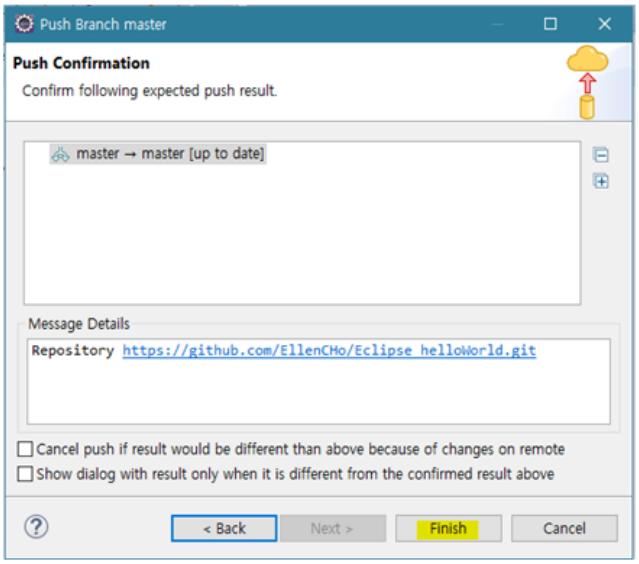


※ 'Configure upstream for push and pull'은 푸시에 필요한 옵션을 선택하는 것이다. 자세한 내용은 뒤에 나온다

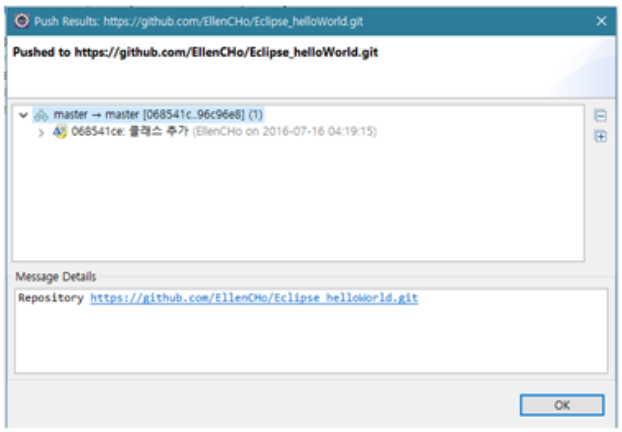
※ 'Force overwrite of branch on remote if it exists and has diverged'는 원격 저장소에는 없는데 로컬 저장소에 존재하거나 변경된 파일이 있는 경우 로컬 저장소 기준으로 원격 저장소에 덮어 쓰기를 하겠다는 옵션이다. 선택하지 않아야 한다.

- ‘Push Confirmation’ 창에서는 푸시할 내역을 최종적으로 확인할 수 있다. 아래에는 두 가지 선택 항목이 있다.
  - Cancel push if result would be different than above because of changes on remote : 원격 저장소에 지금 푸시하는 로컬 저장소와 다른 내역이 있다면 연결과 푸시를 취소한다.
  - Show dialog with result only when it is different form the confrimed result above : 원격 저장소에 지금 푸시하는 로컬 저장소와 다른 내역이 있다면 푸시 결과에 대한 별도의 메시지를 보여준다.

  굳이 선택 항목을 체크할 필요가 없으므로 기본 설정 그대로 두고 <Finish>를 누른다.



- 푸시가 큰 문제 없이 완료되면 결과 내역을 보여준다. <OK>를 클릭하면 완료된 것이다.



- GitHub 저장소 확인