

# Git 이용한 버전 관리

✓ 원리를 알면 IT가 맛있다

**Java Programming for Beginners**



chapter 01.

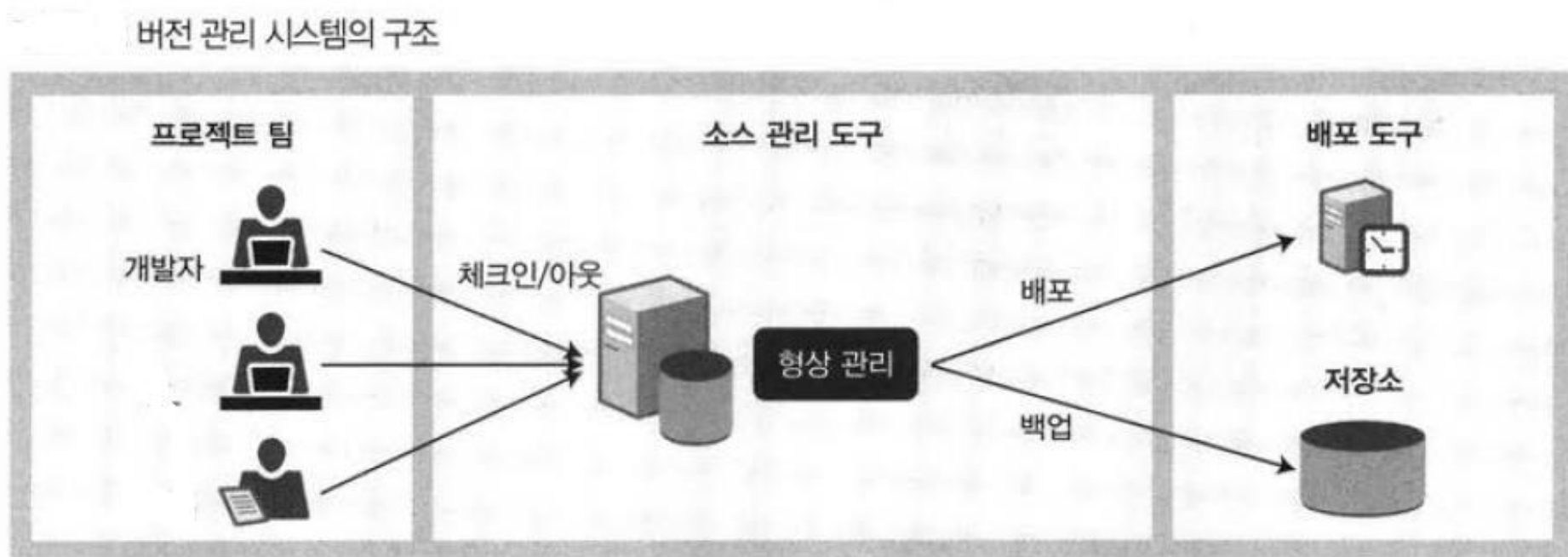
---

# Git 개요

### ○ 버전 관리 ( version control )

버전 관리 (version control, revision control), 소스 관리 (source control), 소스 코드 관리 (source code management, SCM)란 동일한 정보에 대한 여러 버전을 관리하는 것을 말한다. 공학과 소프트웨어 개발에서 팀 단위로 개발 중인 소스 코드나 청사진 같은 설계도 등의 디지털 문서를 관리하는 데 사용된다.

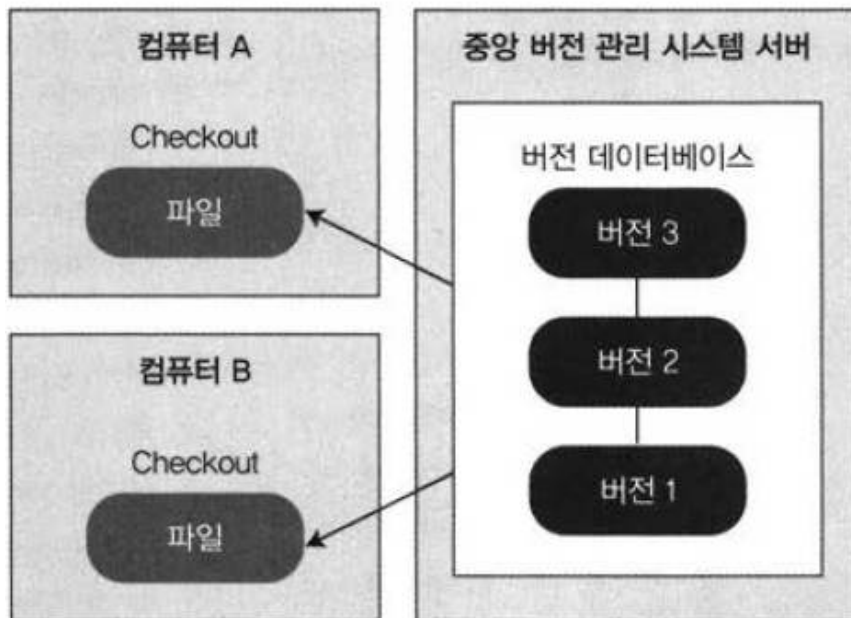
### ○ 버전 관리 시스템



### ○ 버전 관리 시스템 종류

#### 가. 클라이언트-서버 모델

하나의 중앙 저장소를 공유한 후 각각의 클라이언트(개발자)는 저장소의 일부분만 사용하는 형태이다. 자신이 작업하는 부분만 로컬에 임시로 저장한 후 작업함. 이 모델은 중앙 저장소에서 프로젝트 관리의 모든 것을 처리한다. 만약 서버가 고장나면 불완전한 로컬 파일만 남게 된다.

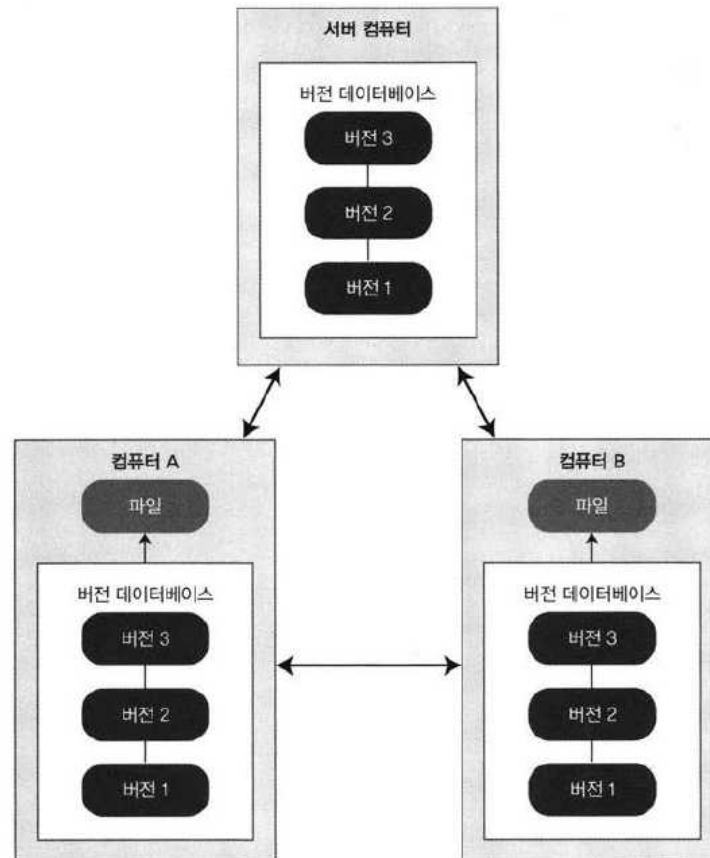


- 대표적인 관리 시스템

CVS, SubVersion 등

### 나. 분산 모델

분산 모델은 프로젝트에 참여하는 모든 클라이언트(개발자)가 전체 저장소에 대한 개별적인 로컬 저장소를 갖고 작업하는 형태이다. '클라이언트-서버' 모델과 달리 클라이언트(개발자) 각자가 온전한 전체 저장소의 사본을 로컬에 가지게 된다.



- 대표적인 관리 시스템

Git

### ○ Git

- 분산 버전 관리 시스템중 하나.
- 2005년 리누스 토르발스에 의해서 만들어짐.



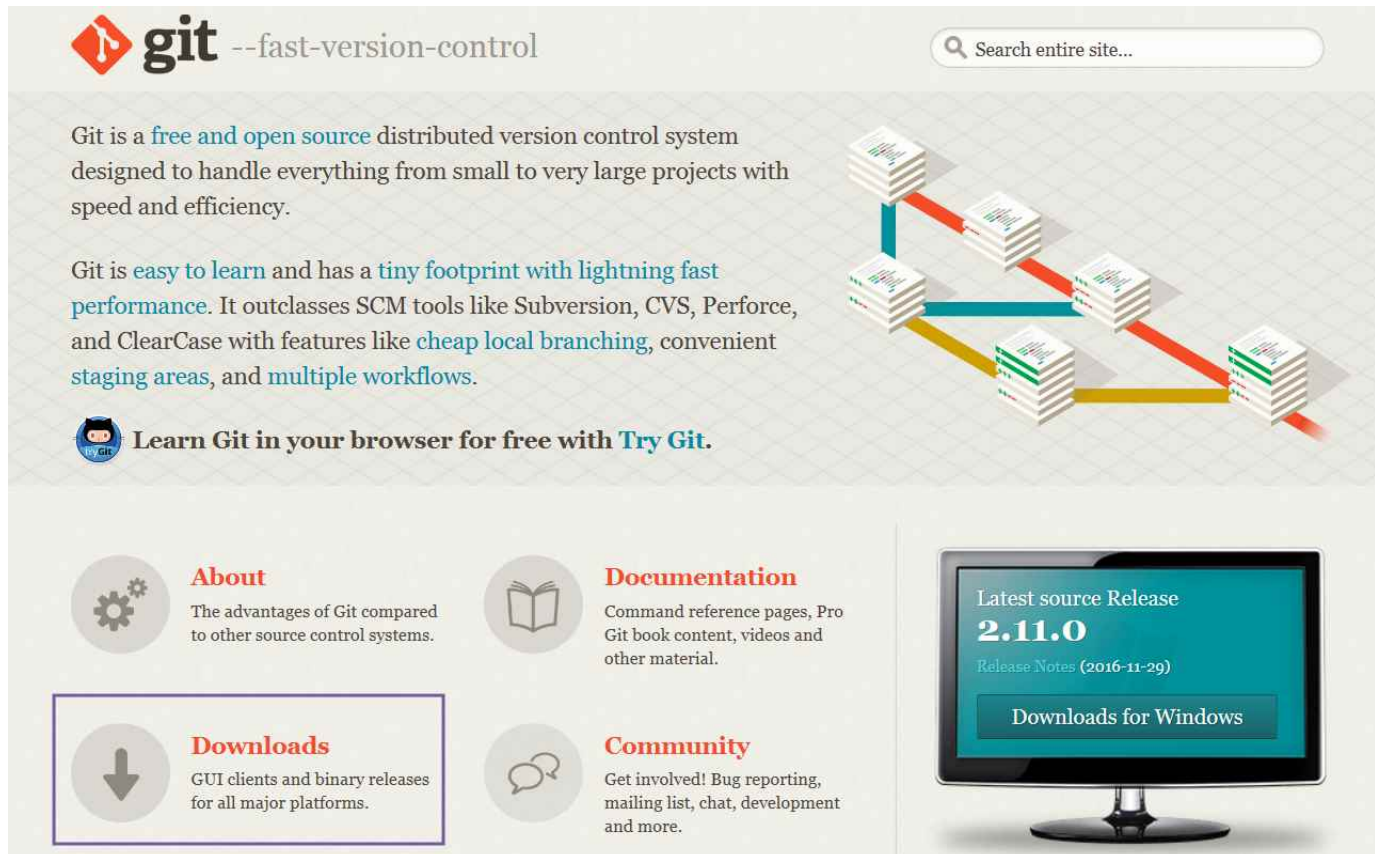
### ○ Git 특징

- 로컬 및 원격 저장소 생성
- 로컬 저장소에 파일 생성 및 추가
- 수정 내역을 로컬 저장소에 제출
- 파일 수정 내역 추적
- 원격 저장소에 제출된 수정 내역을 로컬 저장소에 적용
- Master에 영향을 끼치지 않는 브랜치(branch) 생성
- 브랜치 병합( merge )

### ○ Git 설치

#### 가. Git 다운로드

<https://git-scm.com> 접속 후에 Downloads 링크 선택하기



The screenshot shows the Git website homepage. At the top left is the Git logo with the tagline "--fast-version-control". To the right is a search bar labeled "Search entire site...". Below the logo, there is a paragraph describing Git as a "free and open source" distributed version control system. Another paragraph mentions it is "easy to learn" and has a "tiny footprint with lightning fast performance". Below this is a link to "Learn Git in your browser for free with Try Git." with the GitHub logo. On the right side, there is a diagram illustrating a branching model with stacks of code and colored arrows. At the bottom, there are four main navigation links: "About" (describing advantages), "Documentation" (command reference, Pro Git book, etc.), "Downloads" (highlighted with a purple border, showing GUI clients and binary releases), and "Community" (bug reporting, mailing list, etc.). On the far right, a monitor displays the "Latest source Release 2.11.0" and a button for "Downloads for Windows".

## □ 1) Git 개요

각 운영체제에 맞는 Git 다운로드 클릭하기

**Downloads**

Mac OS X Windows Linux Solaris

Older releases are available and the Git source repository is on GitHub.

**GUI Clients**

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tool users looking for a platform-specific experience.

[View GUI Clients →](#)

**Logos**

## Downloading Git

자동으로 다운로드 됨.



### Your download is starting...

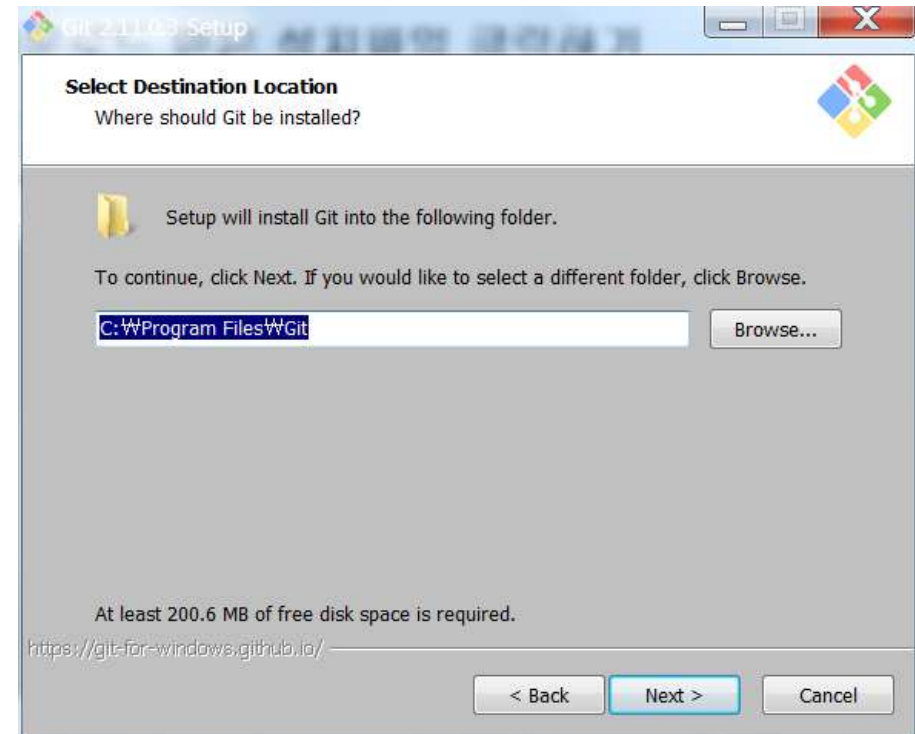
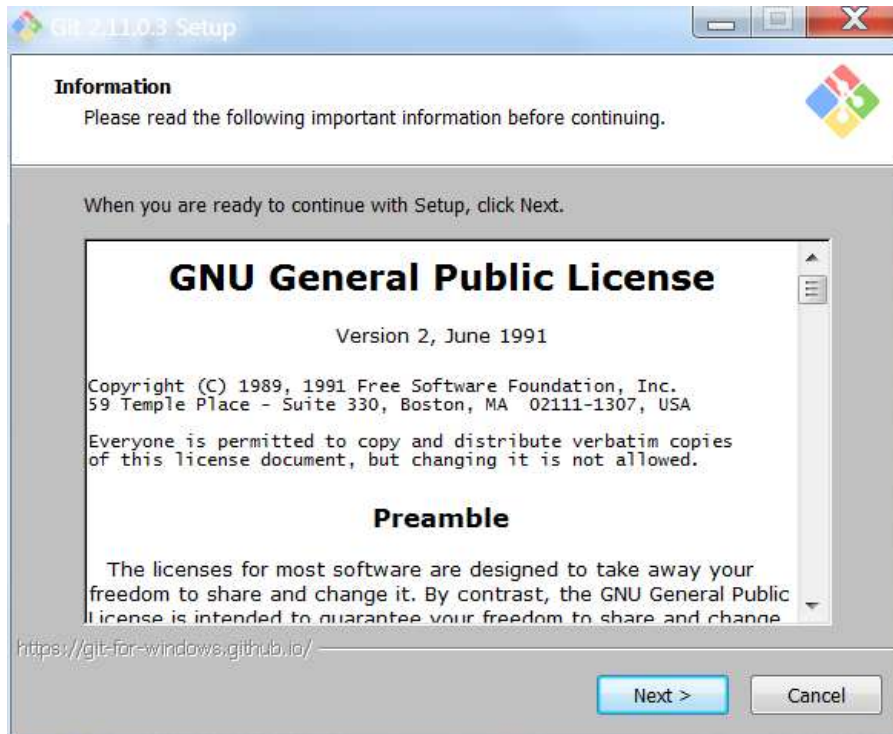
You are downloading the latest (**2.11.0**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **11 days ago**, on 2017-01-14.

If your download hasn't started, [click here to download manually](#).

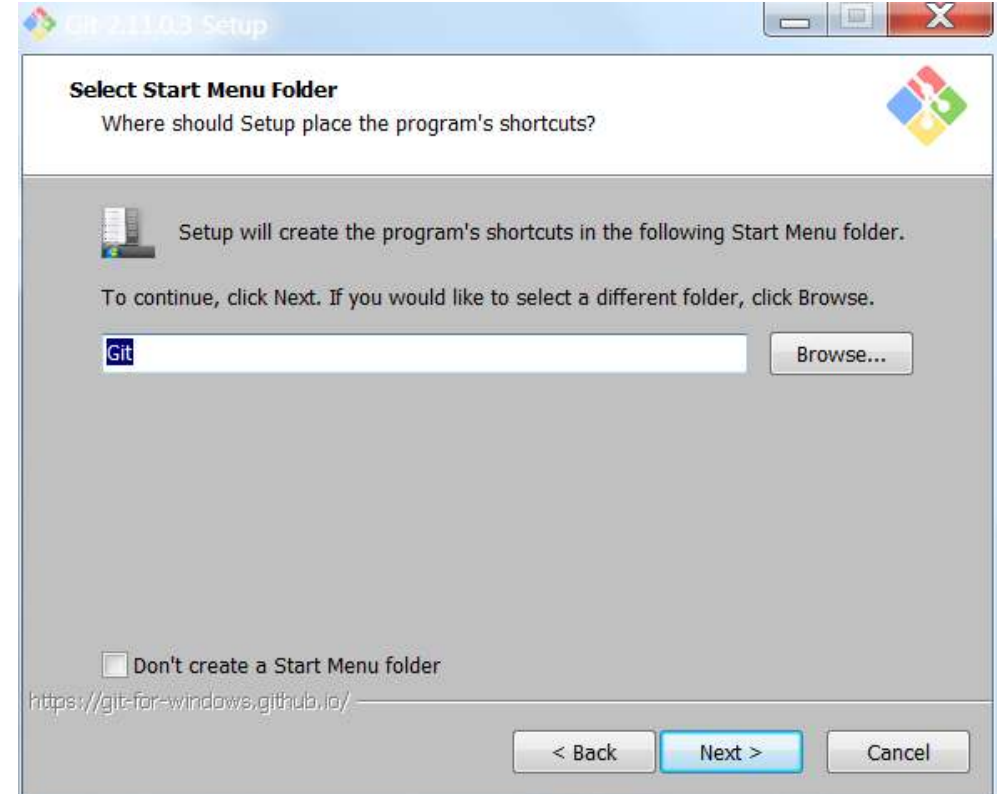
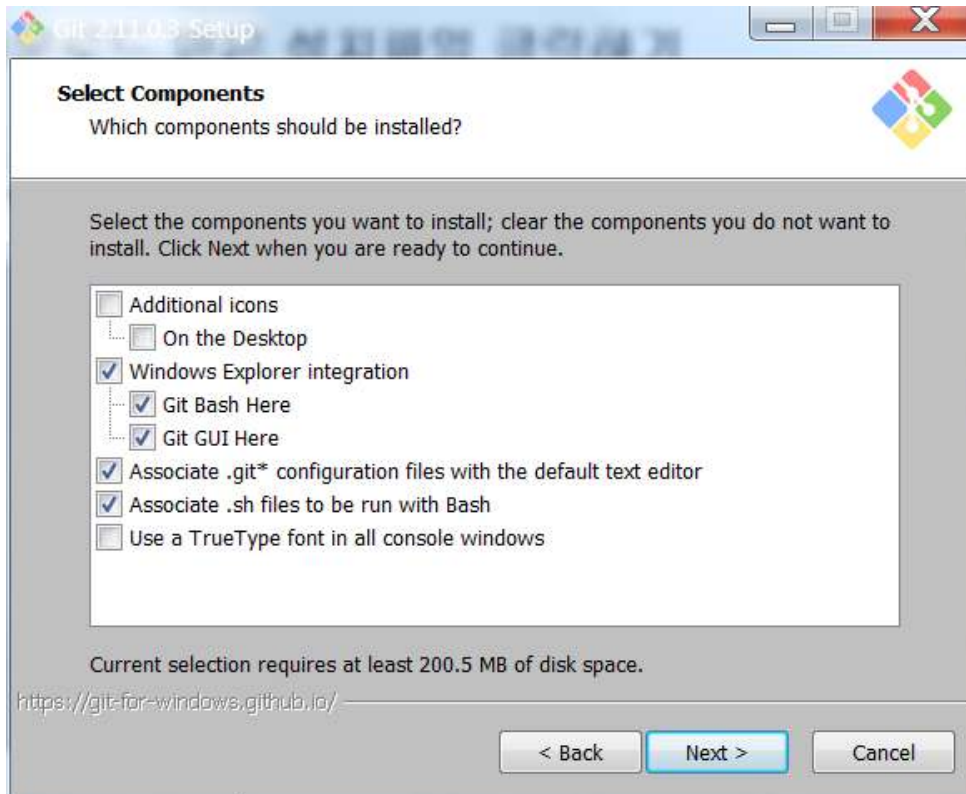


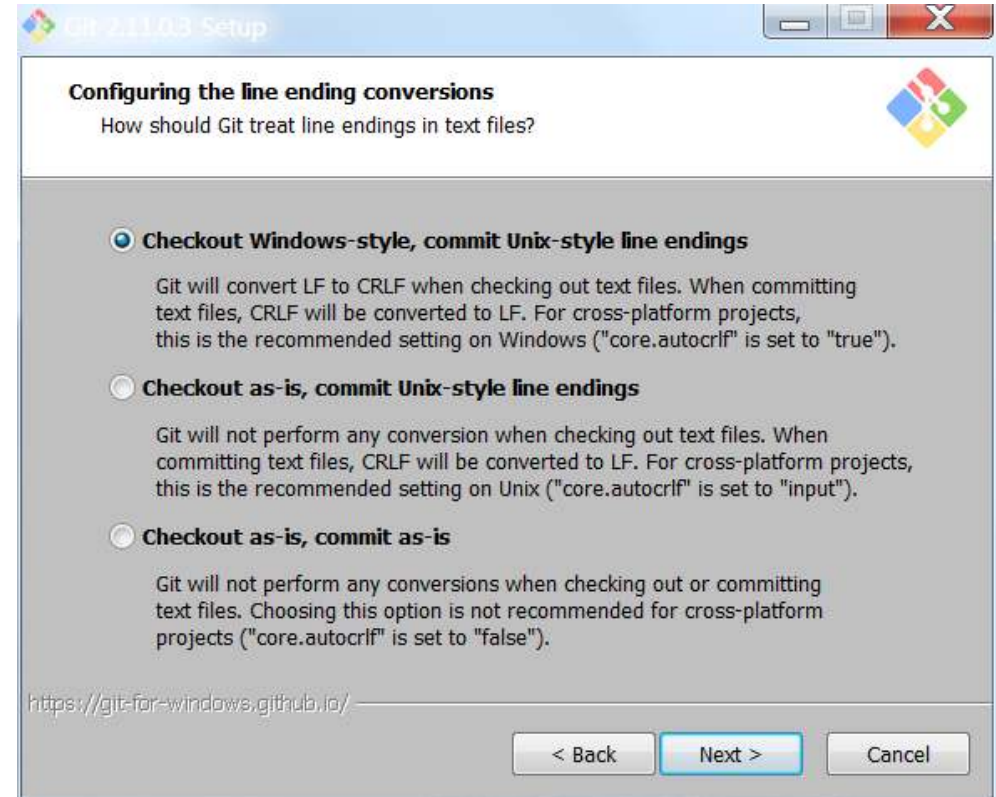
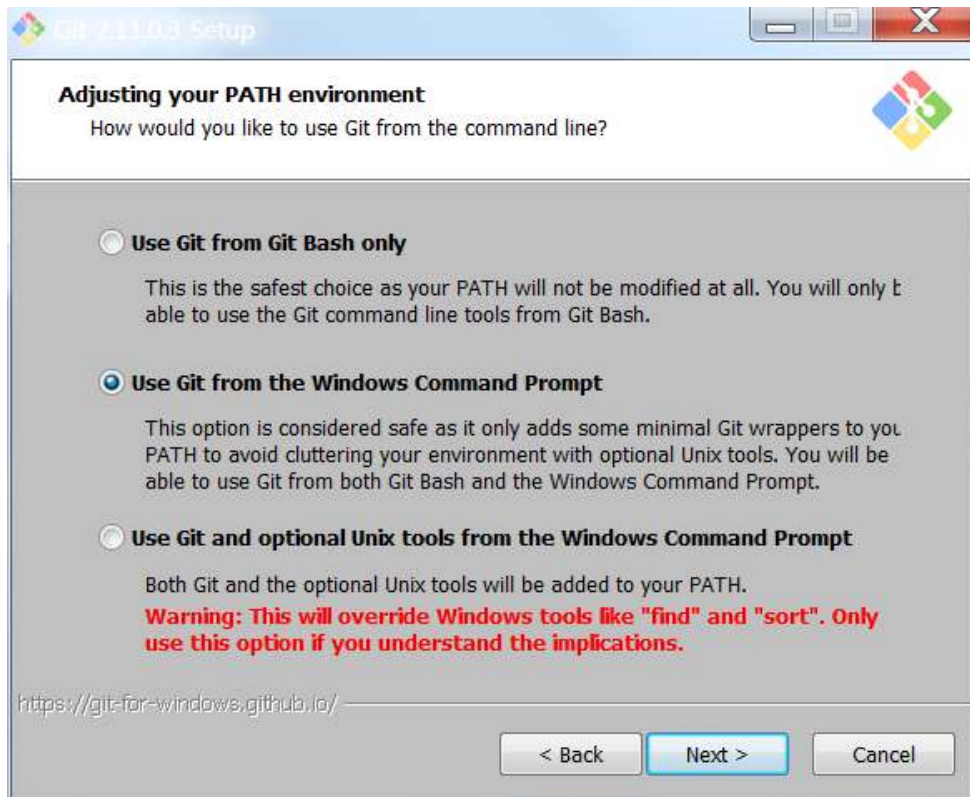
## □ 1) Git 개요

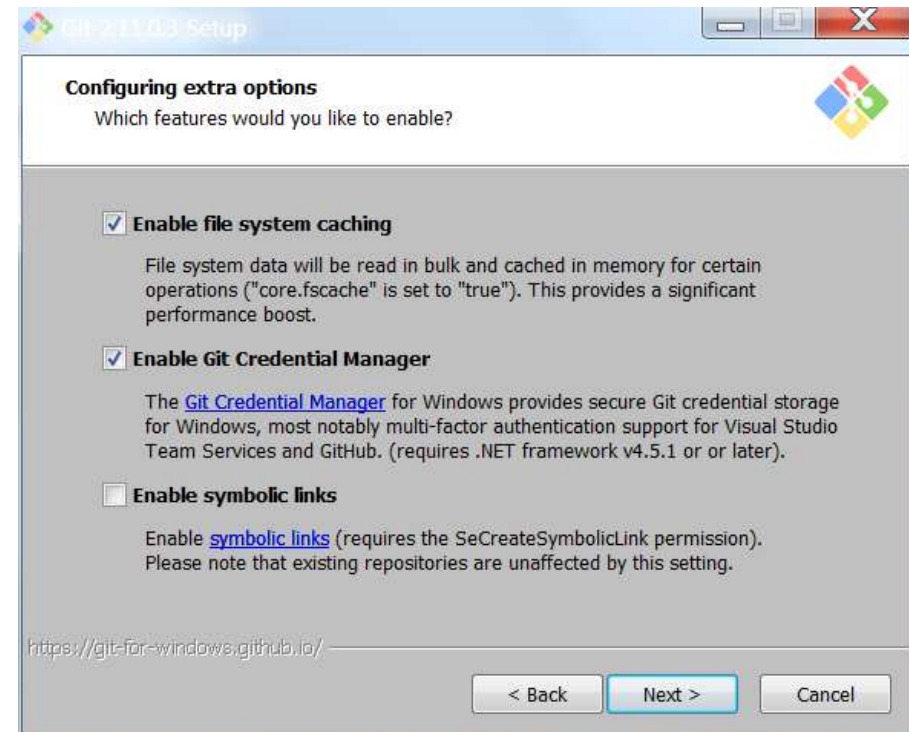
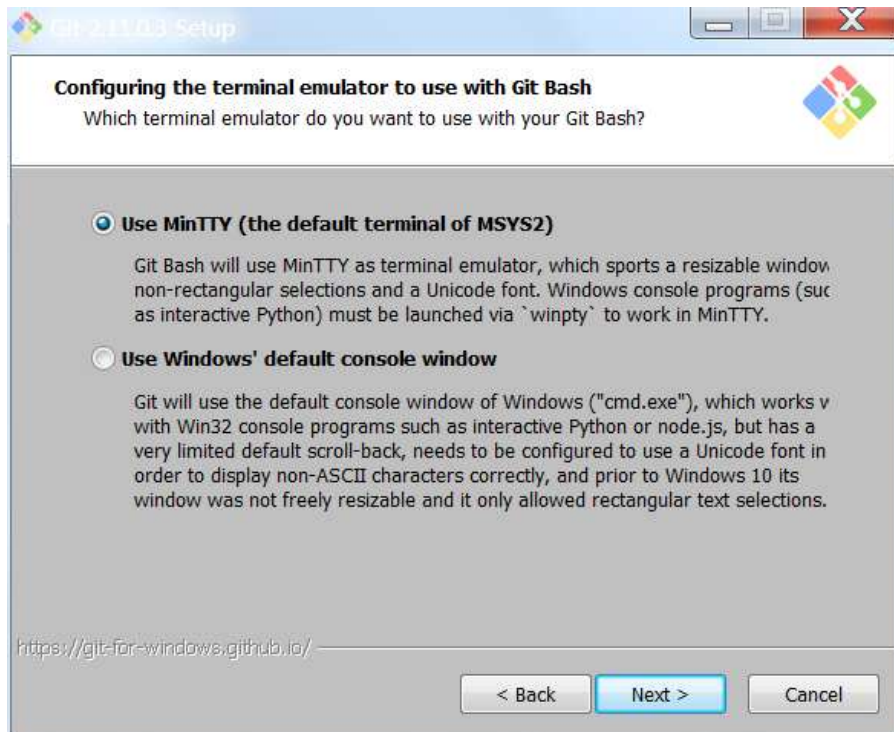
### 다운로드 받은 설치파일 클릭하기



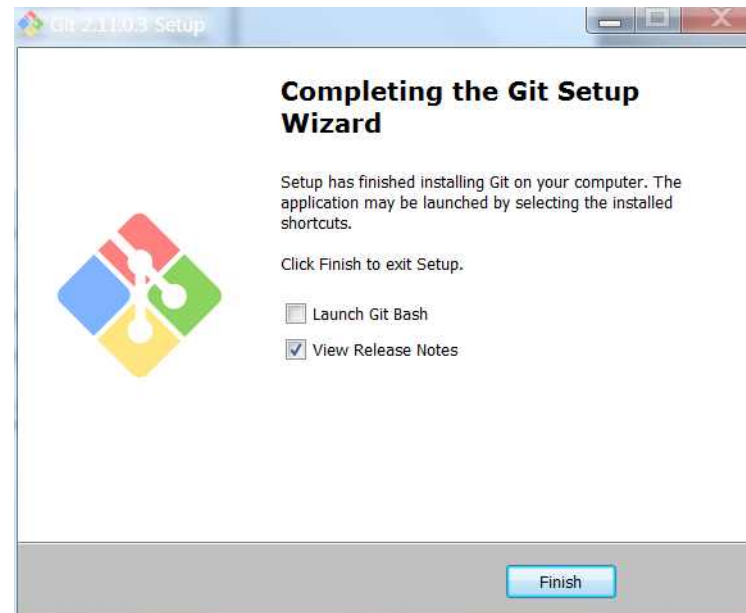
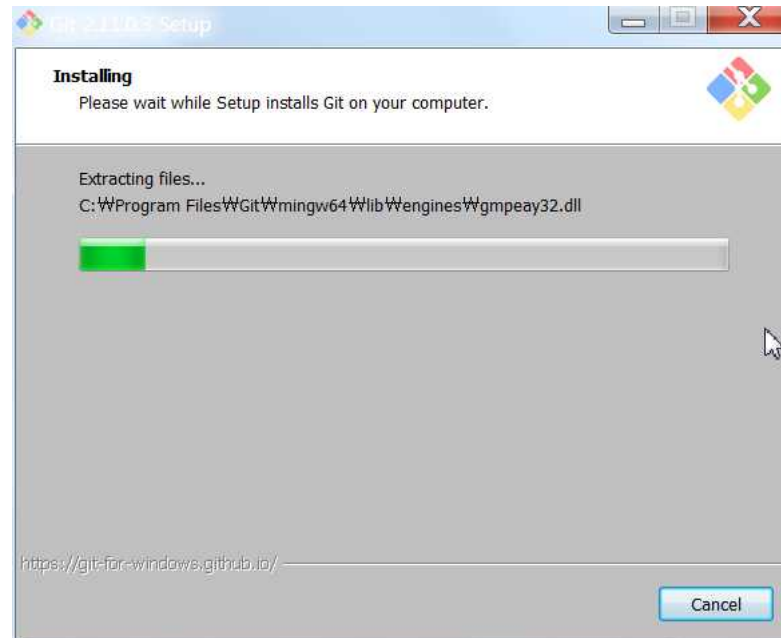
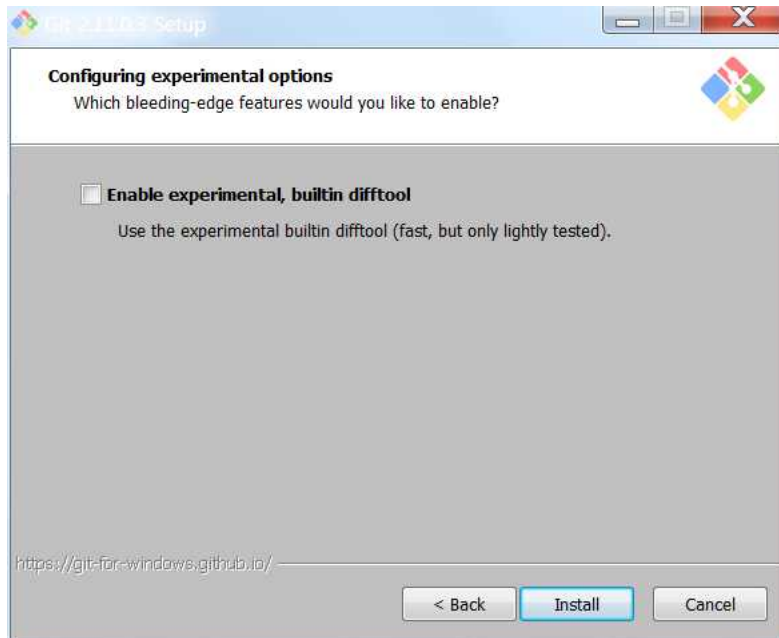
## □ 1) Git 개요







## □ 1) Git 개요





### ○ Git 기본 명령어

#### 저장소 사용에 필요한 Git 기본 명령어

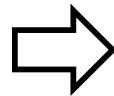
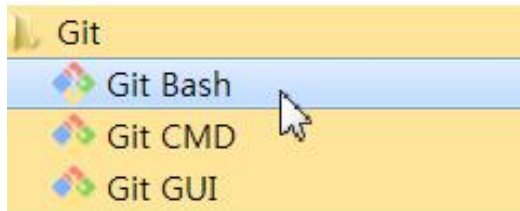
목표	명령어	설명
저장소 생성	git init	실행한 위치를 Git 저장소로 초기화합니다.
저장소에 파일 추가	git add 파일이름	해당 파일을 Git이 추적할 수 있게 저장소에 추가합니다.
저장소에 수정 내역 제출	git commit	변경된 파일을 저장소에 제출합니다.
저장소 상태 확인	git status	현재 저장소의 상태를 출력합니다.

#### 저장소 사용을 위한 branch 명령어

목표	명령어	설명
저장소에 브랜치 추가	git branch 이름	'이름'의 브랜치를 만듭니다.
작업 중인 브랜치 변경	git checkout 브랜치이름	현재 작업 중인 '브랜치이름'을 변경합니다.
브랜치 병합하기	git merge 브랜치이름	현재 작업 중인 브랜치에 '브랜치이름'의 브랜치를 끌어와 병합합니다.

## □ 1) Git 개요

### ○ 1. 저장소 생성 ( git init )



```
kyin@kyin-PC MINGW64 ~  
$ pwd  
/c/Users/kyin  
kyin@kyin-PC MINGW64 ~  
$
```



```
kyin@kyin-PC MINGW64 ~  
$ mkdir git_tutorial  
kyin@kyin-PC MINGW64 ~  
$
```

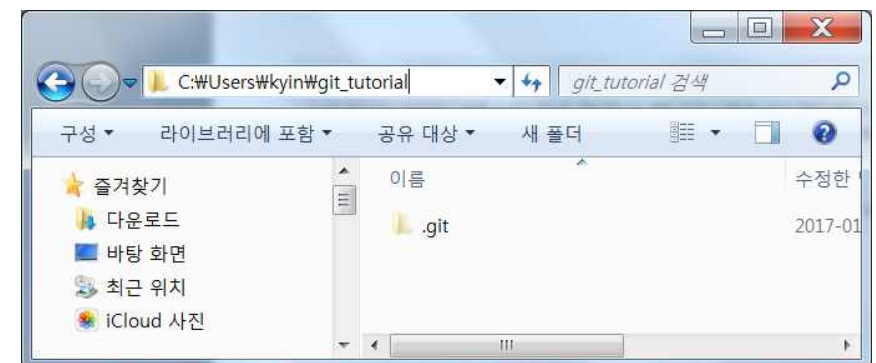


```
kyin@kyin-PC MINGW64 ~  
$ cd git_tutorial/  
kyin@kyin-PC MINGW64 ~/git_tutorial  
$
```



```
kyin@kyin-PC MINGW64 ~/git_tutorial  
$ git init  
Initialized empty Git repository in c:/Users/kyin/git_tutorial/.git/  
kyin@kyin-PC MINGW64 ~/git_tutorial (master)  
$
```

저장소: git\_tutorial  
master 브랜치 생성



## □ 1) Git 개요

### ○ 2. 파일 생성

```
MINGW64/c/Users/kyin/git_tutorial
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ vi hello.txt
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

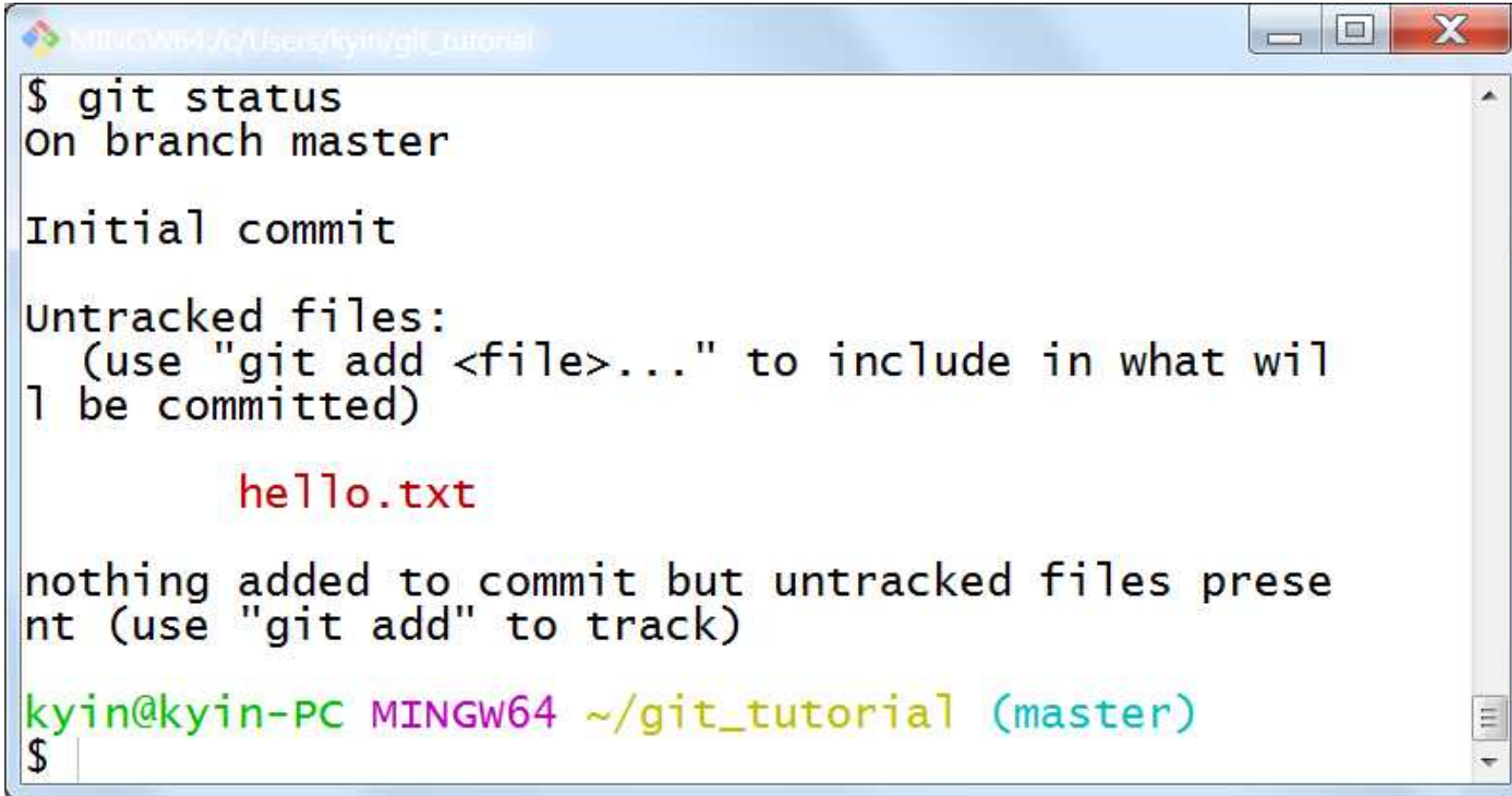
```
MINGW64/c/Users/kyin/git_tutorial
hello
~
~
~
~
<rial/hello.txt [unix] (17:04 25/01/2017)1,5 모두
:wq!
```

```
MINGW64/c/Users/kyin/git_tutorial
$ vi hello.txt
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ cat hello.txt
hello
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

- 0. a또는 i 키보드 클릭
- 1. Hello 문자열 입력
- 2. Esc 키
- 3. :wq! 엔터



### ○ 3. 저장소 상태 확인 ( git status )



```
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will
  be committed)

        hello.txt

nothing added to commit but untracked files present (use "git add" to track)

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

저장소에 Git이 아직 추적하지 않는 파일이 있음을 알려주는 화면이다.

### ○ 4. 저장소에 추적 시킬 파일 등록 ( git add 파일명 )

```
MINGW64/c/Users/kyin/git_tutorial

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git add hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory.

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git status
```

```
MINGW64/c/Users/kyin/git_tutorial

$ git status
On branch master

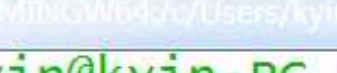
Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   hello.txt

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

## ○ 5. 저장소에 변경 파일 제출 ( git commit )



A screenshot of a Windows command prompt window. The title bar at the top reads "MINGW64/c:/Users/kyin/git\_L..." and includes standard minimize, maximize, and close buttons. The command prompt shows the user "kyin" at the host "kyin-PC" in the directory "~/gi". The prompt character is a dollar sign "\$". The command "git commit" has been entered and is currently being typed, with the cursor at the end of the text.

```
kyin@kyin-PC MINGW64 ~/gi
$
kyin@kyin-PC MINGW64 ~/gi
$ git commit
```

Commit 하기 위해서는 반드시 commit 메시지를 작성해야 된다.

The image is a composite of three screenshots from a Windows command prompt window titled "MINGW64/c/Users/kyin/git\_tutorial".

The top-left screenshot shows the initial commit message prompt. The text displayed is:

```
# Please enter the commit message for initial commit
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
Changes to be committed:
#   new file:   hello.txt
#
~
<it_tutorial/.git/COMMIT_EDITMSG
```

A red box highlights the text "Hello commit message" in the prompt, and an arrow points to it from the right.

The top-right screenshot shows the same prompt with the message "Hello commit message" entered. The text displayed is:

```
# Please enter the commit message for initial commit
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
Changes to be committed:
#   new file:   hello.txt
#
~
<it_tutorial/.git/COMMIT_EDITMSG
```

The bottom screenshot shows the output of the "git commit" command. The text displayed is:

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git commit
[master (root-commit) 309e1a5] Hello commit message
1 file changed, 1 insertion(+)
create mode 100644 hello.txt
```

### ○ 새로운 브랜치(branch) 생성과 이동

브랜치 목록 보기 ( git branch )



```
MINGW64/c/Users/kyin/git_tutorial
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git branch
* master

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

hotfix 브랜치 생성 ( git branch 브랜치명 )



```
MINGW64/c/Users/kyin/git_tutorial
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git branch hotfix

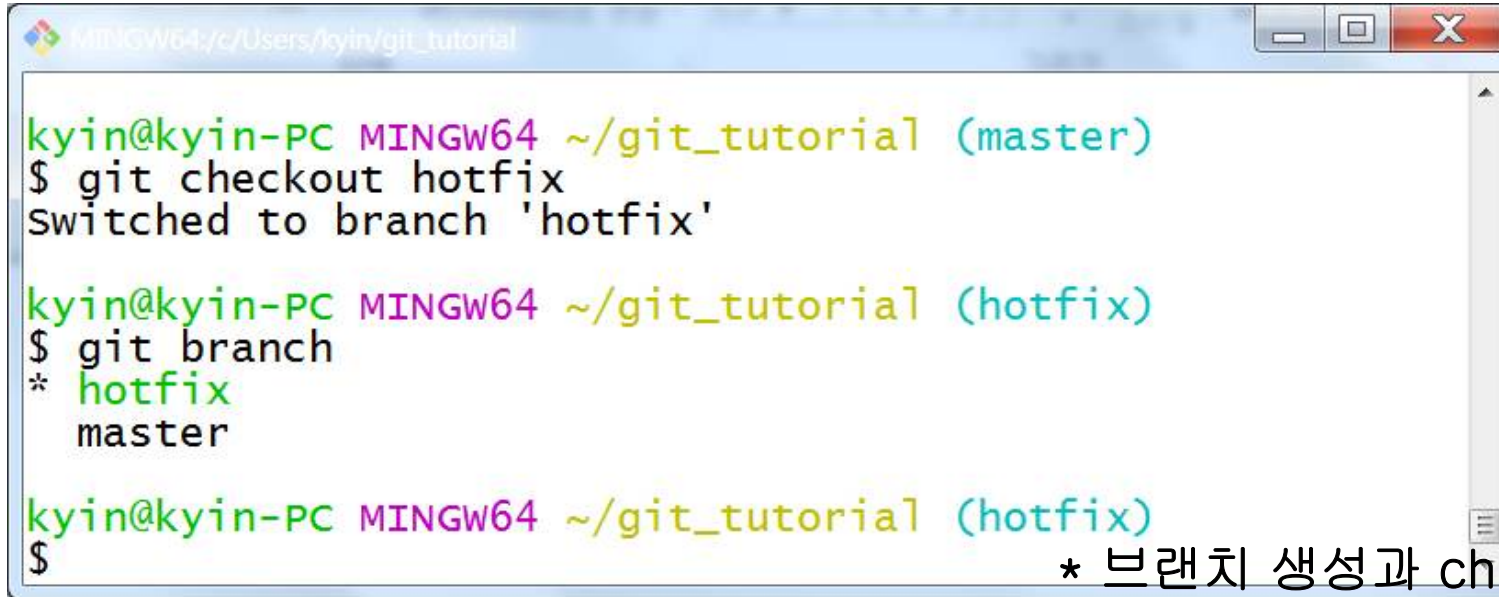
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git branch
hotfix
* master

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

\*는 사용중인 브랜치 의미



Master에서 hotfix 브랜치로 이동 ( git checkout 브랜치명 )



```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git checkout hotfix
Switched to branch 'hotfix'

kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ git branch
* hotfix
  master

kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$
```

\* 브랜치 생성과 checkout 한꺼번에 작업 방법  
git checkout -b 브랜치이름

지금부터 하는 모든 작업은 hotfix 브랜치에만 적용이 된다. (파일수정, 삭제 등)  
마음껏 작업하고 commit 한 후에 master로 checkout하면, hotfix에서 했던 모든  
것을 해당 브랜치의 최종 commit 상태로 보존한 후 master 브랜치의 최종 커밋 상태로  
파일들이 변경된다.

## hotfix 브랜치에서 hello.txt 파일 수정

The first screenshot shows a terminal window titled "MINGW64/c/Users/kyin/git\_tutorial". The prompt is "kyin@kyin-PC MINGW64 ~/git\_tutorial (hotfix)". The user enters "\$ vi hello.txt".

The second screenshot shows the same terminal window after opening the file. It displays "hello world" followed by several tilde (~) symbols indicating more lines. At the bottom, there is a status bar that reads "<tutorial/hello.txt[+] [unix] (17:13 25/01/2017)2,5 모두 :wq!".

The third screenshot shows the terminal window after saving the file. The prompt is "kyin@kyin-PC MINGW64 ~/git\_tutorial (hotfix)". The user enters "\$ cat hello.txt", and the output is "hello world".

## □ 1) Git 개요

hotfix 브랜치에서 hello.txt 파일 등록 및 제출

```
kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ git status
On branch hotfix
Changes not staged for commit:
  (use "git add <file>..." to update what will be commit
ted)
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ git add hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working dir
ectory.

kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ git commit
```

```
world Commit Message
# Please enter the commit message for your change
# with '#' will be ignored, and an empty message
# aborts the commit.
# On branch hotfix
# Changes to be committed:
#   modified:   hello.txt
~
~
~
<torial/.git/COMMIT_EDITMSG[+] [unix] (18:07 25/01/2017)7,1 모두
:wq!
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ git status
On branch hotfix
nothing to commit, working tree clean
```

\*변경된 저장소 파일 모두 커밋  
git commit -a

## □ 1) Git 개요

master 와 hotfix 브랜치에서 hello.txt 파일 내용 비교

```
kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ git checkout master
Switched to branch 'master'

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ cat hello.txt
hello

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git checkout hotfix
Switched to branch 'hotfix'

kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ cat hello.txt
hello
world


kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$
```



## □ 1) Git 개요

### ○ 병합 ( merge ) : git merge 브랜치명

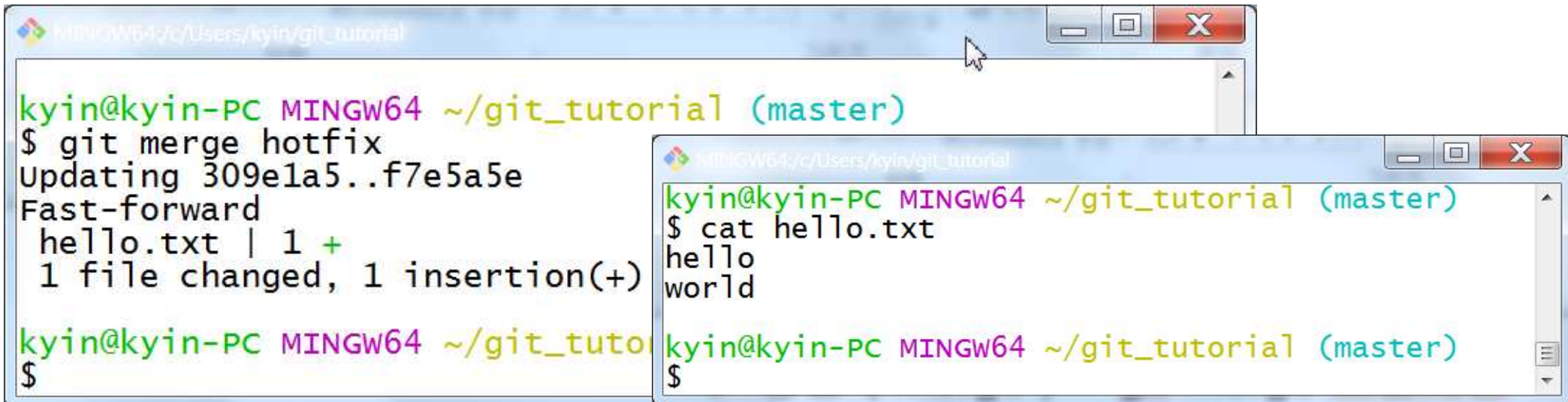
#### 1. master 브랜치로 이동



```
kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ git checkout master
Switched to branch 'master'

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

#### 2. master 브랜치와 hotfix 브랜치 병합



```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git merge hotfix
Updating 309e1a5..f7e5a5e
Fast-forward
 hello.txt | 1 +
 1 file changed, 1 insertion(+)

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ cat hello.txt
hello
world

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

### ○ 불필요한 파일 무시 ( .gitignore )

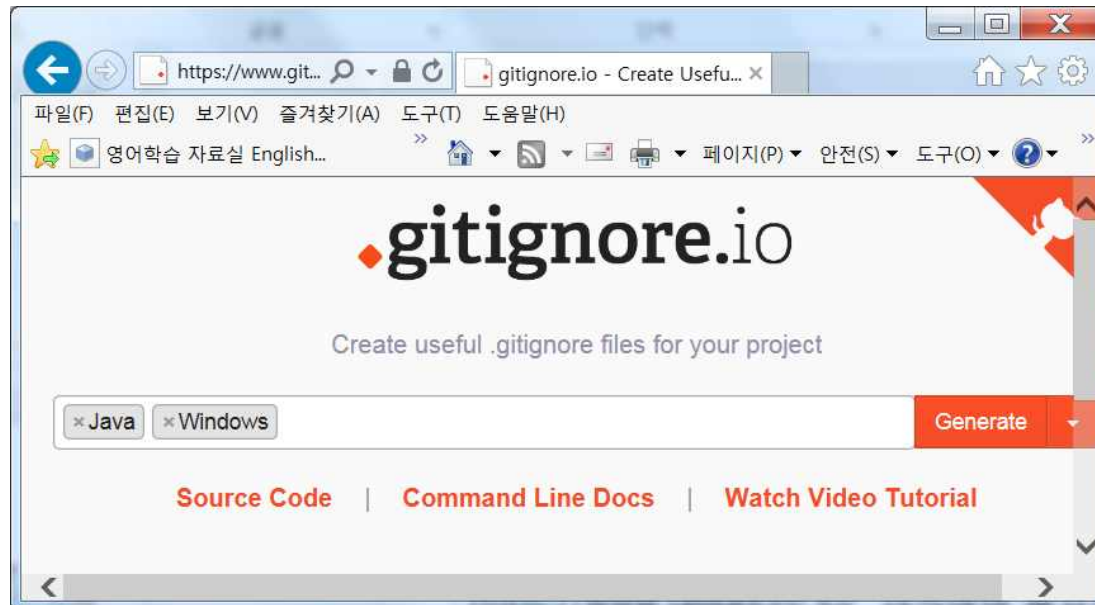
저장 및 추적할 필요가 없는 부수적인 파일들 목록을 만들어서 제외 시키는 방법.



```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ touch .gitignore

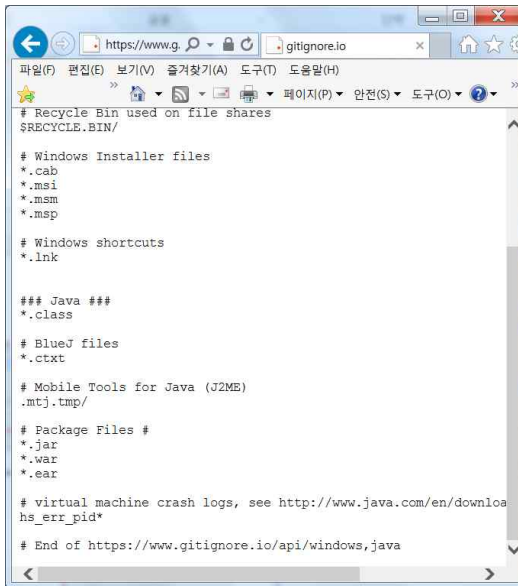
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

<https://www.gitignore.io/> 접속하여 제외시킬 파일 정보 등록



## □ 1) Git 개요

.gitignore 파일에 내용복사



```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ cat .gitignore
# Created by https://www.gitignore.io/api/windows,java

### windows ###
# windows thumbnail cache files
Thumbs.db
ehthumbs.db
```

.gitignore 파일을 저장소에 등록 및 제출

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git add .gitignore

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git commit -m 'add .gitignore file'
[master ddcca3b] add .gitignore file
1 file changed, 43 insertions(+)
create mode 100644 .gitignore

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```



## □ 1) Git 개요

### ○ 충돌 해결

전제 조건은 프로그램 파일의 같은 행에 서로 다른 변경사항이 있는 경우에 충돌 발생됨.

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ cat hello.txt
hello
world

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git commit
On branch master
Changes not staged for commit:
  modified:   hello.txt

no changes added to commit
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ vi hello.txt
```

```
hello
world
master
~
~
~
<ello.txt[+] [dos] (18:32 25/01/2017)3,6 모두
:wq!
```

```
master
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# changes to be committed:
  <DITMSG[+] [unix] (17:22 26/01/2017)1,6 꼭대기
:wq!
```

```
$ git add hello.txt
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git commit
```

## □ 1) Git 개요

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git checkout hotfix
Switched to branch 'hotfix'

kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ cat hello.txt
hello
world

kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$
```

```
kyin@kyin-PC MINGW64
$ vi hello.txt
```

```
hello
world
hotfix
~
~
<hello.txt[+] [dos] (17:27 26/01/2017)3,6 모두
:wq!
```

```
kyin@kyin-PC MINGW64 ~/git_tutorial (hotfix)
$ git commit -m 'hotfix'
[hotfix 0a693b8] hotfix
1 file changed, 1 insertion(+)
```

```
kyin@kyin-PC MINGW64
$ git add hello.txt
```

## □ 1) Git 개요

마지막으로 master에서 hotfix 브랜치 내용과 병합할 때 충돌이 발생됨.  
( 각각 같은 행을 수정했기 때문 )

```
kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git merge hotfix
Auto-merging hello.txt
CONFLICT (content): Merge conflict in hello.txt
Automatic merge failed; fix conflicts and then c
ommit the result.

kyin@kyin-PC MINGW64 ~/git_tutorial (master|MERC
ING)
$
```

병합 도중 충돌이 발생되어 해결 하는중임.

```
$
kyin@kyin-PC MINGW64 ~/git_tutorial (master|MERGING)
$ git commit -a -m 'master hotfix'
[master f686739] master hotfix

kyin@kyin-PC MINGW64 ~/git_ $ cat hello.txt
hello
world
master
hotfix
```

```
$ cat hello.txt
hello
world
<<<<<<< HEAD
master
=====
hotfix
>>>>>>> hotfix

kyin@kyin-PC MINGW64 ~/git_tutorial (master|MERCING)
$
```

```
kyin@kyin-PC MINGW64 .
$ vi hello.txt
```

직접 수정

```
hello
world
master
hotfix

~
~
~
<1/2017>2,5 모두
:wq!
```



chapter 02.

---

# GitHub 개요

### ○ 원격 저장소와 GitHub

Git 원격 저장소를 제공하는 대표적인 서비스가 GitHub 이다. GitHub는 단순히 원격 저장소만을 제공하는 것이 아니라, 여러 가지 프로젝트 진행을 원활하게 하는 도구를 함께 제공한다.

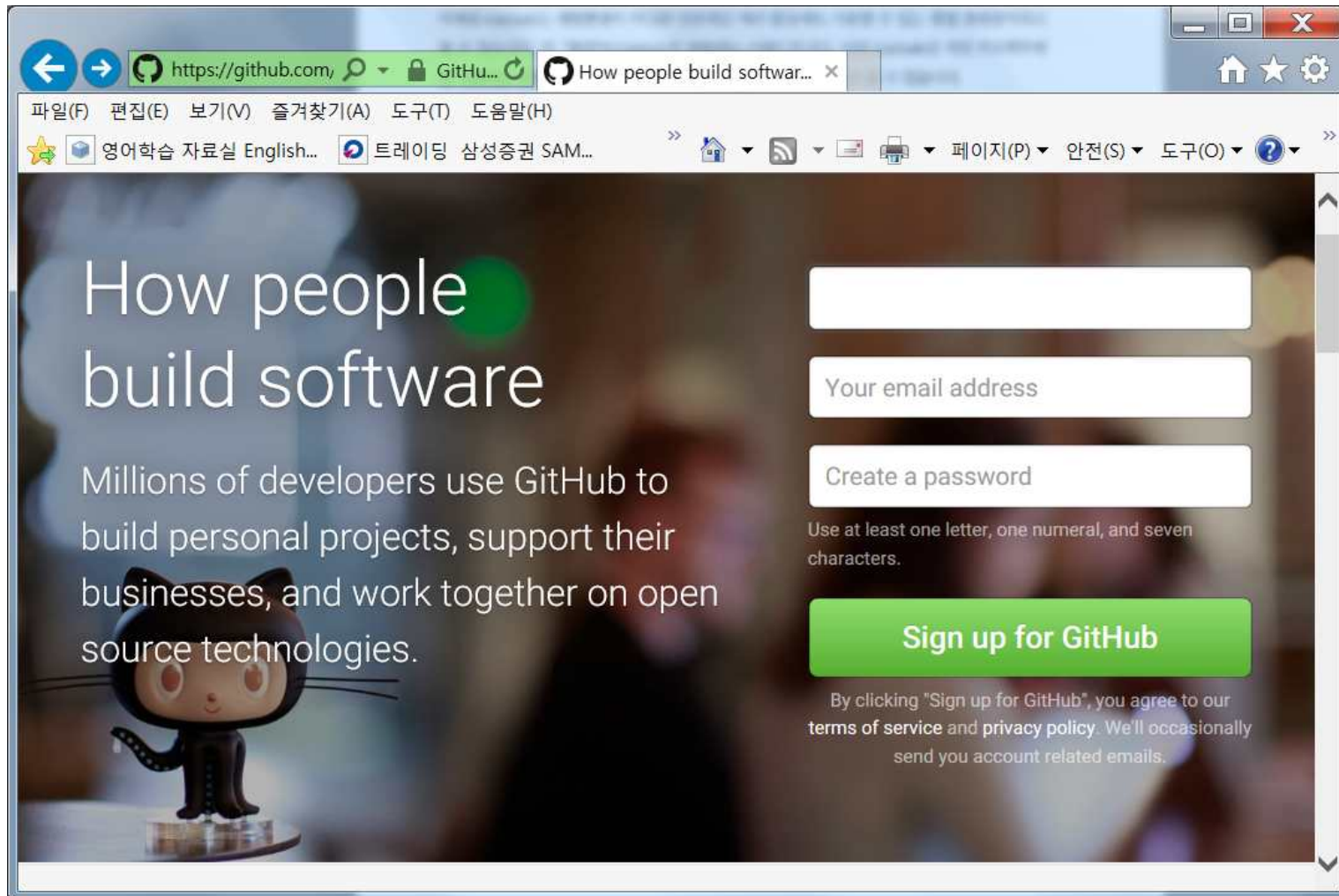
### ○ GitHub 장점

- 전 세계에서 진행되는 오픈 소스 프로젝트가 많이 모여 있어서 이에 참여하고 오픈 소스에 기여할 수 있는 기회가 제공된다.
- 개발자는 GitHub를 이용해 자신이 작성했던 코드 그 자체를 곧바로 제공할 수 있다.
- IT 개발과 관련된 많은 디자이너도 여태껏 사용했던 작품을 포트폴리오로 공개 가능.
- 기획자 역시 자신이 준비했던 기획 문서를 공개할 수 있다.



### ○ GitHub 가입하기

- <http://www.github.com> 접속



Create your personal account

Username

This will be your username — you can enter

Email Address

You will occasionally receive account related email with anyone.

Password

Use at least one lowercase letter, one num

By clicking on "Create an account" button of Service and the Privacy Policy.

Create an account

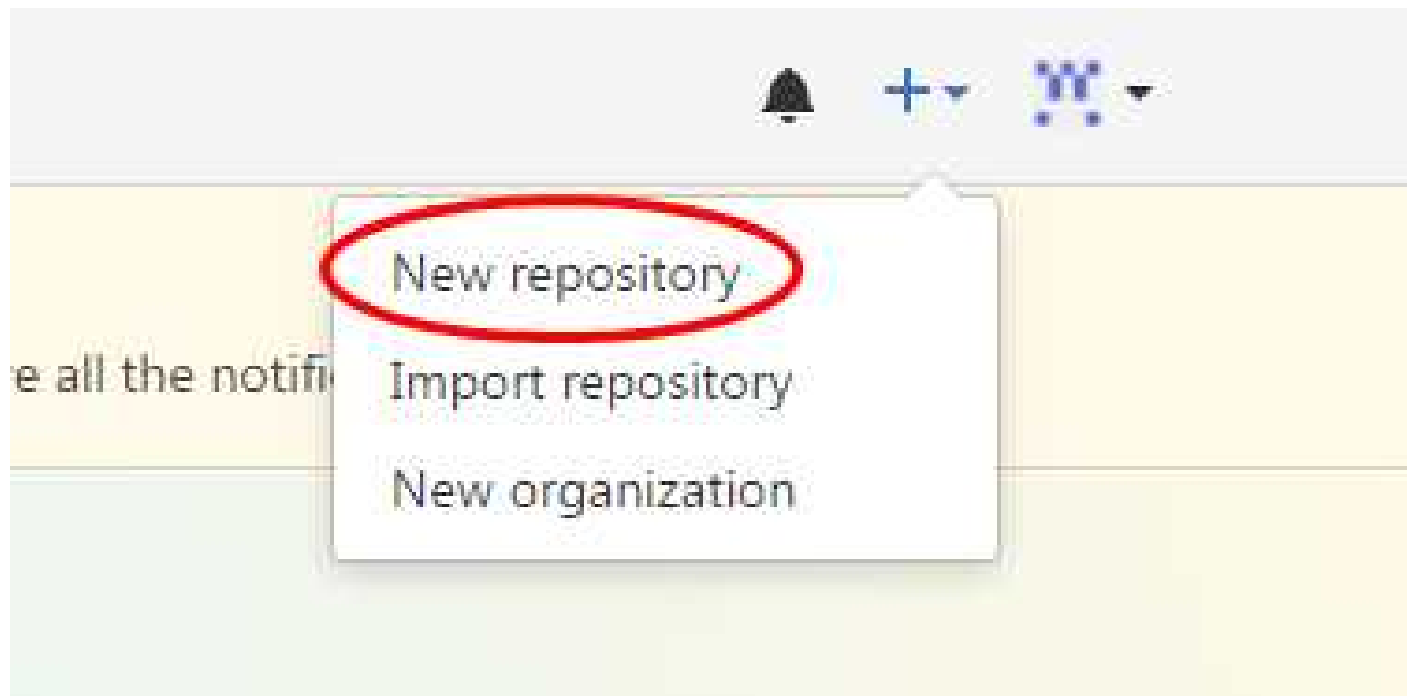
이름  
이메일  
비밀번호

### ○ 원격 저장소 생성

1. [www.github.com](https://www.github.com) 에서 회원 가입

예> > 이름:inky4832 [inky4832@daum.net](mailto:inky4832@daum.net) / 08XXXXXXXX


2. Remote 저장소 생성



### Create a new repository

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

Owner

 inky4832 ▾

Repository name

SamplePro ✓

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

Description (optional)

☒  **Public**

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

☐  **Private**

You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

## □ 2) GitHub 개요

inky4832 / SamplePro

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

No description or website provided. — Edit

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

inky4832 Initial commit

README.md Initial commit

README.md

SamplePro

Clone with HTTPS ⓘ Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/inky4832/SamplePro.git

Copy to clipboard

Open in Desktop Download ZIP

### ○ 원격 저장소와 Git 명령어

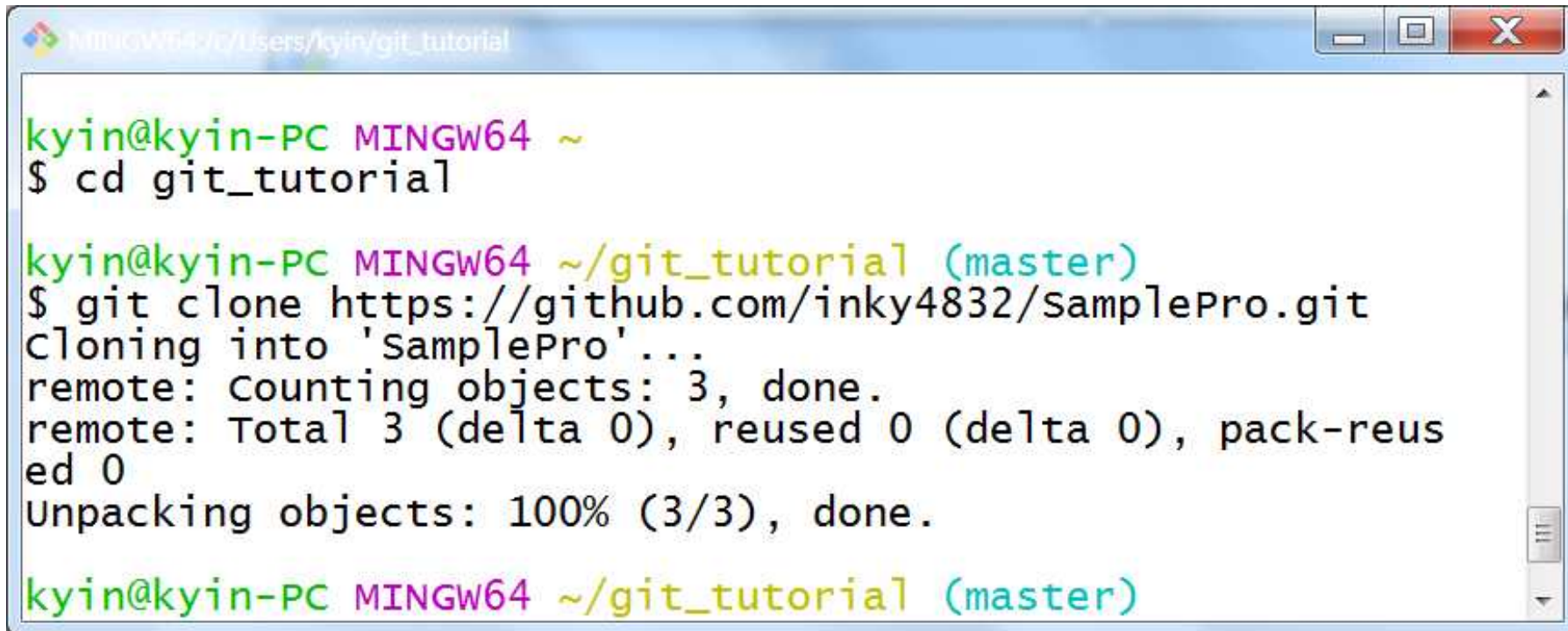
Git 명령어

명령어	기능
git clone	원격 저장소의 모든 내용을 로컬 저장소로 복사합니다.
git remote	로컬 저장소를 특정 원격 저장소와 연결합니다.
git push	로컬 저장소의 내용을 보내거나 로컬 저장소의 변경 사항을 원격 저장소로 보냅니다.
git fetch	로컬 저장소와 원격 저장소의 변경 사항이 다를 때 이를 비교 대조하고 git merge 명령어와 함께 최신 데이터를 반영하거나 충돌 문제 등을 해결합니다.
git pull	git remote 명령을 통해 서로 연결된 원격 저장소의 최신 내용을 로컬 저장소로 가져오면서 병합합니다. git push와 반대 성격의 명령어입니다.

### ○ 원격 저장소 내용을 로컬 저장소로 가져오기 ( git clone )

내가 생성한 원격 저장소를 내 컴퓨터와 연결해서 데이터를 복사하는 작업.

```
git clone https://github.com/사용자이름/SamplePro.git
```



```
MINGW64: c:/Users/kyin/git_tutorial

kyin@kyin-PC MINGW64 ~
$ cd git_tutorial

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
$ git clone https://github.com/inky4832/SamplePro.git
Cloning into 'SamplePro'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.

kyin@kyin-PC MINGW64 ~/git_tutorial (master)
```



## □ 2) GitHub 개요

```
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro
$ cd SamplePro

kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean

kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$
```

현재 접속된 원격 저장소 정보 확인

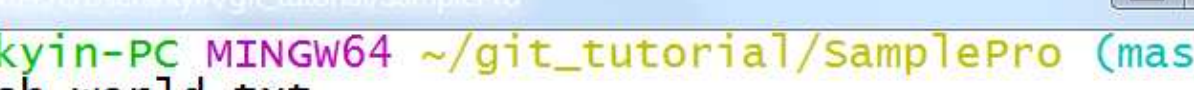
origin 은 원격 저장소의 별명

```
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$ git remote -v
origin  https://github.com/inky4832/samplePro.git (fetch)
origin  https://github.com/inky4832/samplePro.git (push)


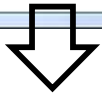
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$
```

## ● 로컬 작업내역을 원격 저장소에 올리기( git push )

기본적으로 commit 된 내역들을 원격 저장소의 master 브랜치에 업로드 한다.



```
MINGW64 ~/Users/kyin/git_tutorial/SamplePro
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$ touch world.txt
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$ vi world.txt
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$
```



The screenshot shows a Windows command prompt window titled "MINGW64 C:\Users\kyin\git\_tutorial\SamplePro". The command prompt displays the command `cat world.txt` and its output, which is the text `world` followed by several tilde characters (`~`) on separate lines. The status bar at the bottom of the window shows the current directory as `<SamplePro/world.txt[+] [unix]`, the date and time as `(11:56 27/01/2017)`, and the file size as `1,5` bytes. The prompt is `:wq!`.



## □ 2) GitHub 개요

```
MINGW64/c/Users/kyin/git_tutorial/SamplePro
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$ git add world.txt
warning: LF will be replaced by CRLF in world.txt.
The file will have its original line endings in your working directory.

kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$ git commit -m 'world.txt file'
[master 74967d9] world.txt file
1 file changed, 1 insertion(+)
create mode 100644 world.txt

kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$
```



```
MINGW64/c/Users/kyin/git_tutorial/SamplePro
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (mast
$ git push origin master
```



git push 원격저장소별칭 로컬브랜치명

git push 원격저장소별칭 --all

## □ 2) GitHub 개요

```
kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/inky4832/samplePro.git
    b2b6c20..74967d9  master -> master

kyin@kyin-PC MINGW64 ~/git_tutorial/SamplePro (master)
$
```



inky4832 / SamplePro

<> Code Issues 0 Pull requests 0 Projects 0 Wik

No description or website provided.

2 commits 1 branch

Branch: master New pull request

개발자1 world.txt file

README.md Initial commit

**world.txt** world.txt file

README.md



Branch: master SamplePro / world.txt

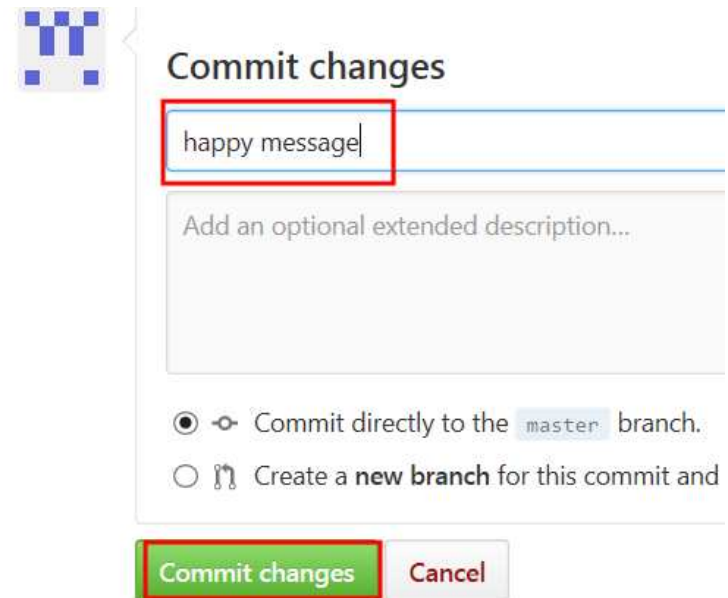
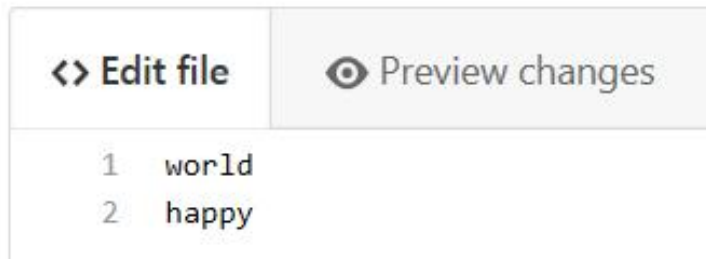
개발자1 world.txt file

0 contributors

2 lines (1 sloc) 6 Bytes

1 world

### ○ 원격 저장소 내용을 로컬 저장소로 가져오기( git pull )




## □ 2) GitHub 개요

```
MINGW64/c/Users/kyin/git_tutorial/SamplePro

kyin@kyin-PC MINGW64 ~/git_tutorial/samplePro (master)
$ git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused
0
Unpacking objects: 100% (3/3), done.
From https://github.com/inky4832/SamplePro
 * branch                master      -> FETCH_HEAD
  74967d9..893e6d3 master      -> origin/master
Updating 74967d9..893e6d3
Fast-forward
 world.txt | 1 +
 1 file changed, 1 insertion(+)

kyin@kyin-PC MINGW64 ~/git_tutorial/samplePro (master)
$
```



```
MINGW64/c/Users/kyin/git_tutorial/SamplePro

kyin@kyin-PC MINGW64 ~/git_tutorial/samplePro (master)
$ cat world.txt
world
happy

kyin@kyin-PC MINGW64 ~/git_tutorial/samplePro (master)
$
```



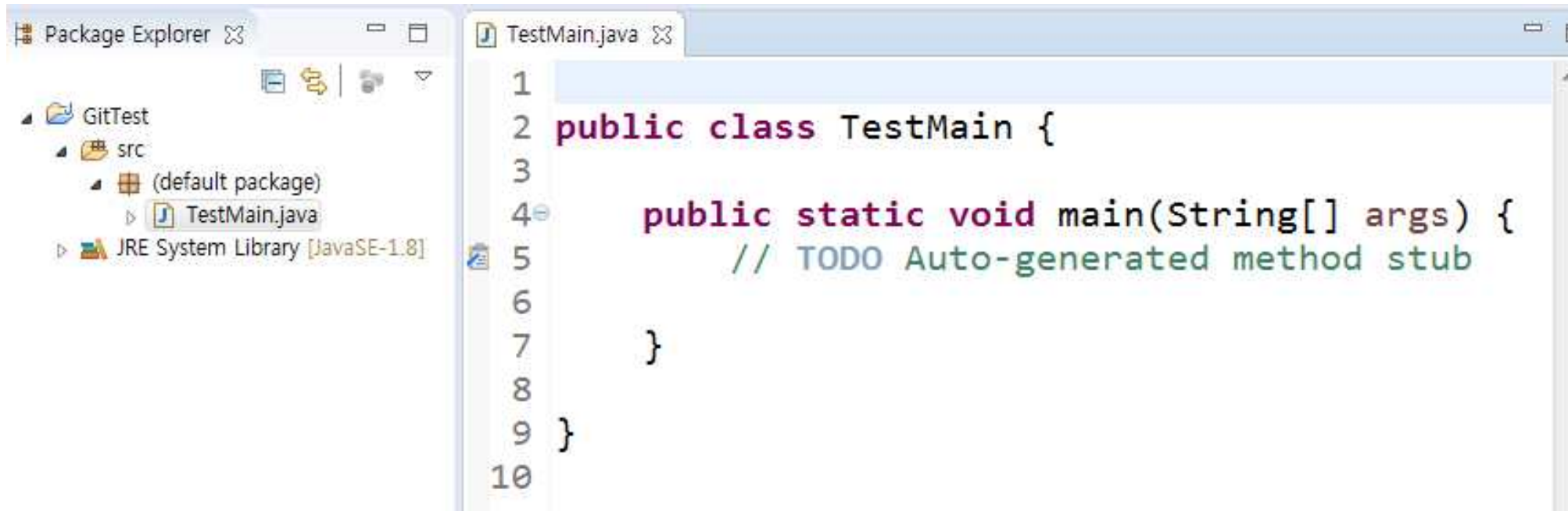
chapter 03.

---

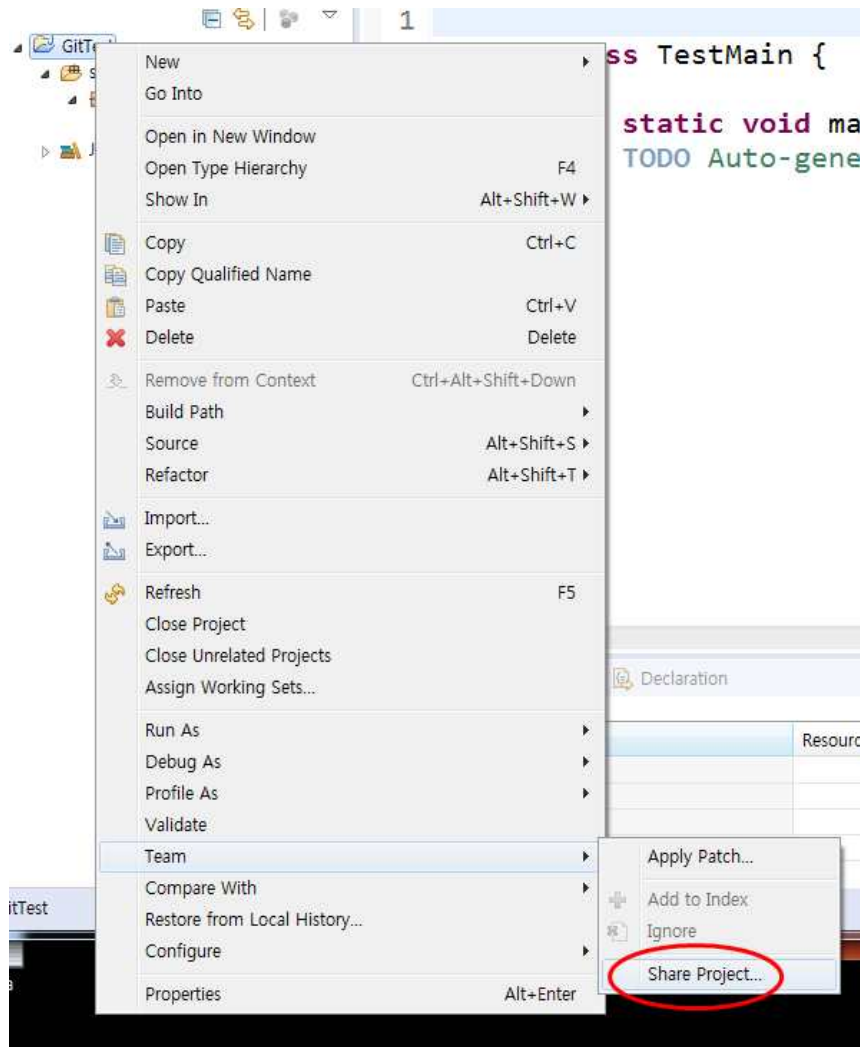
# Eclipse에서 GitHub 사용



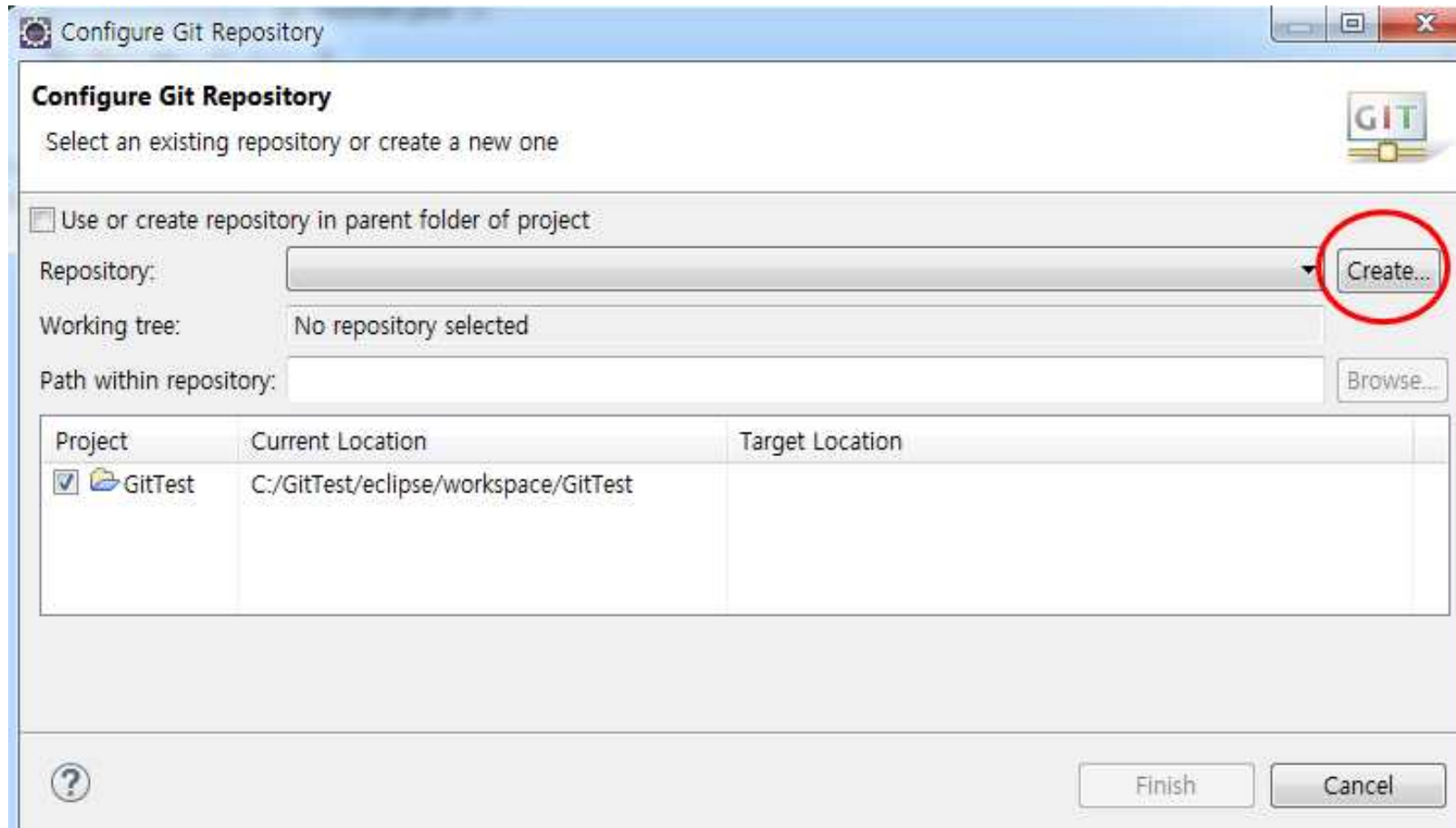
### 1. 이클립스에서 프로젝트 생성 및 소스 작성



### 2. 이클립스에서 Remote 저장소 알려주기1

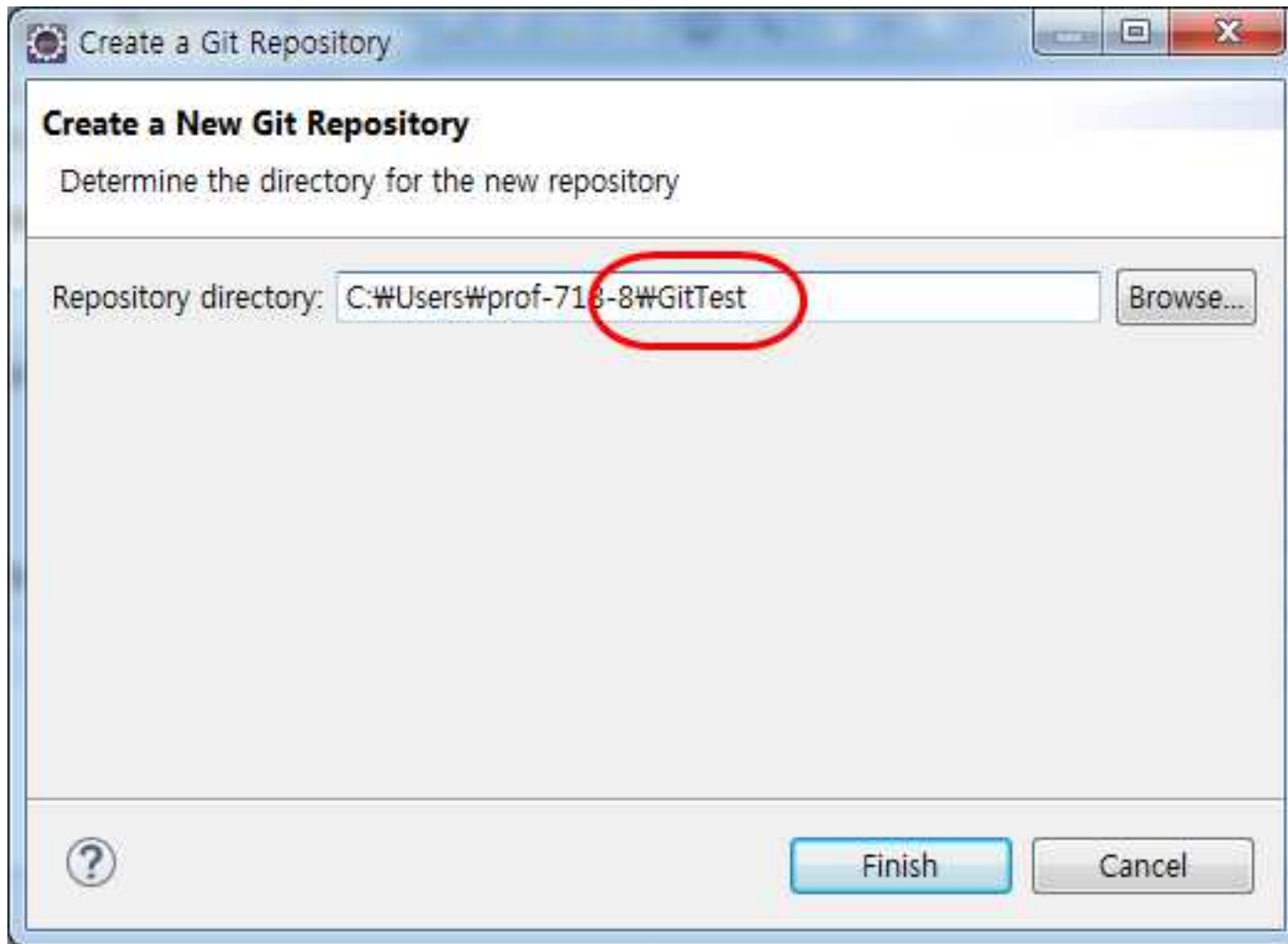


### 3. 이클립스에서 Remote 저장소 알려주기2- 로컬저장소 지정하기

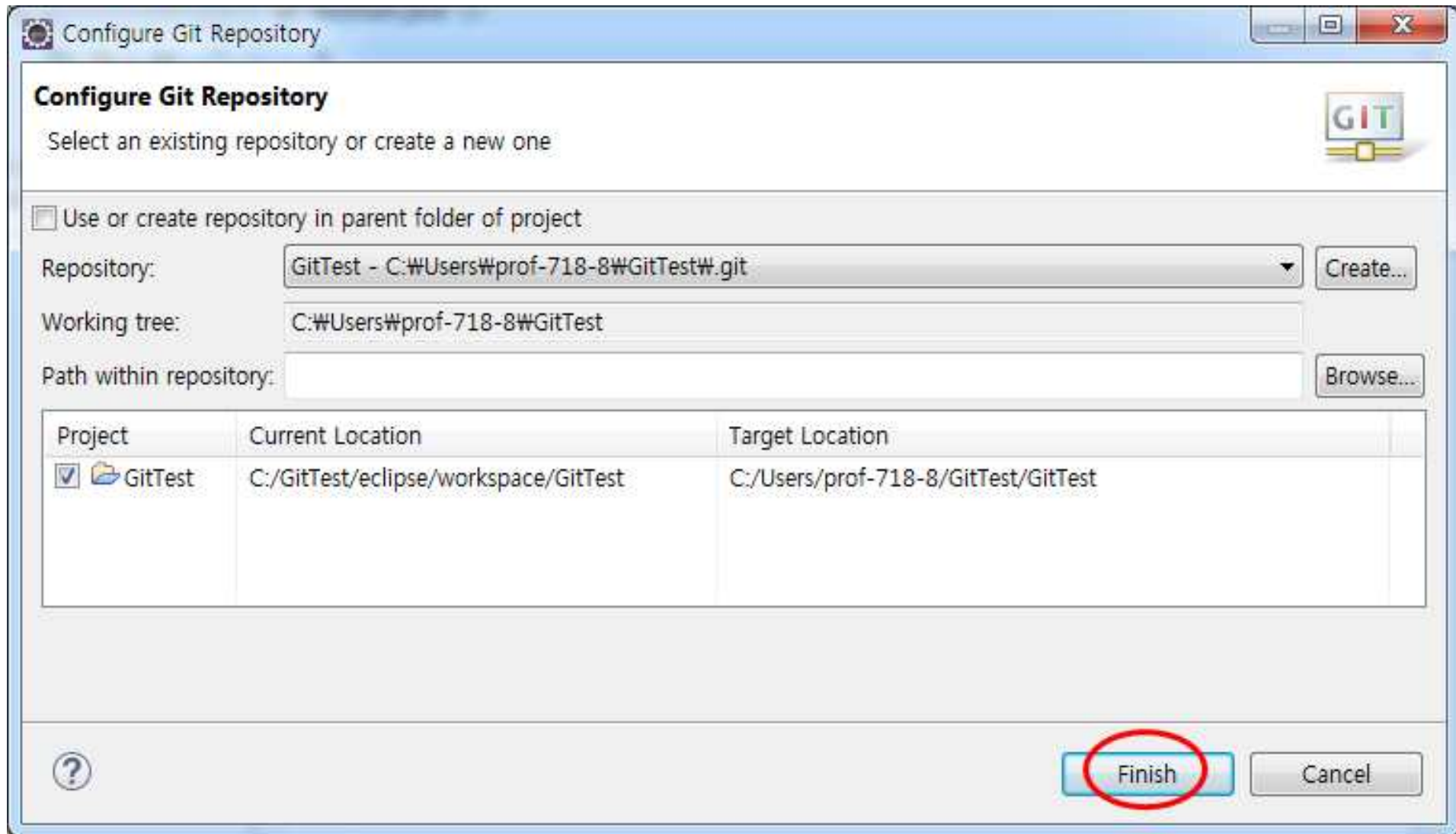


### ❑ 3) Eclipse에서 GitHub 사용

임의의 디렉토리를 지정하여 로컬 저장소를 설정한다.

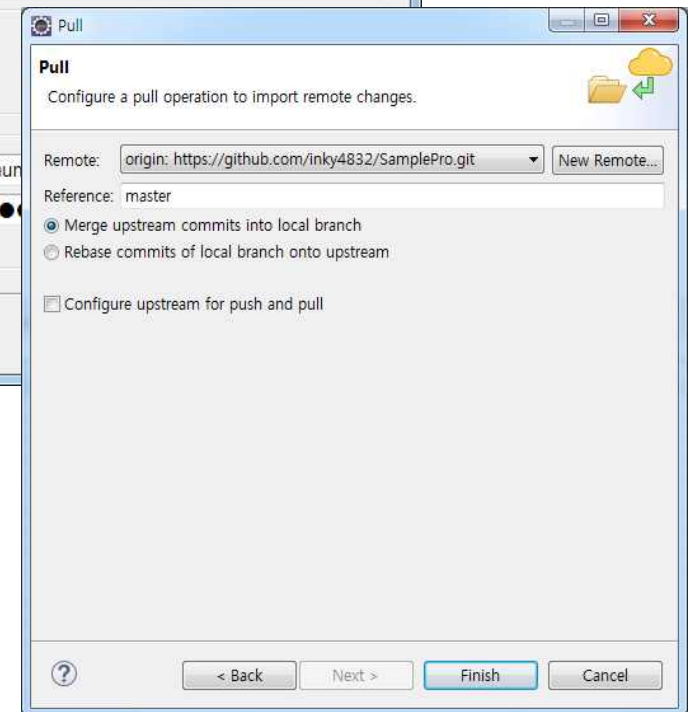
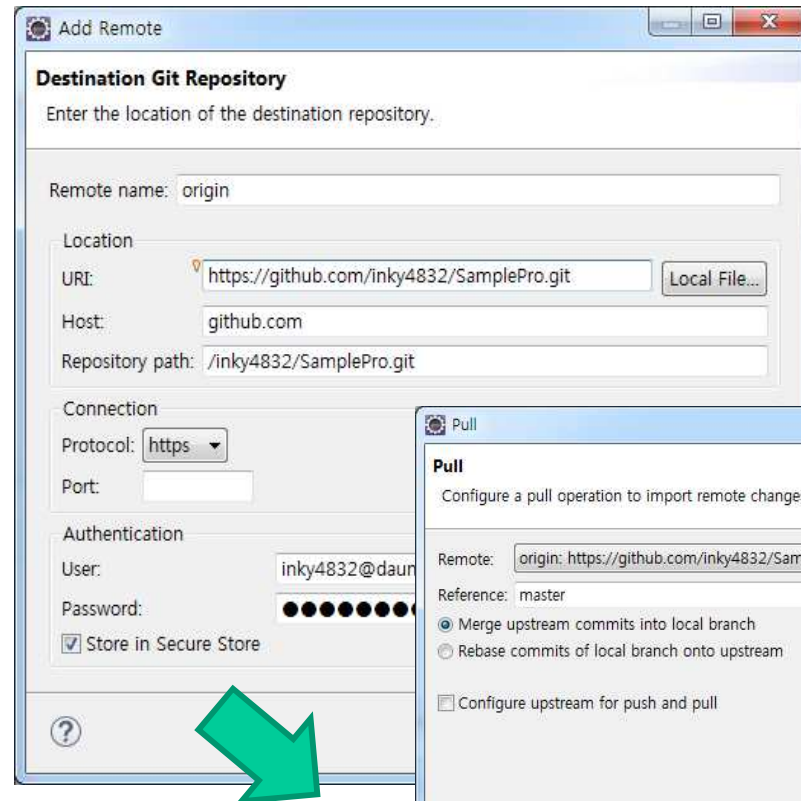
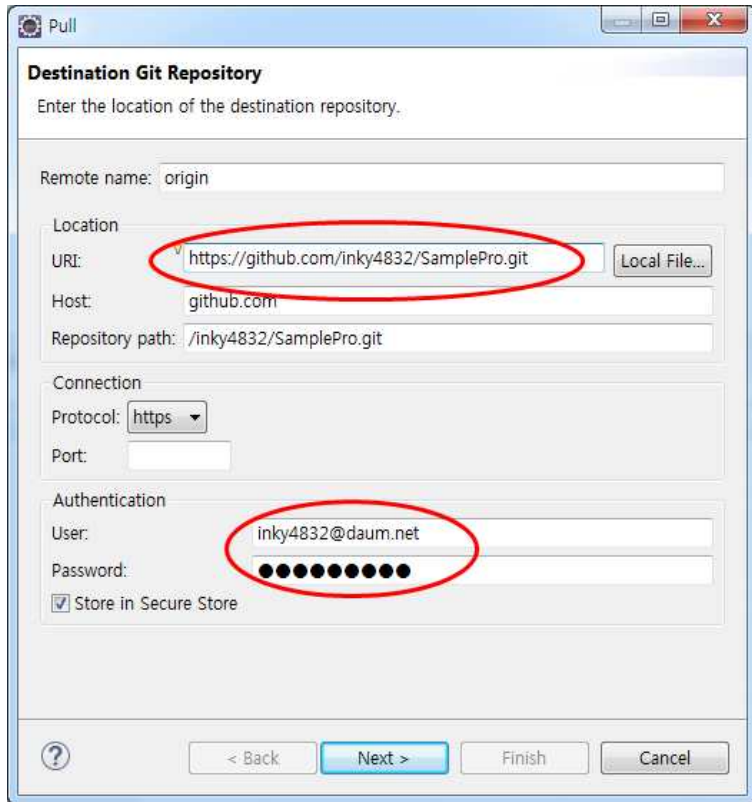


### ❑ 3) Eclipse에서 GitHub 사용



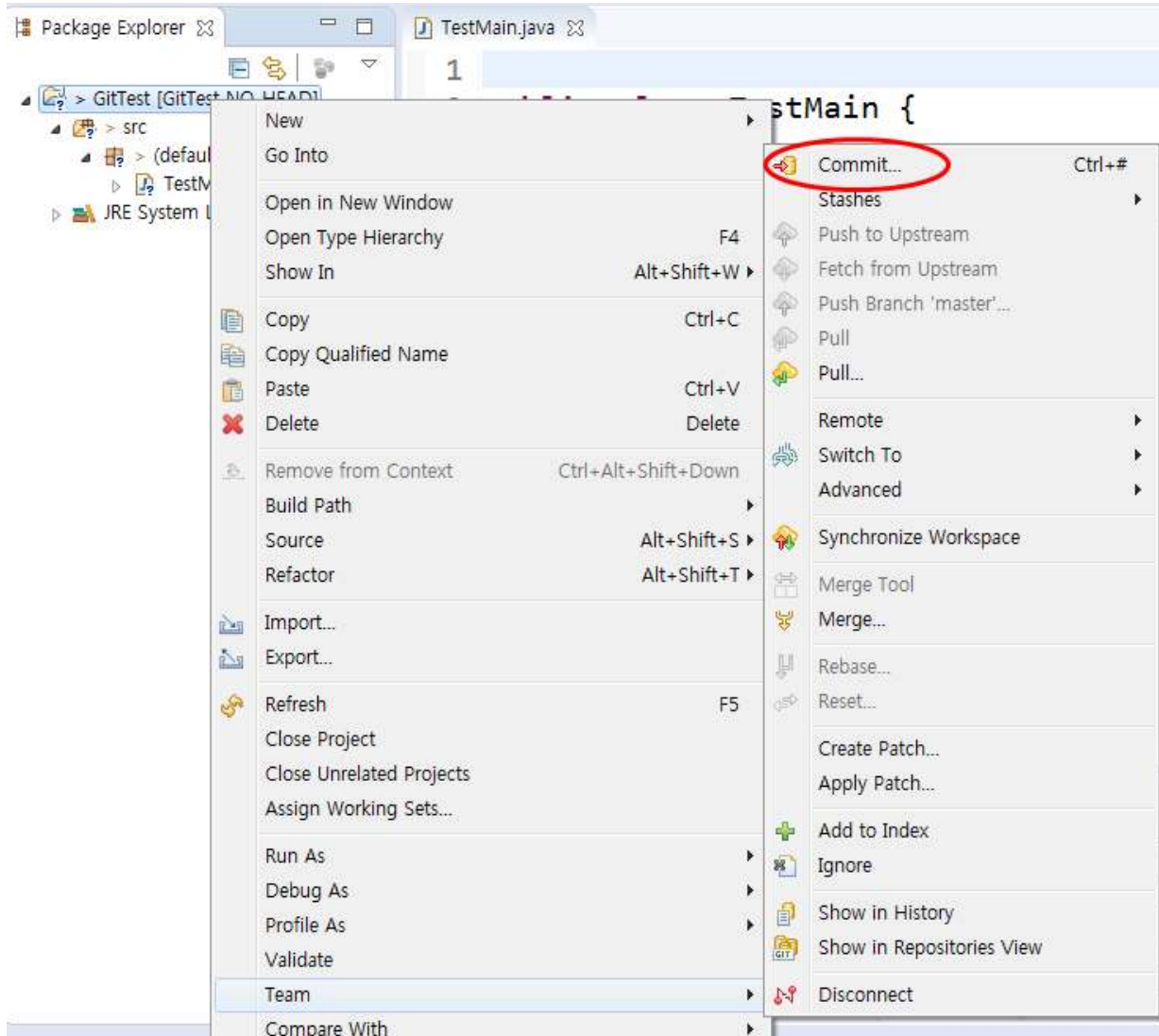


#### 4. Remote 저장소로부터 로컬 저장소로 가져오기 ( Pull )

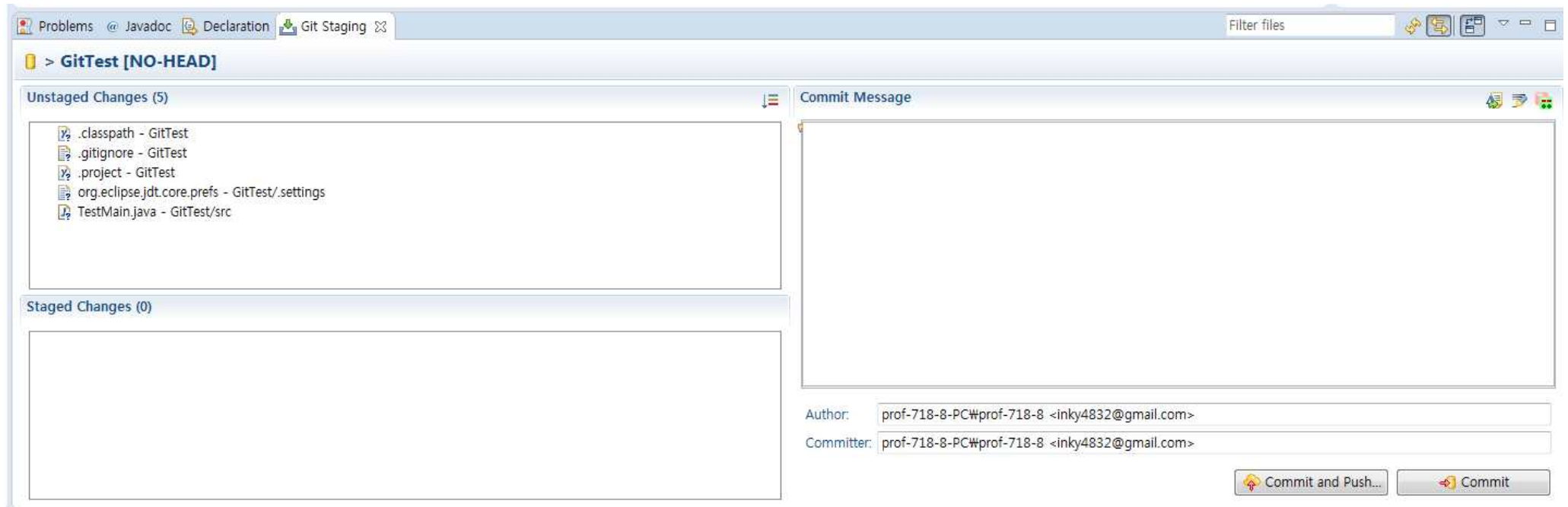


Commit 하기 전에 PULL 먼저

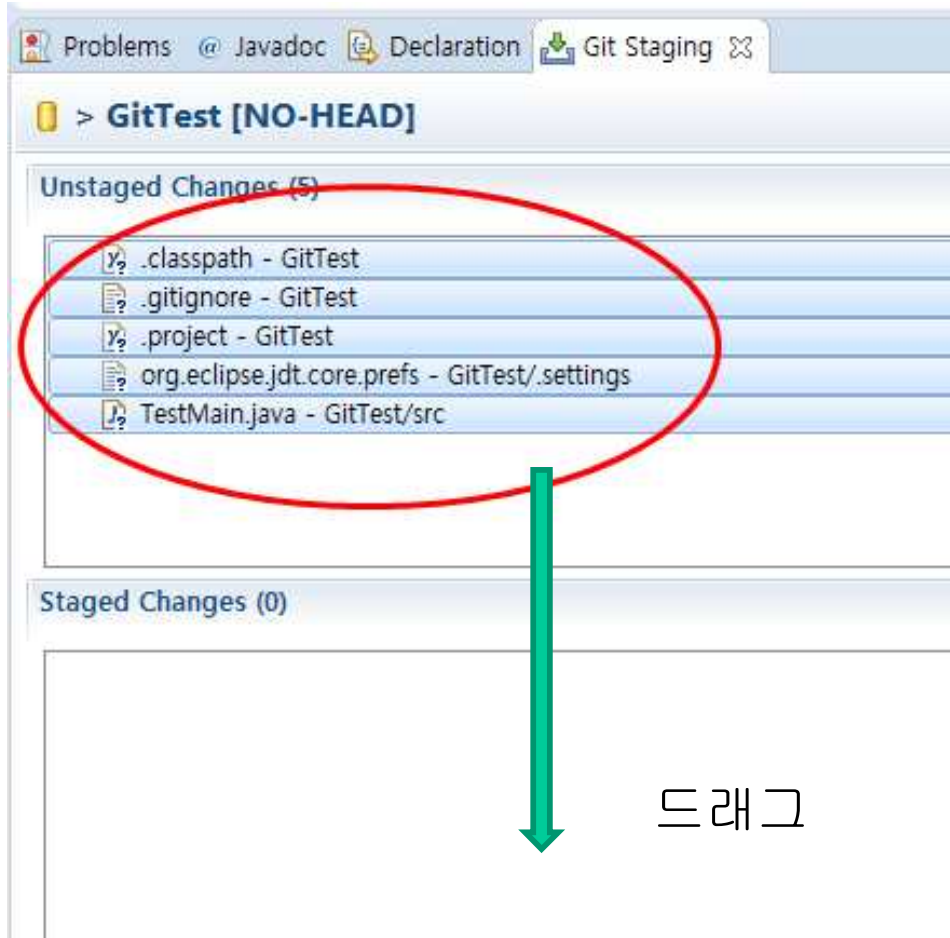
## 5. 로컬 저장소로부터 Remote 저장소로 로드하기 ( commit/push )



## □ 3) Eclipse에서 GitHub 사용

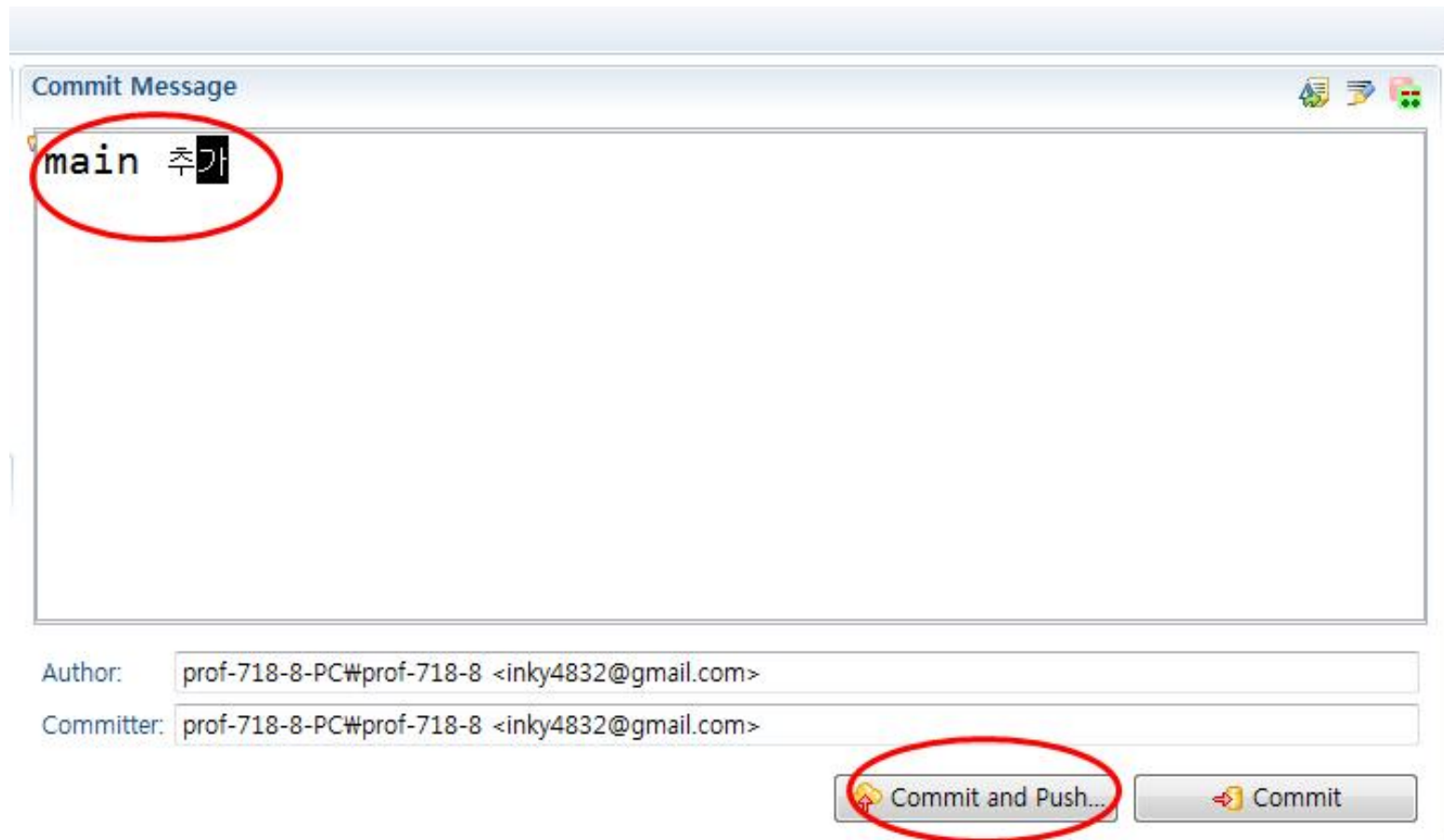


### ❑ 3) Eclipse에서 GitHub 사용



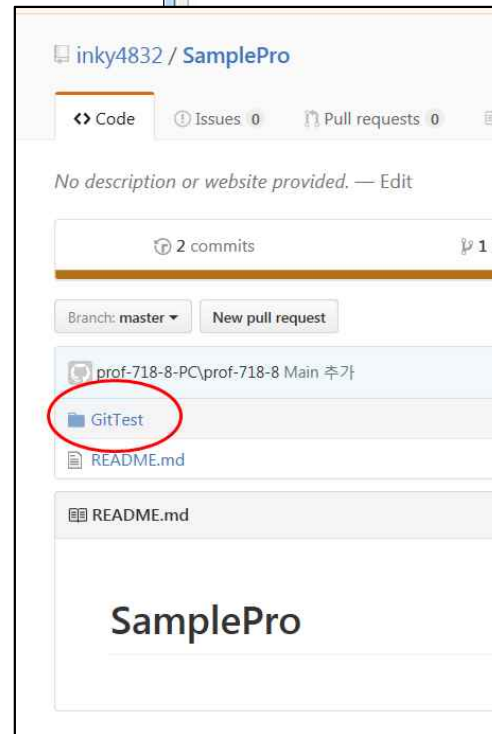
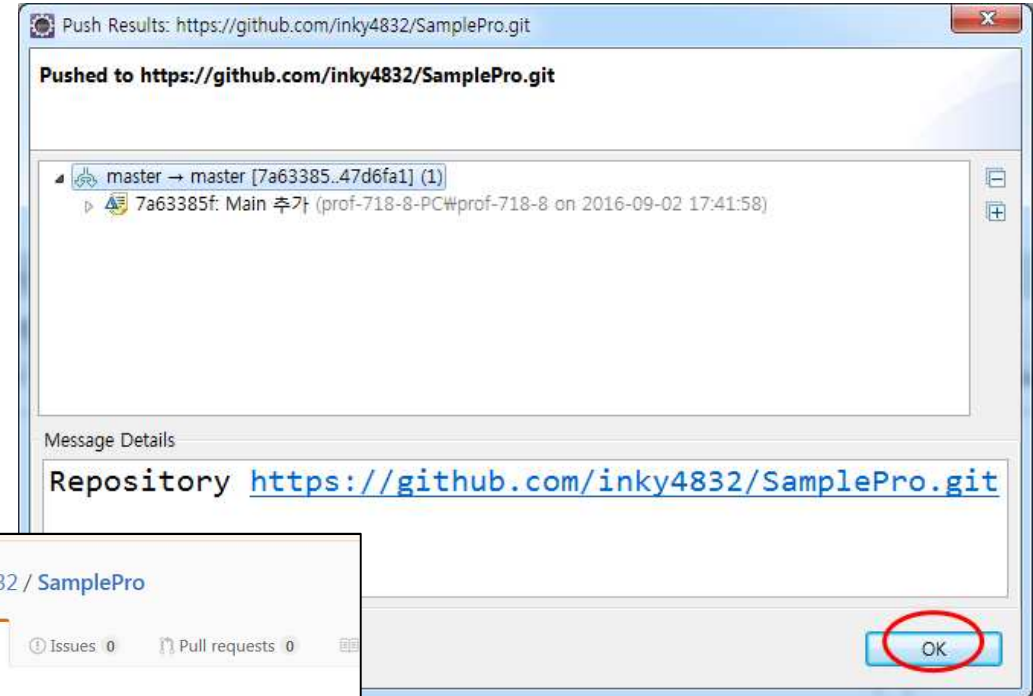
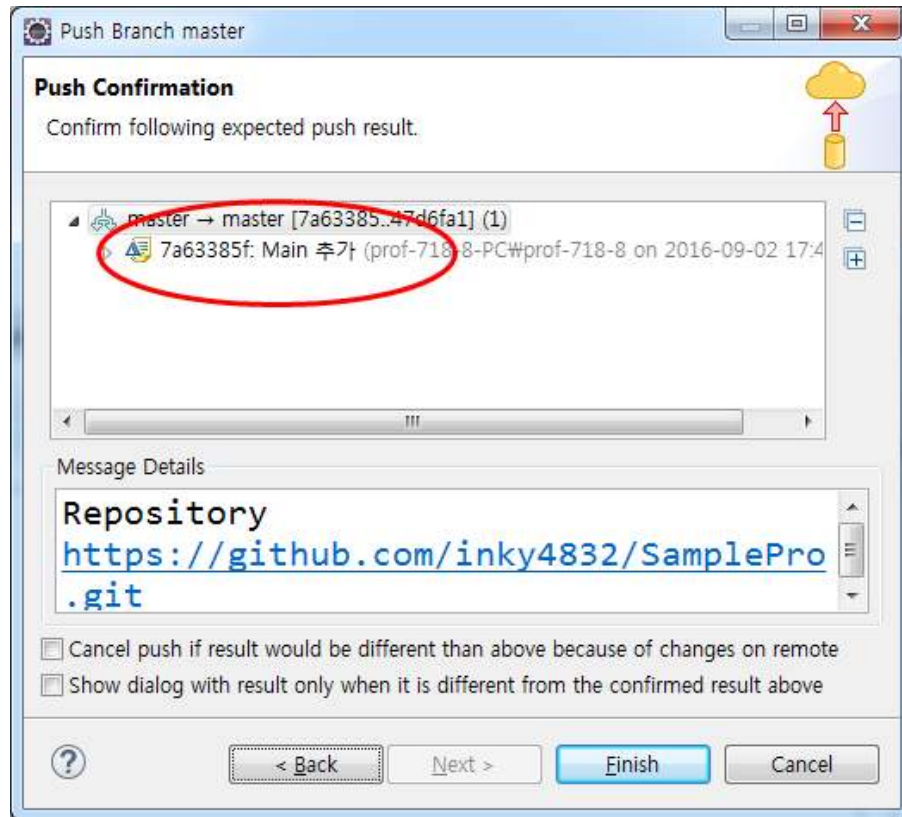
### ❑ 3) Eclipse에서 GitHub 사용

Comit Message 필수, 입력후에 Commit and Push 버튼 선택



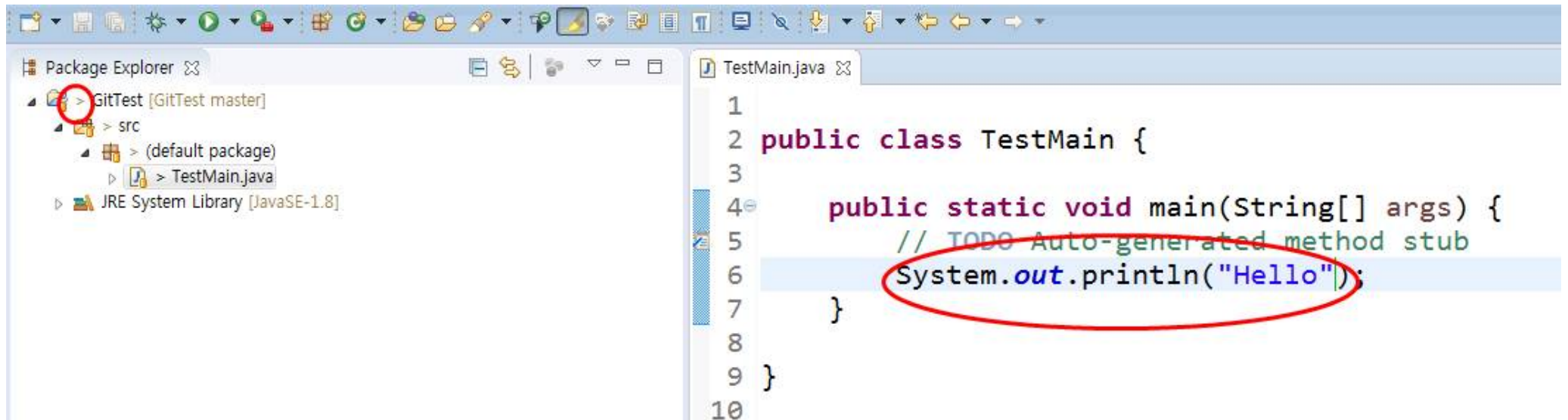


### ❑ 3) Eclipse에서 GitHub 사용

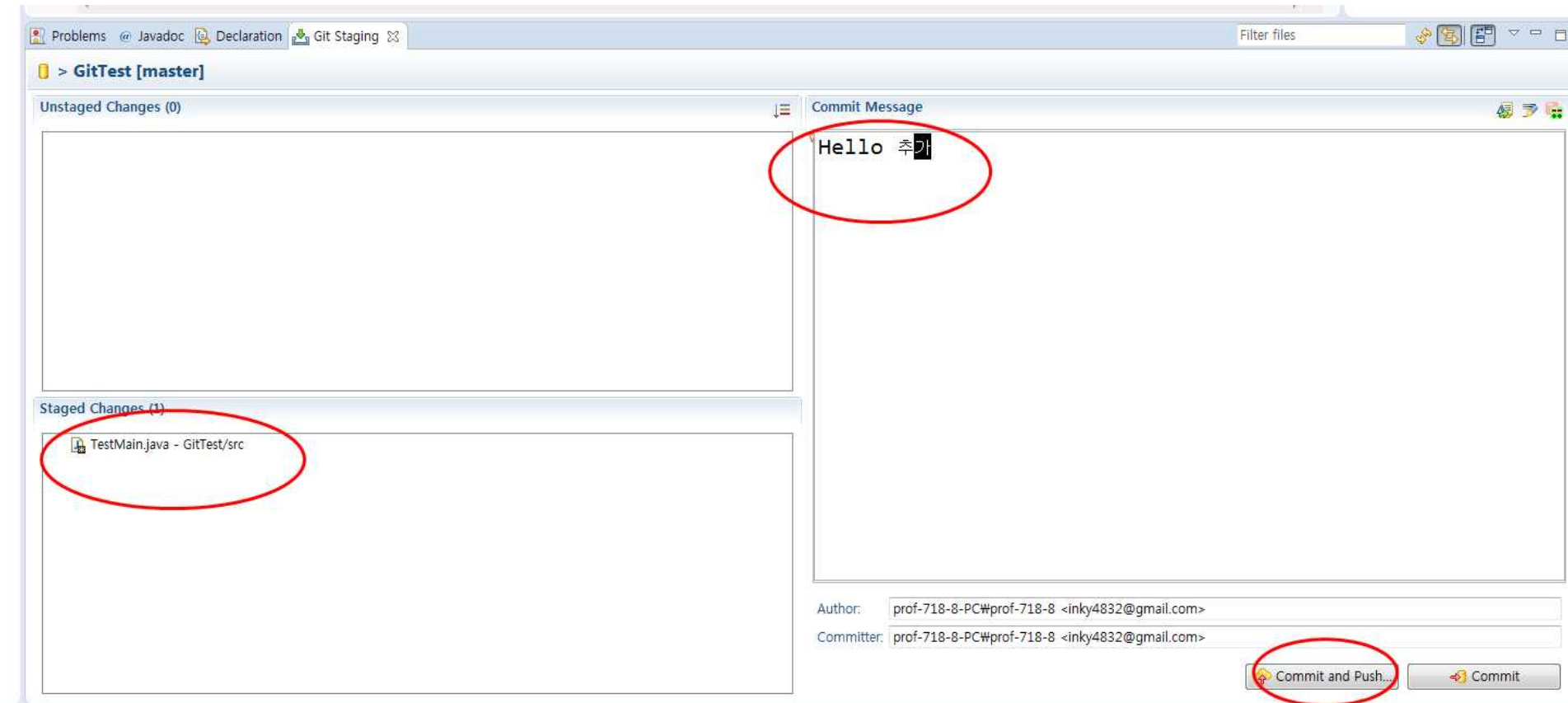


6. 이클립스에서 소스 재 수정하고 commit/push 하기

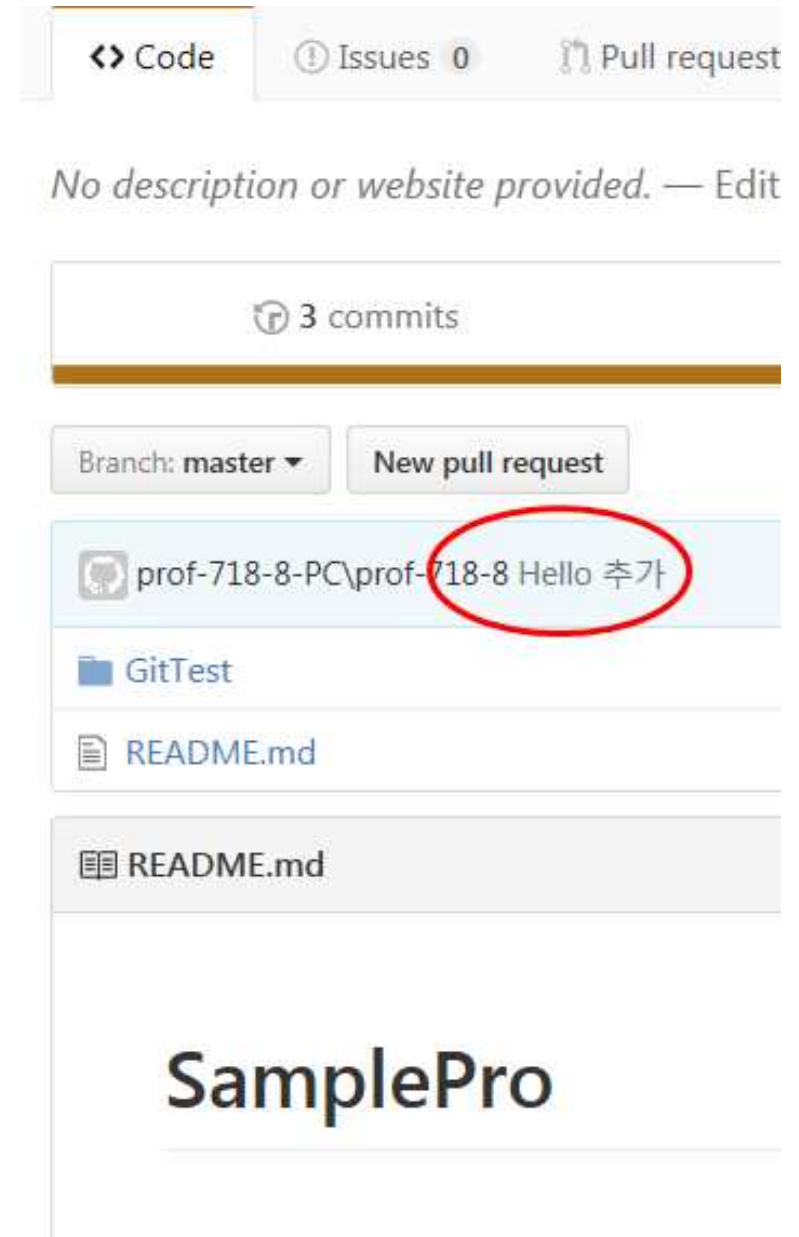
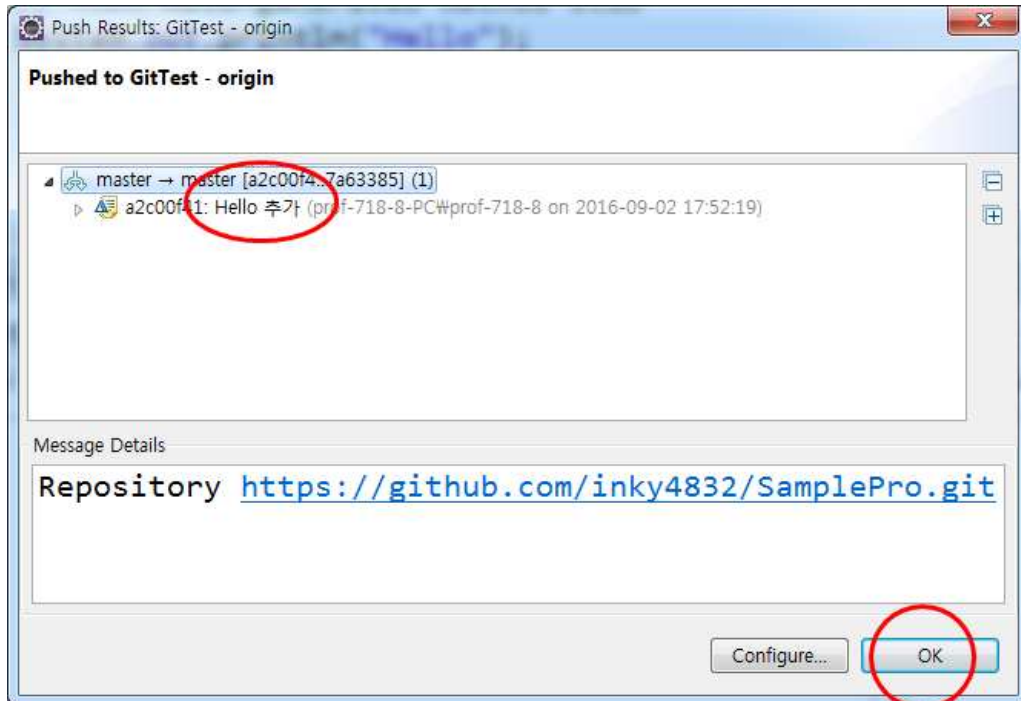
System.out.println("Hello") 추가



### □ 3) Eclipse에서 GitHub 사용



### ❑ 3) Eclipse에서 GitHub 사용



# 협업 사용

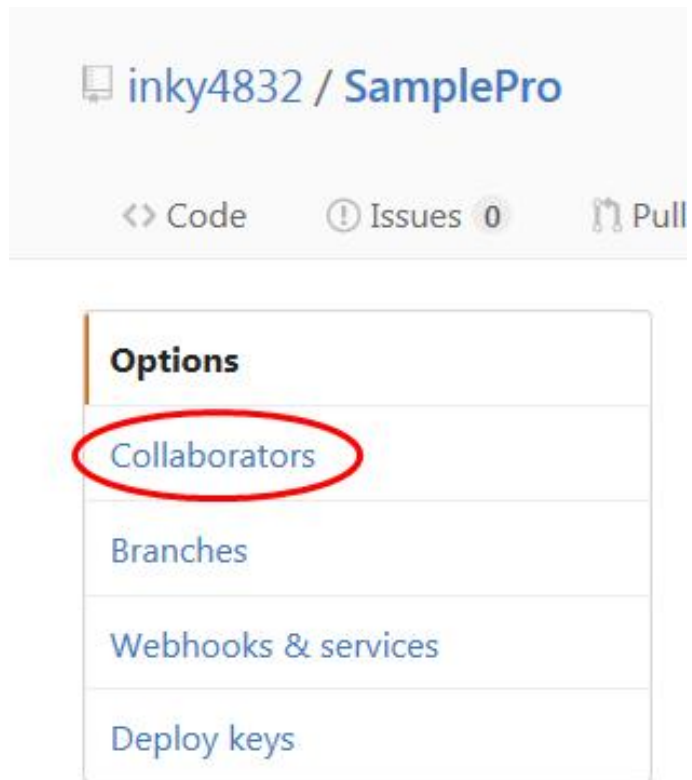


### ❑ 3) Eclipse에서 GitHub 사용

1. www.github.com 에서 조원들 회원 가입  
예> 이름:inky4833 **inky4832@naver.com** / 08XXXXXXX ← 조원 정보
2. inky4832@daum.net 조장이라고 가정한다.



### 3. Collaborators 링크를 선택한다.



### ❑ 3) Eclipse에서 GitHub 사용

4. 협업하고자 하는 사용자명 또는 이메일을 입력하여 등록한다.

The screenshot shows the GitHub repository settings for 'inky4832 / SamplePro'. The 'Collaborators' tab is selected. The search bar contains 'inky4833' and a dropdown shows 'inky4833' as a suggestion. Below, the 'Awaiting inky4833's response' status is shown with 'Copy invite link' and 'Cancel invite' buttons.

Options

- Collaborators
- Branches
- Webhooks & services
- Deploy keys

Collaborators

Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

inky4833

inky4833

Awaiting inky4833's response

Copy invite link

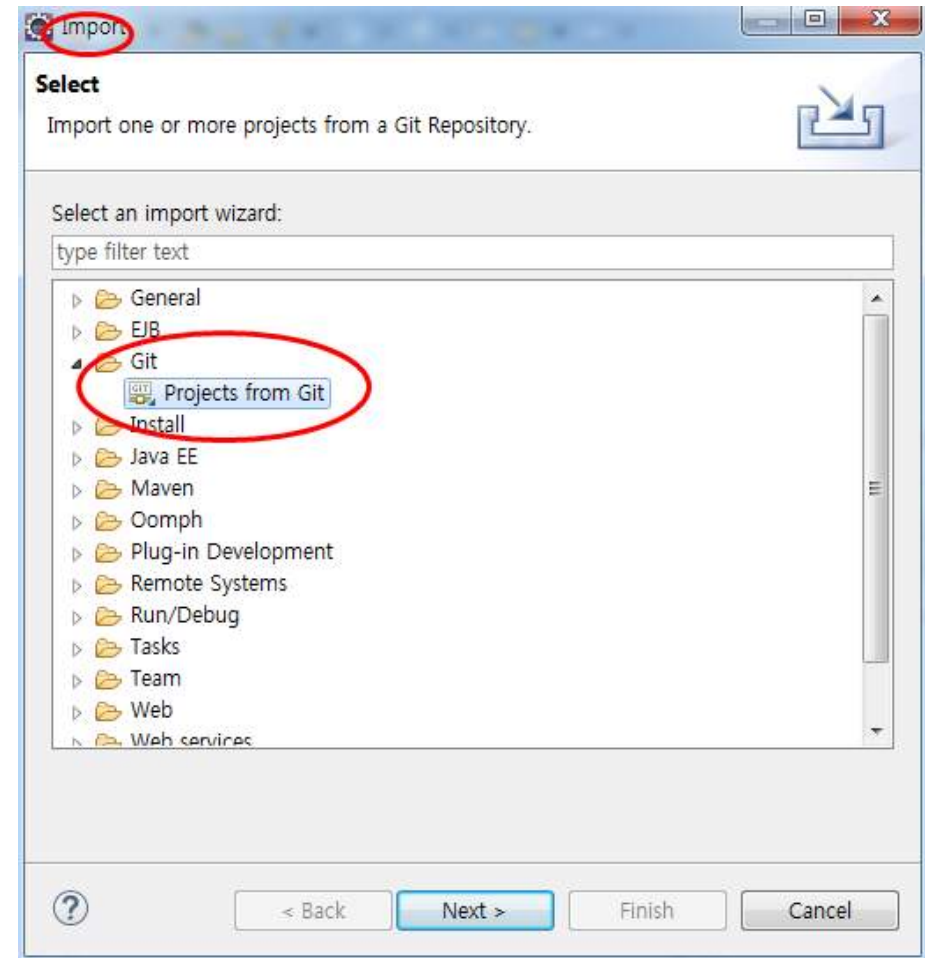
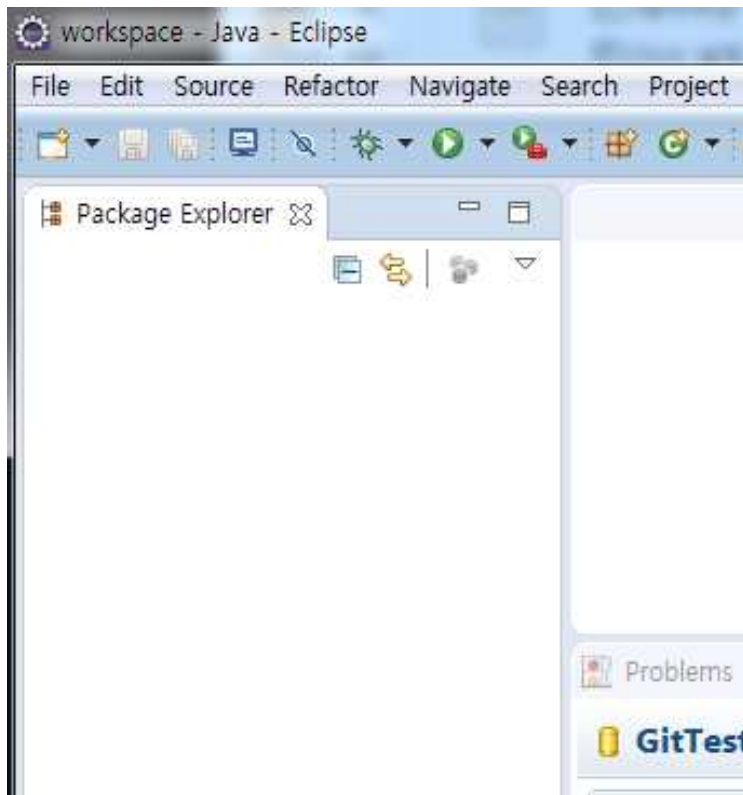
Cancel invite

Search by username, full name or email address

Add collaborator

### ❑ 3) Eclipse에서 GitHub 사용

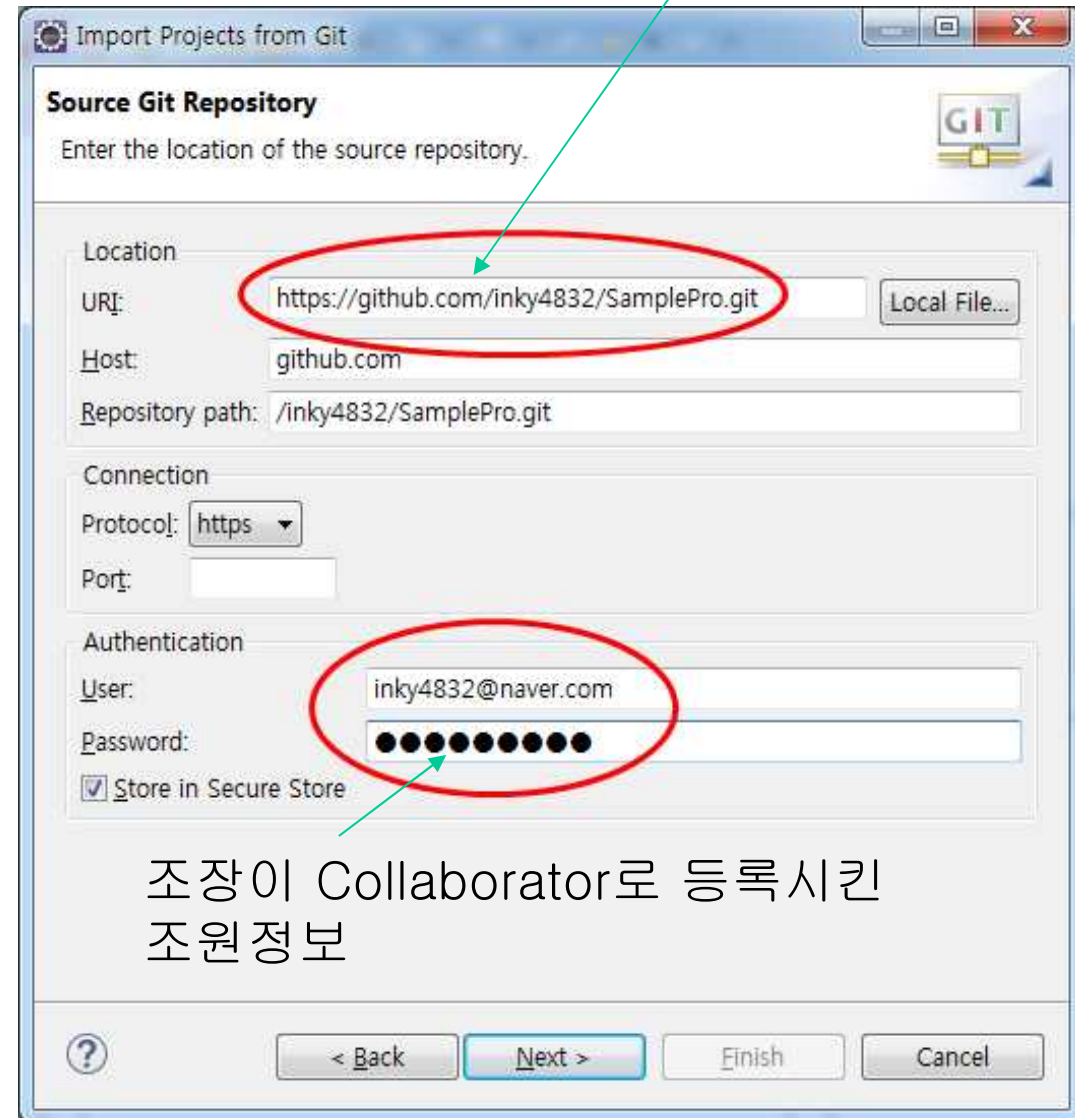
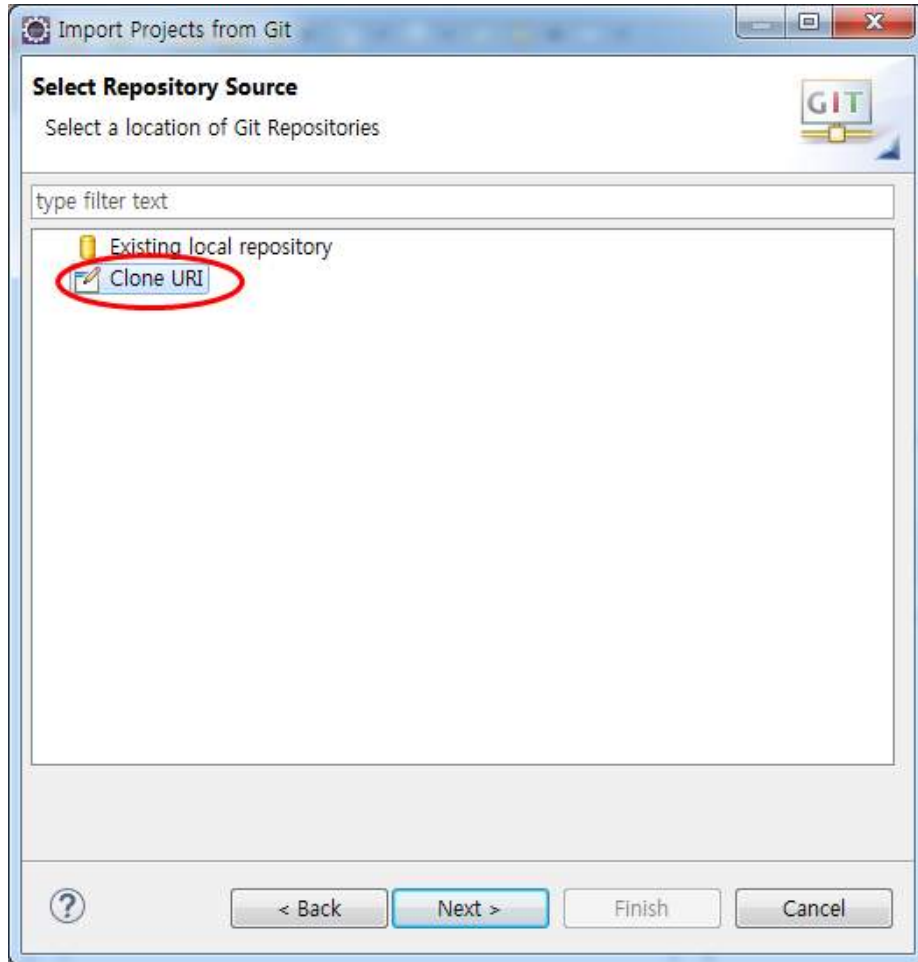
5. 조원들은 조장이 GitHub에 Push한 리소스를 조원 Eclipse로 impor한다.



### ❑ 3) Eclipse에서 GitHub 사용

6. 조장이 생성한 GitHub 주소를 설정한다.

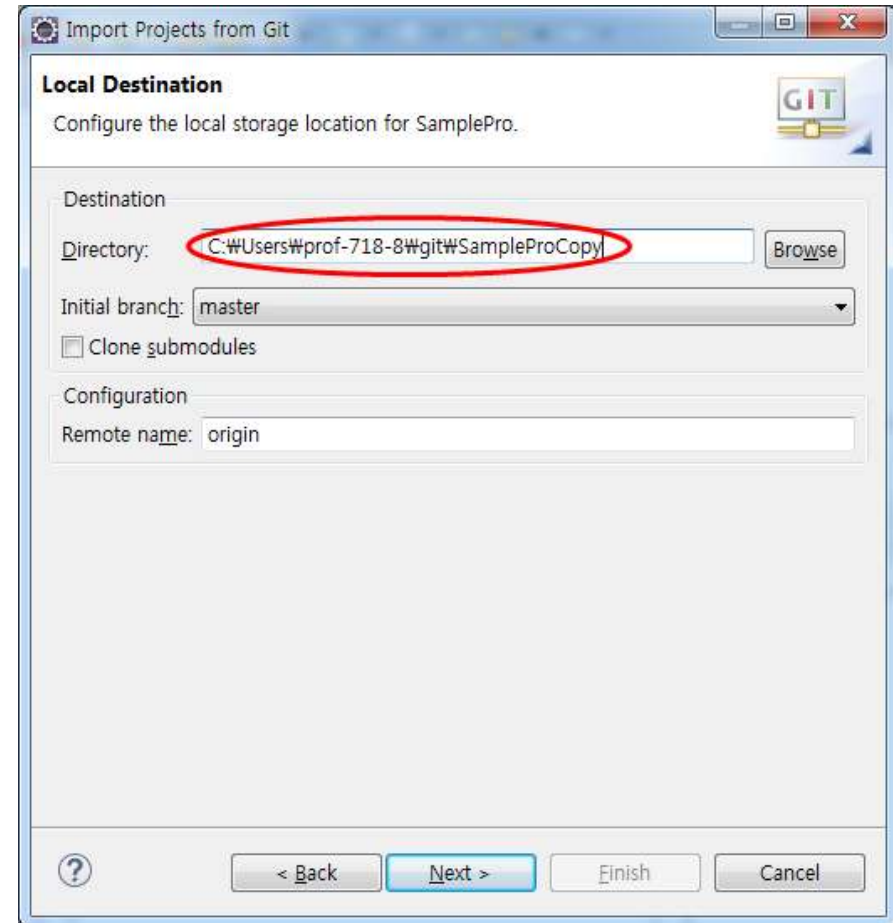
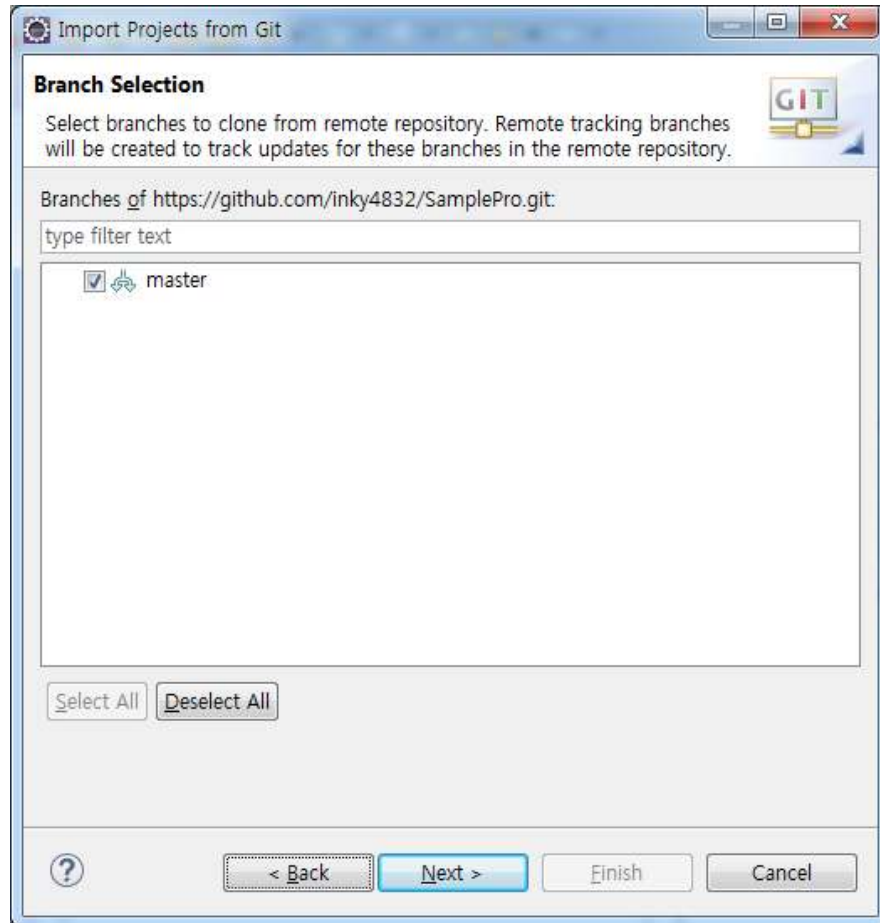
조장이 생성한 Github 주소



조장이 Collaborator로 등록시킨  
회원정보

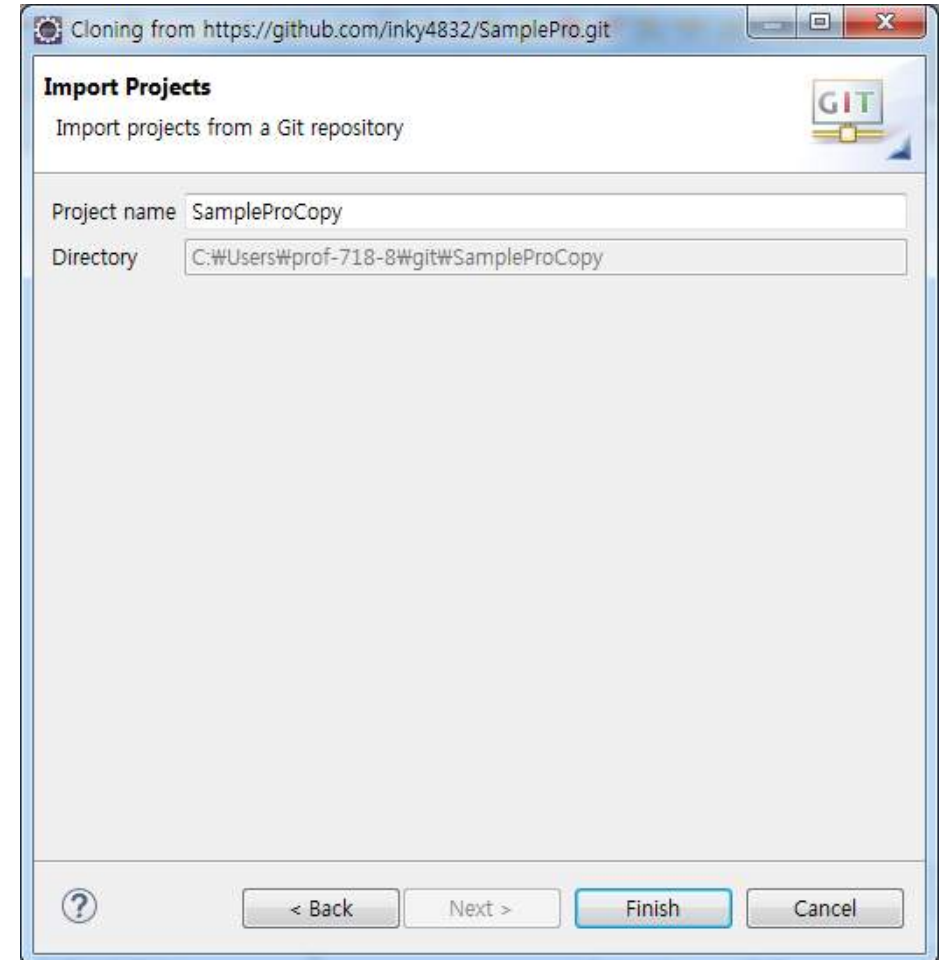
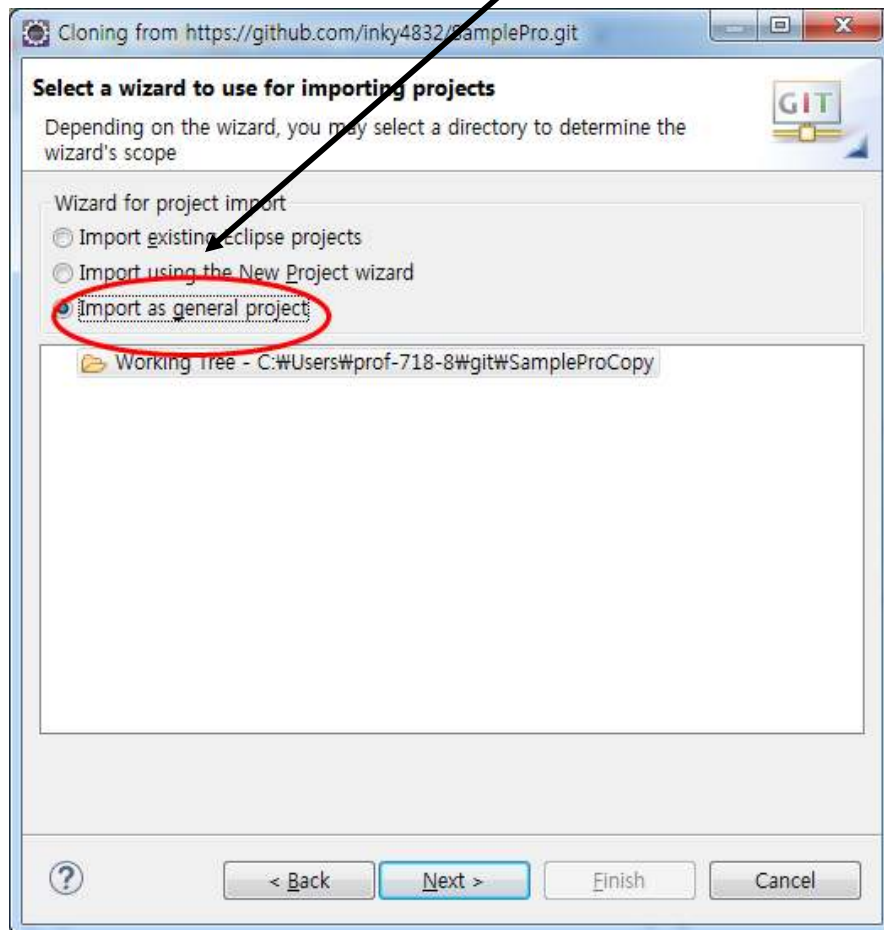


## 7. 조원의 local 저장소 설정



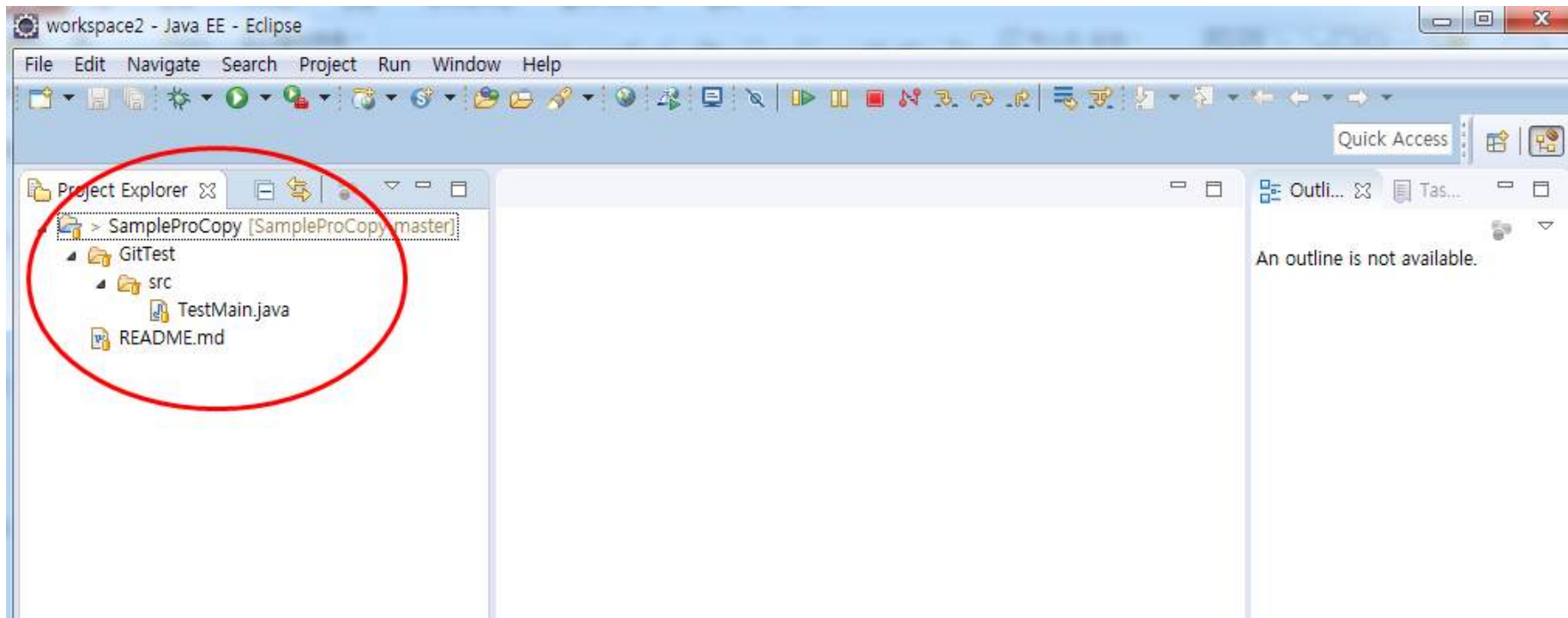
### ❑ 3) Eclipse에서 GitHub 사용

본인 환경에 알맞은 방법 선택



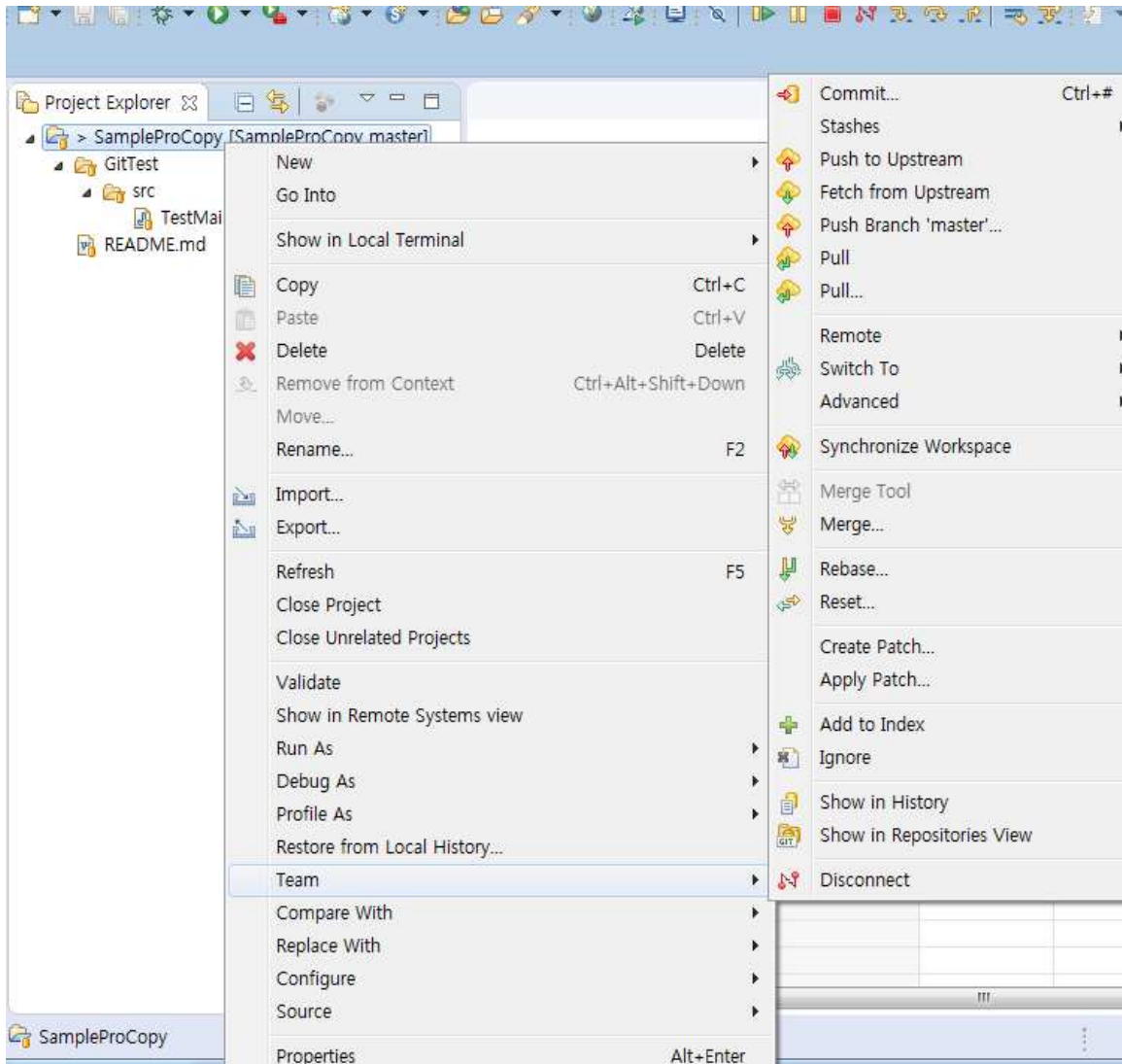
### ❑ 3) Eclipse에서 GitHub 사용

8. 조원 Eclipse로 import 된다.



## ❑ 3) Eclipse에서 GitHub 사용

### 9. 이후에는 ~~~~





**Thank you**

---